BİT-DÜZLEM DİLİMİNDE VEKTÖR KUANTALAMA

BURAK COŞKUN

Yüksek Lisans Tezi
Elektrik – Elektronik Mühendisliği
Anabilim Dalı

AĞUSTOS 2006

VECTOR QUANTIZATION
ON BIT-PLANE SLICING

BURAK COŞKUN

MASTER OF SCIENCE THESIS

Departmant Of Electrical – Electronics Engineering

AUGUST 2006

BİT-DÜZLEM DİLİMİNDE VEKTOR KUANTALAMA

Burak Coşkun

Osmangazi Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Elektrik – Elektronik Anabilim Dalı
Telekomünikasyon Bilim Dalında
Yüksek Lisans TEZİ
Olarak Hazırlanmıştır.

Danışman : Yrd. Doç. Dr. M.Atıf ÇAY

Ağustos 2006

BURAK COŞKUN ‘ un YÜKSEK LİSANS tezi olarak hazırladığı **“ BİT DÜZLEM DİLİMİNDE VEKTÖR KUANTALAMA ”** başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

Üye: Yrd. Doç. Dr. M. Atıf ÇAY

Üye: Doç. Dr. Bilginer GÜLMEZOĞLU

Üye: Yrd. Doç. Dr. Selçuk CANBEK

Fen Bilimleri Enstitüsü Yönetim Kurulu’nun ..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Abdurrahman KARAMANCIOĞLU

Enstitü Müdürü

ÖZET

İmge sıkıştırma, iletim bandı genişliğinin etkin bir şekilde kullanımı ve verilerin en az belleğe ihtiyaç duyacağı şekilde depolanması amacı için geliştirilmiştir. Böylece iletim süresi ve bilgileri saklamak için gerekli donanım en aza indirilmesi sağlanabilmektedir. İmge sıkıştırma gereksiz verilerin atılması temeline dayanmaktadır. İmge sıkıştırma tekniklerinin bir çok uygulama alanı vardır ve önemi giderek artmaktadır.

Kayıpsız sıkıştırma tekniklerinden biri de bit-düzlem kodlamadır. Görüntü bitler bazında düzlemlere ayrılır. En az değerli bit bit-düzlem 0' a ve en çok değerli bit ise bit-düzlem 7 olacak şekilde değerlerine göre sıralanır.

Kayıplı sıkıştırma tekniklerinden vektör kuantalama ise görüntü dizilerinin çerçeveleri arasındaki yüksek benzerlikten yararlanır. Bu yöntemde her çerçeve için kod kitabı bulmak yerine ilk çerçeveden elde edilen kod kitabını kullanarak ardından gelen çerçeveler için bu kod kitabı kullanılır.

Bu çalışma kayıpsız sıkıştırma tekniği olan bit-düzlem kodlamanın kayıplı sıkıştırma tekniği vektör kuantalama üzerinde uygulamasıdır. Bit düzlemi üzerinde vektör kuantalama uygulanarak imge sıkıştırma yapılmıştır. Sonuçlar, vektör kuantalanan resimlerle karşılaştırılmıştır.

Anahtar kelimeler: kod kitabı, kod vektörü, bit-düzlemi, vektör kuantalama, LBG…

SUMMARY

Image compression is developed to increase the efficiency of communication channels and to occupy less spaces in storage media. This technique helps reducing communication period, enabling more data flow and decreasing the cost of hardware for data storage. Image compression is supported by the basis of discarding unnecessary data. The importance of image compression techniques that have lots of usage areas is increasing.

One of the methods of lossless compression techniques is bit-plane compression. The image is separated with respect to planes that base of bits. Bits are arranged by bit-plane 0 corresponds to the least significant bit, conversely bit-plane 7 corresponds to the most significant bit.

Vector quantization, that is one of the lossy compression techniques, utilizes highly correlation between the frames of the image sequences. Instead of finding a new code book for each frames, the code book founded from the first frame is used for the consecutive frames by using correlation between frames.

In this thesis, bit-plane coding applies on vector quantization. Image compression is performed by applying vector quantization on bit-plane slicing. The results are compared with vector quantized images.

Key words: Code book, code vector, bit-plane, vector quantization, LBG…

TEŞEKKÜR

Bu çalışmada yardımını ve desteğini her zaman hissettiğim danışman hocam Yrd. Doç. Dr. M.Atıf ÇAY ‘a en içten teşekkürlerimi sunarım. Bu çalışma süresince bana sürekli destek olan günümüzün modern Mevlana’sı ve örnek aldığım kişi olan Şirket Müdürümüz Ekrem YALÇIN’a, her zaman beni en iyi anlayan Sn. Ali Kemal CÖRÜT’e, desteklerinden dolayı Sn. İrfan ATAMAN’ a , Araş. Gör. Kemal ÖZKAN’a ve beni bugünlere getiren aileme sonsuz teşekkürlerimi bir borç bilirim.

İÇİNDEKİLER DİZİNİ

Sayfa

İÇİNDEKİLER DİZİNİ (devam)

Sayfa

İÇİNDEKİLER DİZİNİ (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

ŞEKİLLER DİZİNİ (devam)

Şekil **Sayfa**

ŞEKİLLER DİZİNİ (devam)

Şekil **Sayfa**

TABLOLAR DİZİNİ

Tablo **Sayfa**

FORMÜLLER DİZİNİ

Formül **Sayfa**

BÖLÜM 1

İMGE SIKIŞTIRMA

İmge sıkıştırma, sayısal bir görüntünün daha az sayıda bit ile temsil edilmesidir. Bu da gereksiz verilerin atılması temeline dayanmaktadır. Matematiksel boyutuyla, 2-D piksel dizilerinin dönüştürülmesi ve ilgisiz veri kümesinde toplanmasıdır. Dönüşüm öncelikle verinin saklanmasında yada görüntünün iletiminde uygulanmaktadır.

İmge sıkıştırma değişik uygulamalarda karşımıza çıkar. Örneğin; televideo konferansta, uzaktan algılamada, hava tahmini için uydu görüntülerinde ve yer kaynağı uygulamalarında, doküman ve tıp alanında, FAX teknolojisinde, askeriyede uzaktan kumanda edilebilir araçlarda, uzayda ve tehlikeli atık kontrol uygulamalarında kullanılmaktadır. (Gonzalez,R.G and Woods,R.E., 2001)

İmge sıkıştırma görüntü işlemenin en önemli alanlarından biridir. İmge sıkıştırma, bilgisayarların gelişmesi, çoklu medya uygulamaların giderek artması ve internetin günlük hayatta kullanılma oranının çoğalmasına bağlı olarak önemi daha da artmıştır. Bunlara ek olarak gelişen video teknolojisi yeni, daha iyi, ve daha hızlı görüntü sıkıştırma algoritmalarının geliştirilme ihtiyacını ortaya çıkarmıştır. Sıkıştırma algoritmaları iki boyutlu (2-D) durağan resimlerle (still) başlamıştır. Durağan resmin sıkıştırılması alanındaki gelişmeler aynı zamanda hareketli resimlerin (video / motion picture) sıkıştırılması uygulamalarının geliştirilmesine de yardım eder. Bundan dolayı görüntü sıkıştırma algoritmaları iki boyutlu tek çerçevelik görüntüler üzerinde yoğunlaşırlar.

Kayıplı görüntü sıkıştırma, görüntü dosyalarındaki önemsiz bilgileri atarak boyutunu küçültmektir. Boyutu azaltılmış dosyaya sıkıştırılmış dosya (compressed file), bu dosyayı kullanarak geri elde edilmiş dosyaya açılmış dosya (decompressed file) adı verilir. Herhangi bir sıkıştırma algoritması uygulanmamış orijinal görüntüye ise sıkıştırılmamış dosya (uncompressed file) denir. Orijinal, sıkıştırılmamış görüntü, ile sıkıştırılmış görüntünün bellekte kapladıkları boyutlarının arasındaki orana da sıkıştırma oranı adı verilir. Sıkıştırma oranı Formül 1.1'deki gibidir.

$$\text{Sıkıştırma oranı} = \frac{\text{Orijinal Görüntü Büyüklüğü}}{\text{Sıkıştırılmış Görüntü Büyüklüğü}} \qquad (1.1)$$

Genellikle Büyüklük $_U$: Büyüklük $_C$ şeklinde yazılır.

Diğer bir gösterim şekli ise piksel başına düşen bit (bits per pixel) sayısıdır. N x N boyutlarındaki bir görüntüde bu oran Formül 1.2'de gösterilmiştir.

$$\textit{Piksel başına düşen bit sayısı} = \frac{\textit{BitSayısı}}{\textit{PikselSayısı}} = \frac{(8)\textit{BaytSayısı}}{NxN} \qquad (1.2)$$

Veri sıkıştırma, terminolojide piksel veya çerçeveler arasındaki tekrarlıkları atıp sadece önemli olan bilgilerin saklanması ve verinin sayılı bir kısmının iletilmesi şeklinde tanımlanır. İletim sistemlerinin bant genişliğinin darlığından dolayı verilerin taşınması ve bilgisayarda saklanması için sıkıştırma işlemi gerçekleştirerek boyutlarının azaltılması gerekmektedir. Başarılı bir sıkıştırmanın anahtarı tanımında da belirtildiği gibi önemsiz bilgilerin atılmasıdır. İki temel görüntü sıkıştırma yöntemi vardır. İlk yöntemde, geri elde edilen yöntemde, veri kaybı yoktur. Bundan dolayı bu sıkıştırma yöntemine kayıpsız (lossless) sıkıştırma denir. Kompleks görüntülerde bu yöntemle

elde edilen sıkıştırma oranı sınırlı olup genellikle 2:1 ile 3:1 arasındadır. Ancak, sadece metinden oluşan görüntülerde kayıpsız sıkıştırma yöntemi daha fazla bir sıkıştırma oranı sağlar. İkinci sıkıştırma yöntemi ise kayıplı (lossy) sıkıştırma olarak isimlendirilir. Çünkü tekrar elde edilen görüntü orijinal görüntünün tamamen aynısı değildir. Görüntülerin kompleks yada basit olmasına bağlı olarak sıkıştırma oranı 10 ila 200 arasında değişir. Basit görüntülerde veya düşük kaliteli sonuçlarda sıkıştırma oranı 100 ila 200 arasında olabilmektedir. (Özkan, K.,2000)

Tablo 1.1'de iki sıkıştırma şekli karşılaştırılmıştır.

	Kayıpsız Sıkıştırma	**Kayıplı Sıkıştırma**
Orijinal görüntünün geri elde edilmesi	Orijinal görüntü geri elde edilir	Orijinal görüntü geri elde edilemez
Sıkıştırma oranı	Sıkıştırma oranı azdır	Sıkıştırma oranı fazladır
İşlem karmaşıklığı	Daha azdır	Daha fazladır
Uygulama alanları	Her veriye	Görüntülere, seslere

Tablo 1.1 Kayıplı ve kayıpsız sıkıştırmanın karşılaştırılması

Sıkıştırma algoritmalarında görüntülerdeki tekrarlılık büyük bir avantajdır. Bir görüntüde, sıkıştırmada faydalanabileceğimiz üç tekrarlılık vardır; kodlama tekrarlılığı, pikseller arası tekrarlılık, insan gözünün ürettiği tekrarlılık. Kodlama tekrarlılığı görüntüyü temsil eden verilerin optimum olarak kodlanmaması olarak görülür. Örneğin, biz piksel başına 8 bit kullanırsak 256 gri seviye değerleri elde ederiz, ancak gerçek görüntü yalnız 16 gri seviye sahip olabilir. Bundan dolayı optimum kodlama yapmış olmayız. Çünkü bu görüntü için piksel başına 4 bit yeterlidir. Bir görüntüde pikseller arasındaki tekrarlılıkta görülür, çünkü gerçek görüntülerde birbirine yakın olan pikseller arasında yüksek oranda ilinti mevcuttur. Bunun sonucu olarak görüntülerde ışıklandırma seviyeleri birden değişme göstermez ve gerçek görüntülerde birbirine yakın olan piksel değerleri çok yakın olur. Video veya hareketli resimlerinin

çerçeveleri arasında, aynı pikseller arasında olduğu gibi büyük bir benzerlik mevcuttur. 3. tekrarlılık ise gözün fiziksel yapısıdır. Bazı bilgiler insan gözü için daha önemlidir. Göz örneğin düşük frekanstaki bilgileri yüksek frekanstaki bilgilere göre daha iyi algılar. (Özkan, K., 2000)

1.1 Doğruluk Kriteri (Fidelity Criteria):

Görüntü sıkıştırma algoritmalarının temel amacının gerekli bilgilerin minimuma indirilmesi olduğunu daha önce söylemiştik. Hangi bilgilerin daha önemli olduğunu bulmak için görüntü doğruluğunun tanımının yapılması gerekir.

Doğruluk kriteri iki sınıfa ayrılır.

A) Nesnel doğruluk kriteri (objective fidelity criteria)
B) Öznel doğruluk kriteri (subjective fidelity criteria)

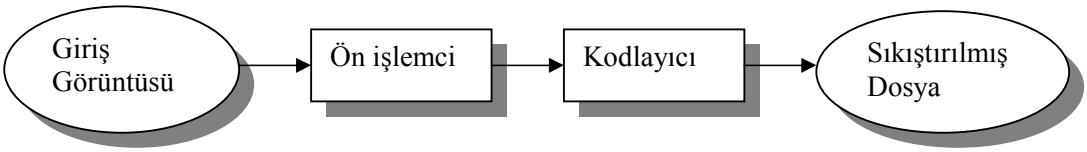
Nesnel doğruluk kriteri, sayısal sinyal işleme ve bilgi teorisinde de olduğu gibi tekrar oluşturulan görüntü ile orijinal görüntü arasında ne kadar hata olduğunun bulunmasıdır. Nesnel kriter aslında çok kullanılmasına karşın görüntünün kalitesi ile bağlantılı değildir. Bununla birlikte aynı görüntünün iki farklı versiyonunun karşılaştırılmasında göreceli bir ölçüm için kullanılır. Öznel doğruluk kriteri ise görüntü kalite derecesi (qualitative scale) olarak tanımlanır. Bu derecelendirme ise insanlar tarafından yapılan testlerle belirlenir. Öznel ölçümlemede test görüntülerinin ve deney düzeninin çok iyi yapılması gerekir. (Gonzalez,R.C and Woods,R.E.,2001)

1.2 İmge Sıkıştırma Modelleri:

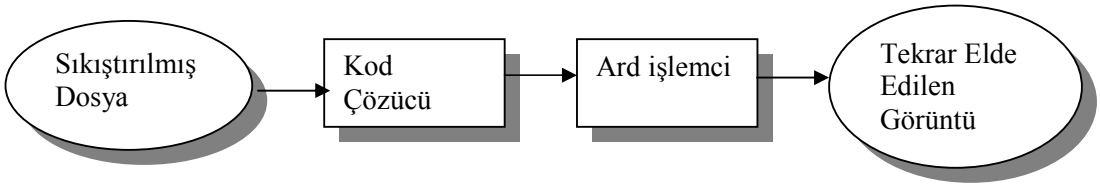
Bir sıkıştırma sistemi iki farklı bloktan oluşur. Bunlar ‘kod çözücü (encoder)’ ve ‘kodlayıcı (decoder)’ dır. Giriş verisinin semboller kümesinden oluşan giriş

görüntüsü kod çözücüye girer. Kanalda iletimi olduktan sonra kodlanmış görüntüsü kodlayıcıya girer ve çıkış görüntüsü $f(x,y)$ oluşur. $f(x,y)$, f(x,y)-orijinal görüntüsü'nün aynısı olabilir yada olmayabilir. (Gonzalez,R.C and Woods,R.E.,2001)

Sıkıştırma sistem modeli iki kısımdan oluşur; sıkıştırıcı ve sıkıştırmayı çözücü. Sıkıştırıcı, ön işlem katı (preprocessing state) ve kodlayıcı katından (encoding state) oluşur. Şekil 1.1' de gösterilmiştir. Sıkıştırmayı çözücü ise kod çözücü katını (decoding state) arka işlemci katı (post processing state) izler. Şekil 1.2 gösterilmiştir.



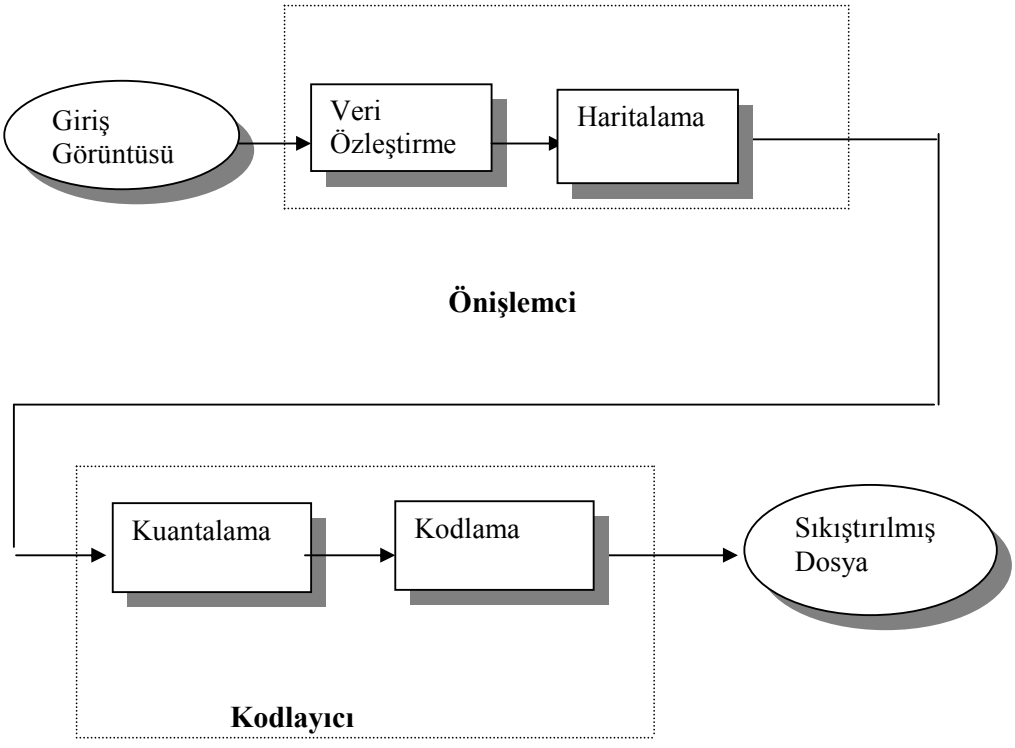
Şeki1 1.1 Sıkıştırma katı



Şekil 1.2 Sıkıştırmayı çözme katı

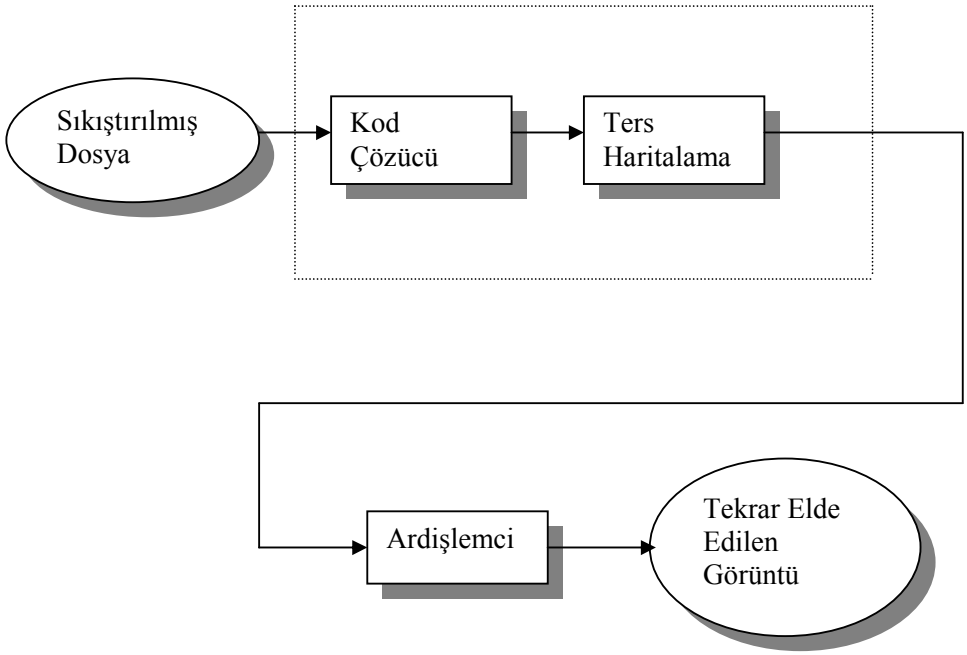
Kodlayıcıdan önce, uygulamalara göre değişmesine rağmen, bir dizi işlemler ile görüntüyü kodlama işlemine hazırlamak gerekir. Sıkıştırılmış görüntünün açılması sırasında ise bu ön işlemler ters şekilde ard işlemler olarak yapılır.

Sıkıştırma kısmı aşağıdaki Şekil 1.3'de görüldüğü gibi kısımlara ayrılır. İlk kısım verinin azaltıldığı kattır. Burada görüntü verisinin gri seviyesinin azaltılması ve / veya uzamsal nicemleme (spatial quantization) veya imge pekiştirme (image enhancement) kullanılabilmesi için düzenlenmesi işlemleri yapılır (örneğin,gürültülerin yok edilmesi). İkinci işlem ise haritalama (mapping) işlemidir. Bu kısımda orijinal görüntü daha iyi bir sıkıştırmanın yapılabildiği başka bir matematiksel boyuta haritalanır. Sonra kodlama işlemlerindeki kuantalama katı ile kesintisiz veriler kesintili verilere çevrilir. Daha sonra kodlamanın yapıldığı kodlama katı gelir. Sıkıştırma algoritmalarında genellikle bütün bu katlar bulunur. (Gonzalez,R.C and Woods,R.E.,2001)



Şekil 1.3 Sıkıştırıcı katının ayrıntıları

Sıkıştırıcıyı çözücü katı da aşağıdaki Şekil 1.4' de görüldüğü gibi çeşitli kısımlara ayrılır. İlk katta, kod çözücü katı, sıkıştırılmış görüntü orijinal görüntüye kod çözme işlemi yardımıyla çevrilir. Sonra bu değerler ters haritalama yöntemiyle orijinal tanım kümesine haritalanır. Son olarak da ardişlemler ile orijinal görüntü elde edilmeye çalışılır. (Özkan, K., 2000)



Şekil 1.4 Sıkıştırmayı çözücü katının ayrıntıları

1.3 Kayıpsız Sıkıştırma Metodu (Lossless or Error Free Compression Methods):

Kayıpsız sıkıştırma metotları pek çok alanda uygulanır. Kayıplı sıkıştırmanın kullanılmasının yasak olduğu yerlerde yani, medikal yada iş dokümanları da kayıpsız sıkıştırmanın uygulama alanlarıdır. (Gonzalez, R.C and Gray,R.M., 1992)

Kayıpsız sıkıştırma metotları bazı görüntü uygulamaları için gereklidir. Örneğin, medikal görüntülerin herhangi bir bilgi kaybı olmaksızın sıkıştırılması lazımdır. Kayıpsız sıkıştırmanın teorisi haberleşme ve bilişim kuramından gelir. Görüntüdeki ortalama bilginin ölçülmesi gerekir ki buna da entropi (entropy) adı verilir. N x N boyutlarındaki bir görüntüdeki entropi ise Formül 1.3 ile bulunur.

$$Entropi = \sum_{p=0}^{L-1} p_i \log_2(p_i) \qquad (1.3)$$

Burada ,

P_i : i. gri seviyesinin olasılığı $\frac{n_k}{N^2}$

n_k : k gri seviyesine sahip toplam piksel sayısı

L : gri seviye sayısı (örneğin 8 bit için 256)

Teorik olarak bu ölçüm, piksel başına düşen ortalama bit sayısının minimum olması o görüntüye daha iyi bir kodlama yapılabileceğini gösterir. Entropi 0 ile $\log_2(L)$ arasındadır ve Formül 1.4' de gösterilmiştir.

$$0 \leq \text{Entropi} \leq \log_2(L) \qquad (1.4)$$

Görüntüde çok rasgelelik (more randomness) varsa, yani gri seviyeleri düzgün dağılmamışsa bu görüntüyü temsil etmek için daha fazla bite ihtiyaç vardır. Bu da bilginin bağımsız olduğunu, her pikselin birbirine az benzediğini gösterir. Kodlama kuramında bu olay şu şekilde açıklanır. Eğer veriyi daha az bit ile temsil etmek istiyorsak verinin birbirine çok benzemesi gerekir.

Entropi ayrıca kodlamanın performansını ölçmek için kullanılır. Kodlamadaki piksel başına düşen ortalama değer Formül 1.5' deki gibidir.

$$L_{ave} = \sum_{p=0}^{L-1} l_i p_i \tag{1.5}$$

Burada ,

l_i = i gri seviyesini kodlamak için kullanılan bit boyutu

p_i = histogram, i. gri seviyesinin olasılığı

L_{ave} ne kadar küçük olursa, o kadar iyi kodlama yapıldığını gösterir. (Özkan, K., 2000)

1.3.1 Değişken Boylu Kodlama (Variable Length Coding):

Kayıpsız sıkıştırmaya en basit yaklaşım gereksiz kodlamanın azaltılmasıdır. Gereksiz kodlama bir görüntüdeki gri seviyelerinin doğal ikilik sistem kod çözücüsüdür. Değişken Boyutlu Kodlama dört çeşittir. Bunlar Huffman Kodlama, Run-Length Kodlama, Lempel-Ziv-Welch Kodlama, Aritmetik Kodlama ve Bit Düzlem Kodlamadır.

1.3.1.1 Huffman Kodlama :

Gereksiz kodlamanın kaldırılmasında kullanılan en popüler teknik Huffman Kodlamadır.

Gri seviye değerlerinin istatistikleri çıkartılarak Huffman algoritması yardımıyla mümkün olabilen minimum sınıra, idealde L_{ave} = Entropi, sahip kod üretilebilinir. Bu metodun sonucu olarak değişken uzunlukta kod (variable length code) elde edilir. Başka bir ifadeyle, kod kelimelerin uzunlukları eşit değildir. Kompleks görüntülerde yalnız Huffman kodlamayı kullanırsak veriyi %10 ile %50 (1.1:1 ile 1.5:1) arasında azaltır. Ama basit görüntülerde bu oran 2:1 veya 3:1 olabilmektedir. (Gonzalez, R.C and Gray,R.M., 1992)

1.3.1.2 Aritmetik Kodlama :

Aritmetik kodlamada kod ile piksel değeri arasında direkt bir bağlantı yoktur. Aritmetik kodlamada giriş verileri 0 ila 1 arasında kayan noktalı (floating point) sayıya transfer edilir. Bu kodlamada verinin olasılık dağılımını (probabilty distribution) kullanır, bundan dolayı teorik olarak entropi ile sınırlandırılmış maksimum sıkıştırma oranını sağlar. (Gonzalez, R.C and Gray,R.M., 1992)

1.3.2 Run – Length Kodlama :

Run – length kodlama görüntü sıkıştırma metodu aynı gri seviye değerine sahip olan komşu piksellerin sayılmasıyla yapılır. Bu sayma işlemine run-length adı verilir. Bu sayılan değerler kodlanır ve saklanır. (Gonzalez, R.C and Gray,R.M., 1992)

1.3.3 Lempel – Ziv – Welch Kodlama :

Lempel – Ziv – Welch veri dizgilerini (strings of data) kodlamada kullanır. Görüntüler için bu veri dizgileri arka arkaya gelen piksel değerleridir. Bir tane tablo yapılır. Bu tabloda dizgiler ve onların kodları vardır. Yeni bir dosya okunduğu zaman dizgi tablosu güncellenir. Eğer yeni bir dizgi gelmiş ise ona yeni bir kod üretilir. Eğer yeni dizgi tabloda hali hazırda var ise bu dizginin kodu sıkıştırılmış dosyaya yazılır. (Gonzalez, R.C and Gray,R.M., 1992)

1.3.4 Bit Düzlem Kodlama (Bit Plane Coding):

Bu teknik tek renkli veya renkli görüntüyü ikili görüntüler serisine ayırır ve her ikili görüntüyü sıkıştırır. Bit Düzlem kodlama ile ilgili daha ayrıntılı bilgi ilerideki bölümlerde verilecektir. (Wang, Z. and Bovik A.C, 2002)

1.4 Kayıplı Sıkıştırma Metodu (Lossy Compression Methods) :

Kompleks görüntülerde yüksek bir sıkıştırma oranı istiyorsak kayıplı sıkıştırma metodunu kullanmamız gerekir. Kayıplı sıkıştırma da görüntü kalitesi ile sıkıştırma derecesi arasında ters orantı vardır. Sıkıştırma oranı artıkça görüntü kalitesi düşer. İmge pekiştirme ve onarımı işlemleri ile tekrar elde edilen görüntü kalitesi artırılır. Kayıplı sıkıştırma metotlarında parametreler kullanıcı tarafından ayarlanabilmekte, bu da sıkıştırma oranını ve görüntünün kalitesini değiştirmektedir. Basit görüntülerde yüksek bir sıkıştırma oranı elde edilse de bu oran görüntüye bağlıdır. Kayıplı sıkıştırma hem uzamsal hem de dönüşüm alanında (spatial / transform domain) yapılabilir. (Gonzalez, R.C and Gray, R.M., 1992)

1.4.1 Gri Seviyeli Run Length Kodlama (Gray Level Run Length Coding) :

Run-length kodlama tekniği kayıpsız sıkıştırmada olduğu gibi kayıplı sıkıştırmada da kullanılır. Kayıplı sıkıştırmada gri seviye sayısını azaltıp kayıpsız sıkıştırmada kullanılan tekniğin aynısı uygulanır. (Gonzalez, R. C. and Gray, R. M., 1992)

1.4.2 Blok Esrime Kodlama (Block Trancation Coding):

Blok esrime kodlama görüntüyü bloklara bölerek, her bloktaki gri seviyesini azaltır. Bu azaltma işlemi her bloktaki alt görüntülerin istatistiklerine bakarak kuantalama işlemi ile gerçekleştirilir. Kuantalamanın derecesi ise belirtilmiş hata kriterini minimum yapacak şekilde seçilir. (Gonzalez, R.C and Gray,R.M., 1992)

1.4.3 Vektor Kuantalama (Vector Quantization) :

Vektör kuantalama vektörlerin daha az boyuta sahip olan başka vektörlere haritalanma işlemidir. Görüntü kodlamada vektörler daha küçük alt görüntülere veya bloklara ayrılır. Vektör kuantalama ile ilgili daha ayrıntılı bilgi Bölüm 3' de verilecektir. (Gerso, A. and Gray, R.M, 1992)

1.4.4 Farkları Öngörücü Kodlama (Differential Predictive Coding) :

Farkları öngörücü kodlama bir sonraki piksel değerini önceki piksel değerine bakarak öngörme olayıdır. Gerçek değerle öngörülen değer arasındaki fark kodlanarak gerçekleştirilir. Yüksek bağımlı görüntülerde bu tekniğin avantajları çoktur. Çünkü bu tür görüntülerde komşu pikseller arasındaki fark az olmakta, dolayısıyla daha az sayıda bit gerekmektedir. (Gonzalez, R.C and Gray,R.M., 1992)

1.4.5 Dönüşüm Kodlama (Transform Coding) :

Bilindiği üzere sayısal görüntü pikselleri birbirine yüksek oranda bağımlıdır. Diğer bir ifadeyle, görüntü pikselleri arasında yüksek bir oranda tekrarlılık mevcuttur. Görüntü sıkıştırmanın amacı da bu tekrarlılıkları azaltıp görüntüyü daha az bit sayısı ile temsil etmektir. Bu tekrarlılığı azaltmanın bir yolu öngörme yöntemidir. Öngörmeli kodlamada asıl piksel ile öngörülen piksel arasındaki fark kodlanmaktadır. Bu fark pikselin değerinden daha azdır.

Dönüşüm kodlama ise tekrarlılığı azaltmanın bir başka yoludur. Dönüşüm kodlamada görüntü pikselleri bloklara ayrılarak dikgen dönüşüm uygulanır. Dönüşüm alanındaki katsayılar piksellerden daha az bağımlıdır. Dönüşümün performansı ise bağımlılığı ne kadar azalttığı ve enerjiyi ne kadar sıkıştırabildiği ile ölçülür. Birçok dönüşüm yöntemi olmasına rağmen imge sıkıştırmada popüler olan kesikli kosinüs dönüşümüdür. Çünkü kesikli kosinüs dönüşümü bağımlılığı azaltma özelliğine yaklaşır. Bu dönüşüm görüntü ve video sıkıştırma standartları olan JPEG, MPEG de kullanılmaktadır.

Vektör dönüşüm kodlama klasik tekniklerin (örneğin skalar dönüşüm kodlama) vektörselleştirilmiş halidir. Vektörleştirme işlemi ise piksel yerine piksel bloklarının yerleştirilmesidir. Buradan da anlaşılacağı gibi vektör dönüşüm kodlama klasik skalar dönüşüm kodlamanın geliştirilmiş halidir. Diğer bir ifadeyle skalar dönüşüm kodlama vektör dönüşüm kodlamanın özel bir halidir.

Vektör dönüşümünün iki özelliği vardır :

Özellik 1) Vektörler arası bağımlılığı azaltmak (Reduction of inter-vector correlation)

Vektör dönüşümü diğer koşullar sabit kalmak şartıyla, minimum kodlama bozulması elde etmek için vektör dönüşüm alanındaki vektörler

arasındaki bağımlılığı azaltmak için kullanılır. Minimum bağımlılık maksimum enerji sıkıştırmasına sebep olur.

Özellik 2) Vektör içi bağımlılığı korumak (Preservation of intra-vector correlation)

Vektör dönüşümü diğer koşullar sabit kalmak şartıyla, minimum kodlama bozulması elde etmek için vektör dönüşüm alanındaki vektör elemanları arasındaki bağımlılığı koruması da gereklidir.

Bundan dolayı optimum vektör dönüşümünün bu iki özeliğe sahip olması gerekir.(Özkan, K.,2000)

1.4.5.1 Kesikli Kosinüs Dönüşümü :

Sinyalleri zaman bölgesinde gösterildiği zaman her zaman noktasına karşılık gelen voltaj değerleri görünür. Bu sinyale hızlı Fourier Dönüşümü uygulandığı zaman aynı sinyalin frekans örnekleri olarak görülür. Eğer sinyal farklı frekans bileşenleri toplamından oluşuyorsa zaman bölgesindeki gösterimde bu frekans bileşenleri görülmez. Aynı sinyale hızlı Fourier Dönüşümü uygulandığında bu frekans bileşenleri ayrıntılı olarak görülebilmektedir. Hemen hemen bütün dönüşümlerde olduğu gibi hızlı Fourier Dönüşümü de tersinirdir. Kesikli Kosinüs Dönüşümü Hızlı Fourier Dönüşümüne benzer ve aynı sonucu verir. Bununla birlikte orijinal sinyal 2-boyutlu ise Kesikli Kosinüs Dönüşümü uygulandığında 3-boyutlu sinyal elde edilir. Bu durumda sinyal grafiksel bir görüntüdür ve X,Y eksenleri verinin boyutlarını, Z ekseni ise sinyalin genliğini ekranda gri seviye değerleri olarak gösterir. Formül 1.6 kesikli kosinüs dönüşümünün, Formül 1.7'de bu dönüşümün piksel ifadesinin açılımıdır. (Özkan, K.,2000)

$$KKD(i,j)=\frac{1}{\sqrt{2N}}C(i)C(j)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} piksel(x,y)\cos\left[\frac{(2x+1)i\,\pi}{2N}\right]\cos\left[\frac{(2y+1)j\,\pi}{2N}\right] \quad (1.6)$$

$$piksel(x,y)=\frac{1}{\sqrt{2N}}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}C(i)C(j)\,KKD(i,j)\cos\left[\frac{(2x+1)i\,\pi}{2N}\right]\cos\left[\frac{(2y+1)j\,\pi}{2N}\right] \quad (1.7)$$

$$C(x)=\frac{1}{\sqrt{2}} \quad \text{eğer x = 0, değilse 1}$$

1.4.6 JPEG (Joint Photographic Expert Group) :

Resmin her renk bileşeni 8x8 bloklara bölünür ve her bir bloğa kesikli kosinüs dönüşümü uygulanır. Daha sonra dönüşüm katsayıları kuantalanır. Bit sayısı ise frekansa bağımlıdır. Yani daha önemli olan alçak frekanslara daha fazla bit ataması, daha az öneme sahip olan yüksek frekanslara daha az bit ataması yapılır. Kodlanmış veri artık transfer edilmeye veya saklanmaya hazırdır. MPEG ile JPEG arasındaki en önemli fark ise JPEG'de devinim sezimi kestirici (motion detection estimator) olmamasıdır. Bununla birlikte hareketli resimlerde her çerçeveye ayrı olarak JPEG algoritması uygulanarak kodlama yapılabilir. JPEG kod çözücü işlemi ise kodlayıcının tam tersidir.

JPEG hem gri seviyeli hem de renkli görüntülere uygulanabilir. JPEG kesintisiz tonlu görüntülerde en iyi sonucu vermektedir. Eğer renkli görüntülerdeki renk değerleri ani değişiyorsa sıkıştırma sonuçları kötü olmaktadır. Sıkıştırma oranı ve görüntü kalitesi parametrelerin kullanıcı tarafından ayarlanabilmesiyle değişiklik göstermektedir.

JPEG algoritması:

Adım 1. Gri seviyeli görüntülerde dönüşüm işlemi yapılmaz, ancak renkli görüntüler RGB uzayından ışık/renk uzayına (YUV uzayına) dönüştürtürülür. Işık elemanları gri seviyelerini, diğer iki eleman da renk bilgilerini içerir. Bunun amacıda insan gözünün ışık bilgilerinin renk bilgilerine göre daha duyarlı olması dolayısıyla renk bilgilerinde daha fazla sıkıştırma yapılabilmesidir. Eğer renk uzayı değiştirilmek istenmezse algoritmanın geri kalan kısmı her renk elemanına ayrı olarak uygulanır. Bununla birlikte bütün elemanların ışık kalitesinde sıkıştırma yapıldığından sıkıştırma oranı daha az olmaktadır.

Adım 2. Pikselin her elemanına örnek indirgemesi yapılır. Işık elemanları tüm çözünürlükte, renk elemanları ise genellikle dikey yönde 2:1 oranında ve yatay yönde 2:1 veya 1:1 oranında azaltılır. Bu adım sayesinde görüntü bilgileri 2:1 veya 3:1 oranında azalmış olur. Nümerik olarak yüksek kayıp olsa da insan gözünün renk çözünürlüğü zayıf olduğundan izleme kalitesini etkilememektedir. Örnek indirgemesi gri seviyeli görüntülere göre daha fazla oranda sıkıştırılabilmektedir.

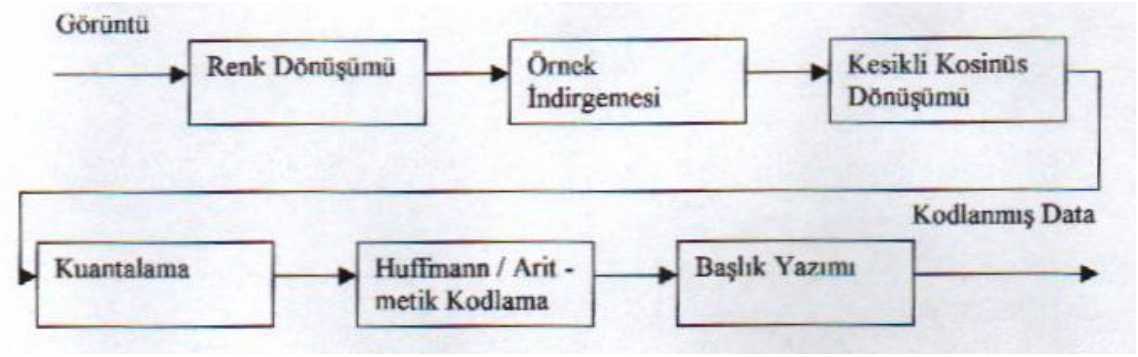
Adım 3. Her görüntü elemanı 8x8 bloklara bölünür. Her bloğa Kesikli Kosinüs Dönüşümü uygulanır. Kesikli Kosinüs Dönüşümünün kendisi yuvarlama hataları dışında tersinirdir.

Adım 4. Her bloktaki, 64 frekans elemanına kuantalama katsayılarına ayrılır ve sonuçlar tam sayıya yuvarlanır. Burası bilginin asıl kaybolduğu adımdır. Kesikli Kosinüs Dönüşümü çıkışı tam sayı değildir. Yüksek frekans bilgileri düşük frekans bilgilerine göre daha düşük duyarlıktadır, ancak göz tarafından daha az hissedilir. Ayrıca ışık bilgileri renk bilgilerine göre daha duyarlı kuantalanır.

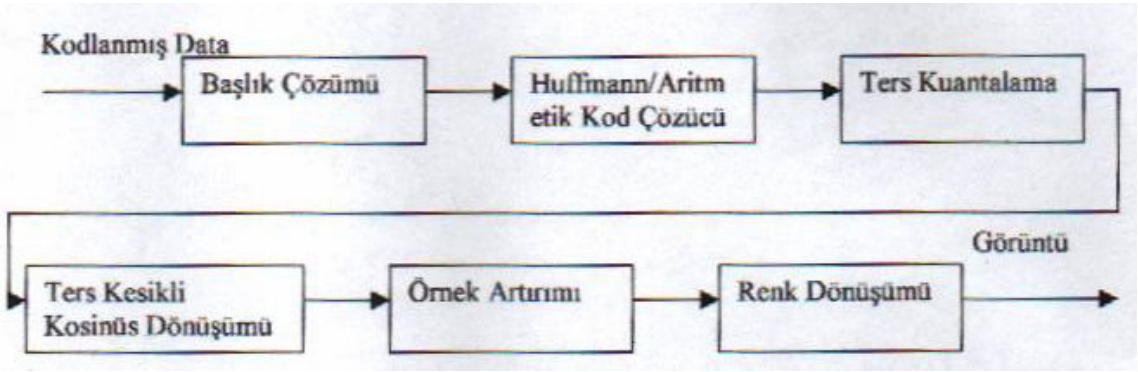
Adım 5. Azaltılmış katsayılar Huffman veya aritmetik kodlama ile kodlanır. Bu adım kayıpsızdır ve görüntü kalitesini etkilemez.

Adım 6. Uygun başlık yazılır. Başlıkta kodu çözebilmek için bütün sıkıştırma parametreleri bulunur. Bu parametreler ayrıca kuantalama tablosunu ve Huffman kodlama tablosunu içerir. Eğer kod çözücüde kuantalama ve Huffman kodlama tablosu var ise kodlayıcıda bı tablolar ayrıca yazılmaz.

Kodlayıcı işlemi Şekil 1.5 de gösterilmiştir. Kod çözücü işlemi ise kodlaycının tam tersidir. Şekil 1.6 gösterilmiştir. Kodlayıcı kuantalama tablosundaki katsayıları çarparak Kesikli Kosinüs Dönüşüm katsayılarına yaklaştırır. Yaklaştırılmış katsayılar ters Kesikli Kosinüs Dönüşümünden geçirilerek tekrar oluşturulmuş görüntü elde edilir. Yüksek kaliteli kod çözme için ekstra adımlar yapılabilir. Bunlar pikseller arasındaki kesikliliği giderir. (http://cobweb.ecn.purdue.edu/~ace/jpeg-tut/jpegtut1.html)



Şekil 1.5 JPEG kodlayıcı



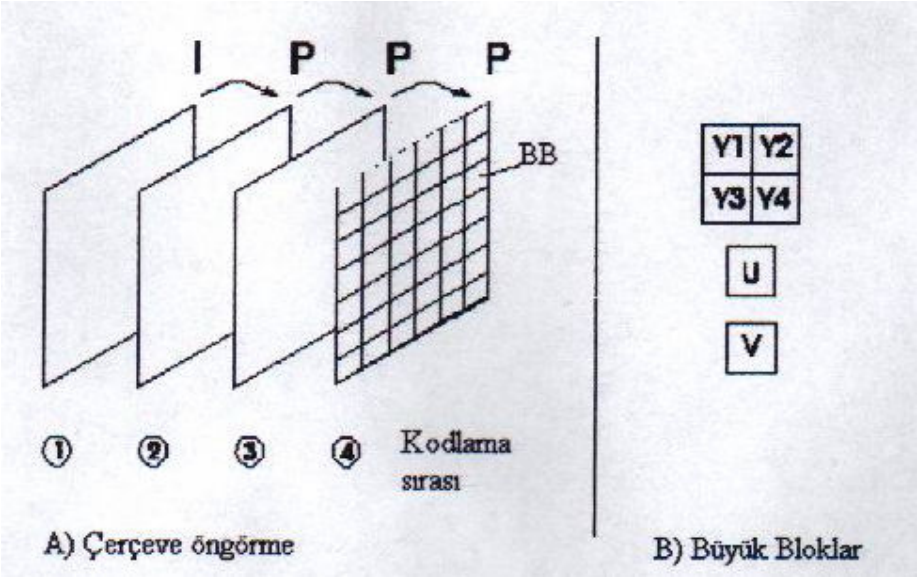
Şekil 1.6 JPEG kod çözücü

1.4.7 MPEG (Moving Picture Expert Group):

MPEG hareketli görüntülerin sıkıştırılmasında kullanılır. JPEG sıkıştırmaya çok benzer. Aralarındaki en önemli fark ise MPEG de çerçeveler arasındaki benzerlikler dikkate alınır. MPEG de resmin her bileşeni 8x8 bloklara bölünür ve her bloğa kesikli kosinüs dönüşümü uygulanır. Katsayılar öngörme yapıldıktan sonra bloklara kuantalama işlemi yapılır.

Video dizilerinde genellikle geçici ve uzamsal boyutta istatistiksel ve öznel tekrarlılık vardır. MPEG aynı çerçeve içindeki pikseli komşu piksellerden (çerçeve içi kodlama tekniği kullanılır) veya komşu çerçevedeki piksellerden (çerçeveler arası kodlama tekniği kullanılır) yararlanarak tahmin eder. Çerçeve içi kodlama tekniğinde görüntü 8x8 bloklara bölünüp, her bloğa kesikli kosinüs dönüşümü uygulanır. Bununla birlikte, eğer çerçeveler arası ilinti yüksekse, bu durumda çerçeveler arası kodlama tekniği olarak farkları öngörücü kodlama kullanılır. (çerçeveler arasında devimin dengeli öngörme yapılır) Hemen hemen bütün video kodlama tekniklerinde olduğu gibi ve JPEG'de de anlatıldığı gibi MPEG'de örnek indirgemesi yapılır. Devimin dengelemeli öngörme çerçeveler arasındaki geçici boyuttaki tekrarlılığı azaltmak için etkili bir yoldur. Devimin dengelemeli öngörmenin temeli çerçeveler arasındaki hareketi tahmin etmektir. Eğer video dizilerindeki pikseller uzamsal boyutta yaklaşık aynı ise, çerçeveler arasındaki hareket az sayıdaki hareket parametreleri ile tanımlanabilir. En iyi öngörme bir pikselin bir önceki çerçevedekine göre hangi yönde değiştiğinin öngörülmesidir. Hareket vektörleri arsında da yüksek ilinti olduğu için piksel yerine piksel bloklarının hareketi öngörülür.

MPEG video sıkıştırmanın temeli blok yapısına dayanır. MPEG kodlayıcıda ilk çerçeve, çerçeve içi kodlama tekniği kullanarak kodlanır (I - resmi). Daha sonra gelen çerçeveler ise çerçeveler arası kodlama tekniği kullanılarak kodlanır (P-resmi). Çerçeveler arası öngörmede (P-resimleri) önceki kodlanmış I veya P çerçevesinden öngörülür. Giriş çerçevesindeki piksellerin her renk elemanı Şekil 1.7 'de görüldüğü gibi gruplanarak büyük bloklar oluşturur. Her blok 8x8 pikselden oluşur.



Şekil 1.7 MPEG bloklama yapısı

MPEG'in blok diyagramı Şekil 1.8'de görülmektedir. İlk çerçeve (I-resmi) bir önceki veya sonraki çerçeveden öngörülmeden kodlanır. Işık ve renk bloklarına Kesikli Kosinüs Dönüşümü uygulanır (KCD). Kesikli Kosinüs değişiminden elde edilen bu katsayı kuantalanır (K). Sıfırdan farklı kuantalama değerleri değişken uzunlukla kodlayıcı tarafından kodlanır.(DUK)

Kod çözücü işlemi ise kodlayıcının tersidir. İlk önce değişken uzunluktaki kod çözücüyle kod kelimeleri çözülür, ters kuantalama yapılır. Daha sonra ters kesikli kosinüs dönüşümden geçirilerek orijinal görüntüye benzetilen görüntü elde edilir. (http://bmrc.berkeley.edu/frame/research/mpeg)

KKD : Kesikli Kosinüs Dönüşümü

K : Kuantalama

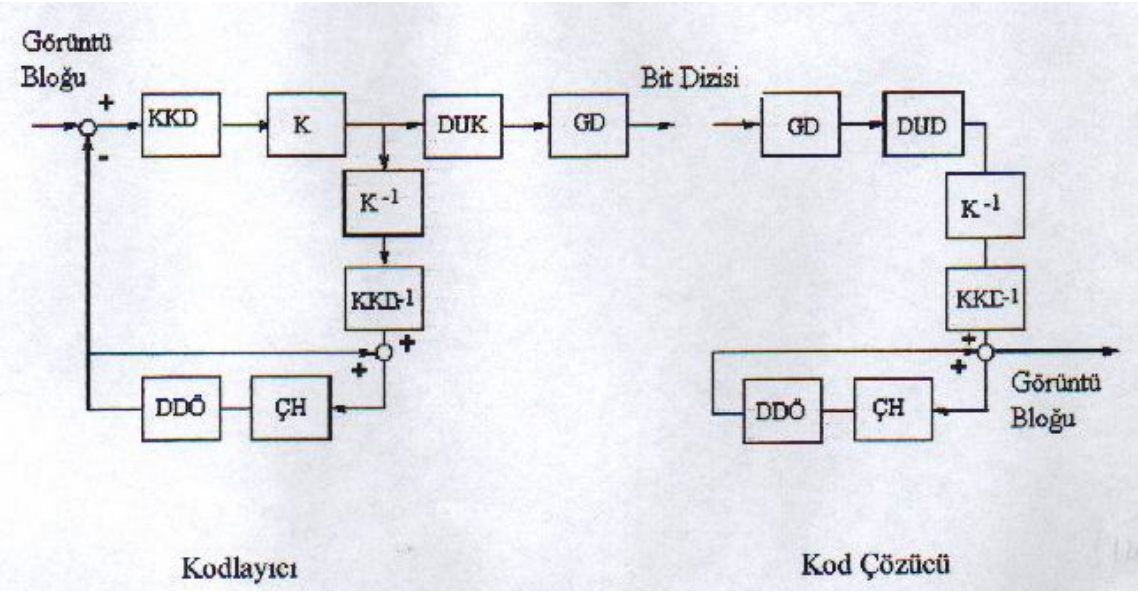
ÇH : Çerçeve Hafızalayıcı

DDÖ : Devinim Dengelemeli Öngörme

DUK : Değişken Uzunluktaki Kodlama

DUD : Değişken Uzunluktaki Kod Çözücü

GD : Görüntü Depolayıcı



Şekil 1.8 MPEG kodlayıcı ve kod çözücü blok yapısı

BÖLÜM 2

SKALER KUANTALAMA

2.1 Skaler Kuantalamanın Tanımı:

Kuantalama, analog sinyalin dijital sinyallere çevrilmesi olarak bilinmektedir. En basit haliyle kuantalama daha önceden belirlenmiş nümerik değerlerden en yakın olan nümerik değerlere atama işlemidir. Çoğunlukla giriş değerleri analogtur, yani kesintisiz genlikte herhangi bir değer olabilir. Bununla birlikte çıkış değerleri ise dijitaldir ve bir set şeklinde, { 1, 2, 3, ..., N }, ifade edilir. Burada N değeri çıkış setinin büyüklüğüdür. Kuantalama veri sıkıştırmanın veya kodlamanın önemli bir kısmıdır. Analog sinyallerin etkin bir kodlamayla iletim kanallarında daha etkili bant genişliğine ulaşması sağlanabilmektedir. Fakat sinyal eninde sonunda kullanıcı için analog sinyale çevrileceğinden, kuantalama esnasında ortaya çıkan bozulma sinyali kodlamada istenmeyen bir durumdur.

Analog kaynağın (sürekli zaman, sürekli genlik) dijital kaynağa (kesikli zaman, kesikli genlik) çevrimi iki kısım içermektedir. Bunlar örnekleme ve kuantalamadır. Örnekleme, sürekli zaman sinyalini, düzenli zaman aralıklarında sinyal değerlerini ölçerek kesikli zaman sinyallerine çevirmektir. Kuantalama sonucunda elde edilen kesikli genlikteki sinyal kuantalama hatası veya gürültü nedeniyle giriş sinyalinden farklıdır.

Sinyal örneklerinin her birinin ayrı olarak kuantalanmasına “ skaler kuantalama veya kodlama “ denir. Sinyal örneklerinin bir bloğu veya sinyal parametrelerin bir bloğunun kuantalanmasına ise “ blok kuantalama veya vektör kuantalama “ denir.

Çok kısaca, N-noktalı skalar (tek boyutlu) kuantalamanın tanımı :

$$\mathbf{K : R \rightarrow C}$$

Burada ;

R reel doğru ve

$C \equiv \{ y_1, y_2, y_3, ..., y_N \} \subset R$ çıkış seti veya kod kitabıdır.

Boyutu ise $| C | = N$ dir. Çıkış değerleri, y_i , ise çıkış seviyesi, çıkış noktaları veya yeni üretilen değerler diye adlandırılır.

Ayırma duyarlığı (resolution) veya kod oranı (code rate) tanımı ise, r, $r = \log_2 N$ dır. Bu eşitlik belirlenmiş kuantalama değeri için ne kadar bit gerektiğini ölçer ve orijinal analog genliğinin doğruluk derecesini belirler. Eğer r değeri tamsayı (integer) ise her y_i tek ikili r-sıralı (unique binary r-tupple) olur. Bu durumda kuantalama değerini belirtmek için sabit oranlı kod (fixed rate code) kullanılır. Alternatif olarak, farklı kuantalama değerleri için değişen uzunlukta basamağa (digit) sahip kelimeler kullanılabilir ki buna da değişken oranlı kod (variable rate code) ismi verilir.

Haberleşme teknolojisinde örnek dizileri belli zaman aralıklarında örneklenir ve sabit bit miktarı ile kuantalanır. İletim oranının (transmission rate) tanımı ise, R, her örnek için ne kadarlık bit iletileceğidir ve $R = \log_2 N$ eşitliği ile bulunur. Kod oranı ile iletim oranı nümerik olarak eşittir. Bundan dolayı her iki orana birden tek kelimeyle oran ismi verilir.

Eğer örnekleme dönemi (sampling period) T ise iletim oranı normalize edilerek bir saniyede iletilebilecek bit miktarı ile ifade edilir ve saniyedeki bit miktarı (bits per

second) diye adlandırılır. (http://ocw.mit.edu/NR/rdanlyers/Electrical-Engineering-and-Computer-Science)

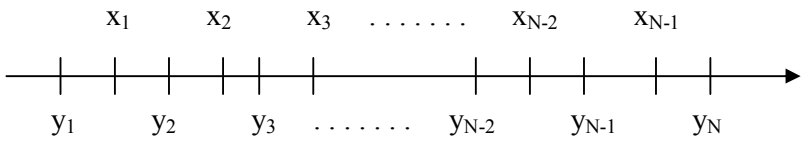
Skalar kuantalama yapabilmek için çok sayıda teknik geliştirilmiştir. Bunlardan en çok kullanılan iki tanesi ise şöyledir. (Gerso, A. and Gray, R.M, 1992)

2.1.1 Tek boyutlu kuantalayıcı (One-dimensional quantizer):

Bu kuantalıyıcıda aralık son noktaları (interval endpoints) aşağıdaki gibi sıralanır. Şekil 2.1'de gösterilmiştir.

$$x_0 < y_1 < x_1 < y_2 < x_2 < \ldots\ldots\ldots < y_N < x_N$$

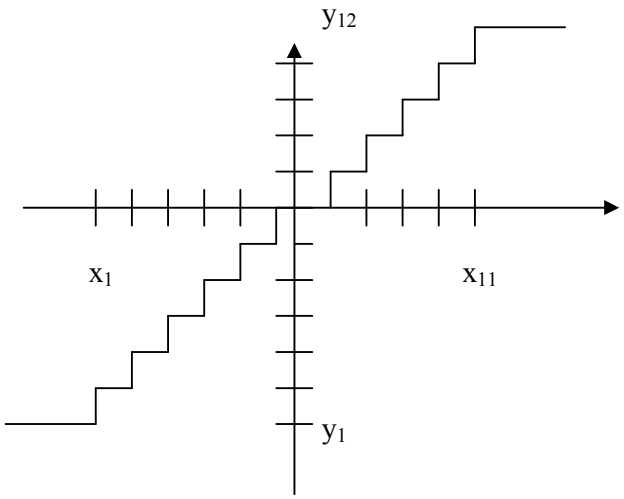
Aralık son noktaları ve çıkış değerleri kuantalayıcı tarafından belirlenir.



Şekil 2.1 Tek Boyutlu kuantalıyıcı. Aralık son noktaları ve çıkış seviyeleri yatay çizgide gösterilir.

2.1.2 Merdiven Kuantalayıcı (Staircase Quantizer):

Merdiven kuantalayıcının grafiği Şekil 2.2' de gösterilmiştir. Giriş çıkış karakteristiği N basamaklı merdiven şeklindedir. Yatay adım aralıkları giriş vektör aralıklarını, dikey adımlar ise çıkış sınır noktalarını gösterir. Bu kuantalayıcıya orta basamaklı kuantalayıcı adı da verilmektedir. (Gerso, A. and Gray, R.M,1992)



Şekil 2.2 Merdiven kuantalayıcının grafiği

2.2 Skalar Kuantalayıcı Dizayn Algoritması:

Skalar kuantalayıcı algoritması ilk olarak Lloyd tarafından geliştirilmiştir. Lloyd algoritması skalar kuantalama için geniş ölçüde kullanılmaktadır. Bu algoritma başlangıç kod kitabı ile başlar ve bitme kriteri sağlanasıya kadar öz yinelemeli olarak devam eder.

Kod kitabı geliştirmek için kullanılan Lloyd öz yinelemesi ise şöyledir:

a) Verilen kod kitabı, $C_m = \{ y_i \}$, için optimum kuantalama hücreleri bulunur. Bu bulma işleminde en yakın komşu şartı aranır. En yakın komşu şartı Formül 2.1' deki gibidir.

$$R_i = \{ x: d(x, y_i) \leq d(x, y_j) ; bütün\ j \neq i \} \quad (2.1)$$

b) Merkezleme şartı kullanılarak optimum yeni üretilmiş kod kitabı bulunur. Formül 2.2' de gösterilmiştir.

$$Merkez\ (R) = \frac{1}{|| R ||} \sum_{i=1}^{||R||} X_i \quad (2.2)$$

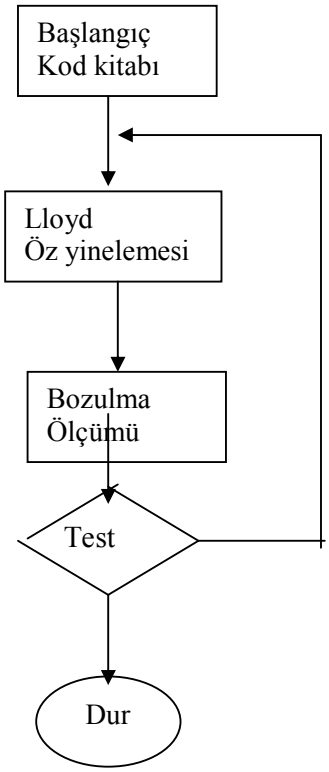
Lloyd Algoritması :

Adım 1) Öz yineleme sayısı, m, bire eşitlenir. Başlangıç kod kitabı ile başlanır.

Adım 2) Verilen kod kitabı, C_m, Lloyd öz yinelemesine tabi tutularak yeni kod kitabı, C_{m+1}, üretilir.

Adım 3) Yeni kod kitabının, C_{m+1} , ortalama bozulması hesaplanır. Eğer bozulma bir önceki bozulmaya göre çok az bir değişim gösteriyorsa algoritma durdurulur. Diğer şartta öz yineleme sayısı artırılır ve ikinci adıma dönülür.

Şekil 2.3’ de Lloyd algoritmasının akış diyagramı gösterilmiştir. (Özkan, K.,2000)



Şekil 2.3 Lloyd algoritmasının akış diyagramı

BÖLÜM 3

VEKTÖR KUANTALAMA

Veri sıkıştırmanın temel amacı kabul edilebilir doğruluk veya görüntü kalitesinde iletim veya veri saklamak için bit sayısını azaltmaktır. Uyarlanır farksal vurumlu kod kişlenimi (differential pulse code modulation), dönüşüm kodlaması (transform coding), hyprid kodlama ve bunların uyarlanır varyosyanları (adaptive variations) gibi bir çok bant genişliği azaltma teknikleri geliştirilmiştir. Bu teknikler bit oranını azaltırken görüntüdeki tekrarlıklardan yararlanılır. Örneğin dönüşüm kodlamada, ilk olarak vektörler bloklara ayrılır ve her blok dönüşüme uğratılır. Dönüşüm örnekleri daha sonra skalar olarak kuantalanır. Öngörücü kodlamada şimdiki örnekle öngörülen örnek arasındaki fark kodlanır.

Shonnon'un oran bozulma kuramına göre teorik olarak daha iyi bir performans elde edebilmek için skalerler yerine vektörler kullanmak gerekir. Eğer sinyal örnekleri veya sinyal parametreleri istatistiksel olarak birbiriyle bağımlı ise örnekler veya parametreler bloklar halinde alınarak bu bağlılık kullanabilmektedir. Böylece vektör kuantalama kullanarak ulaşılan bit miktarı skaler kuantalamayla ulaşılan bit miktarı ile karşılaştırıldığında daha düşük olduğu görülmektedir. Görüntü sinyal örneklerinde bu bağımlılık mevcuttur. Bu imge kodlamada vektör kuantalamayı kullanmak için geçerli bir sebeptir.

Vektör kuantalama, skaler kuantalamanın bir vektör üzerine gerçek sayılar kümesinde kuantalanmasıdır. Skaler kuantalama öncelikle analogtan sayısala çevriminde kullanılırken, vektör kuantalama sayısal sinyal işlemede kullanılır. Burada giriş sinyali sayısal gösterimin bir şekli olarak ve istenilen çıkışta orijinal sinyalin sıkıştırılmış versiyonudur. (Gerso, A. and Gray, R.M,1992)

Bir kuantalayıcıya optimum diyebilmek için çıkıştaki bozulmanın minimum olması gerekmektedir. Bir kuantalayıcıda bozulma yerel olarak (blok içinde) minimum ise yapılan kuantalama genel olarak da minimumdur. Böylece, orijinal görüntü sinyal bloklarının en iyi temsil edileceği bir kod kitabı oluşturalabilinecektir.

3.1 Vektör Kuantalamanın Tanımı :

Birçok araştırmacı vektör kuantalama veya çok boyutlu kuantalama üzerine kapsamlı çalışmalar yapmışlardır. Görgül veriden (emprical data) optimum vektör kuantalama dizaynı Linde, Buzo, Gray'in çalışmalarında sınıflandırma yaklaşımı ile amaçlanmıştır. Bu algoritma LBG algoritması olarak da bilinmektedir.

Vektör kuantalama K Boyutlu Euclidian uzayının, R^k, aynı uzaydaki sınırlı alt sete, Y, haritalanması diye tanımlanır. Yani,

$$\mathbf{K : R^k \rightarrow Y}$$

Burada;

$Y=(y_i ; i=1,2, \ldots,L)$	tekrar üretilmiş vektör seti
L	Y uzayındaki vektör sayısıdır.

Vektör kuantalama, N boyutlu giriş vektörleri, x, ve yine N boyutlu daha az sayıdaki çıkış vektörlerine, y, haritalamaktır. y vektörleri, x vektörünün kuantalanmış değeri olmaktadır. Formül 3.1' de ifade edilmiştir.

$$\mathbf{y = K (x)} \qquad (3.1)$$

K (.), kuantalama operatörüdür.

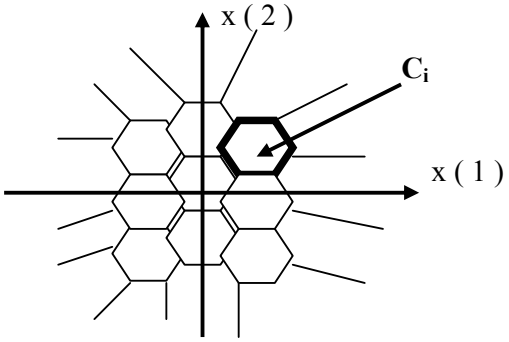
y, tekrar oluşturulan vektör veya x ile ilişkili çıkış vektörü olarak tanımlanabilir. Tipik olarak;

$Y = \{ y_i ; 1 < i < L\}$, ve $y_i = (y_{i1} , y_{i2}, ... , y_{iN})$

yazılabilir. Y seti, tekrar oluşturulan kod kitabı veya basit olarak kod kitabı olarak tanımlanır. L kod kitabının büyüklüğüdür ve kuantalama seviyesinin derecesini belirler. Bu açıklamaların sonucu olarak L-seviyeli kod kitabı ve L-seviyeli kuantalayıcı aynı anlama gelmektedir. Kod kitabı dizaynı için x giriş vektörünün N-boyutlu uzayı, L parçaya veya hücreye ayrılmaktadır. $\{ C_i , 1 < i < L \}$ ile gösterilebilinir ve C_i hücresi y_i vektörü ile temsil edilmektedir. Eğer x giriş vektörü C_i hücresinde ise kod vektörü olarak y_i atanır. Formül 3.2' de açıklanmıştır.

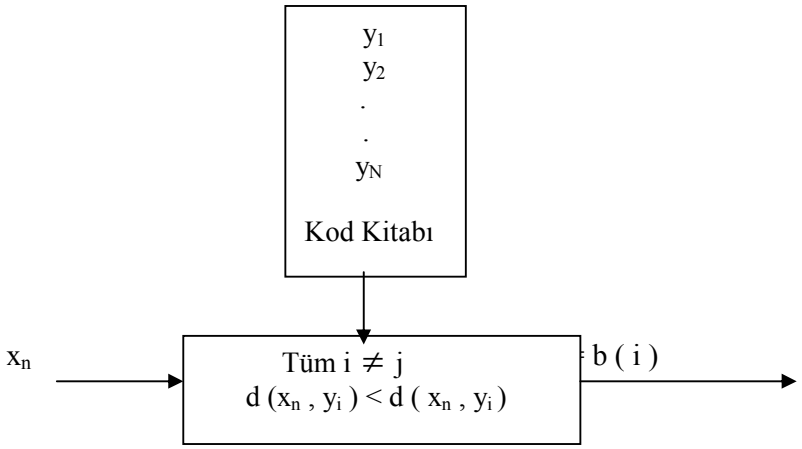
$$\mathbf{x} \in \mathbf{C_i} \qquad \mathbf{K (x) = y_i} \tag{3.2}$$

Şekil 3.1' de vektör kuantalamaya (N = 2) iki boyutlu uzayda örnek verilmiştir. İki boyutlu uzay 22 hücreye (L = 22) bölünmüştür. Her giriş vektörü C_i hücresindeki bir vektör ile kuantalanmaktadır. Hücre şekilleri çok farklı olabilmektedir. Kod vektörlerin uzaydaki pozisyonları diğer hücrelerle bağlantılıdır. (Gerso, A. and Gray, R.M,1992)



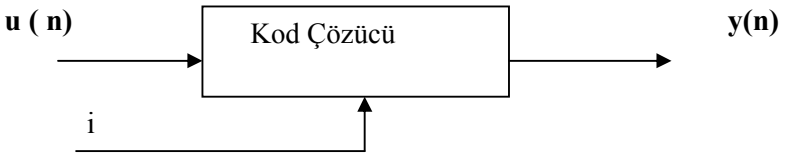
Şekil 3.1 Kuantalama hücre örneği

Kuantalama iki fonksiyonun birleştirilmesi ile olur : kodlayıcı ve kod çözücü. Kodlayıcıda, giriş vektörü ve K (x) ile belirlenen tekrar üretilmiş vektörünü indisinin ikili gösterimi görülür. Giriş vektörü ve her kod vektörü arasındaki mesafe hesaplanarak minimum mesafeye sahip kod vektörünün indisinin ikili gösterimi çıkışta kodlanır. Şekil 3.2 'de gösterilmiştir.



Şekil 3.2 Vektör kuantalama kodlayıcı

Kod çözücüde ise, ikili indisleri kullanarak giriş vektörüne benzeyen vektörler, y, üretilir. Şekil 3.3'de gösterilmiştir.



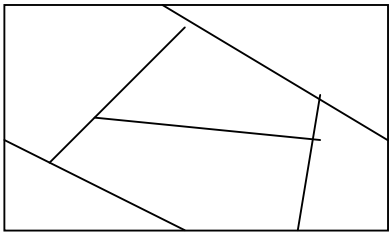
Şekil 3.3 Vektör kuantalama kod çözücü

Orijinal giriş vektörü, x, tekrar üretilmiş vektör, y, arasında ne kadarlık bozulma olduğunun ölçülmesi gerekir ki bu da d(x,y) ile gösterilir. Daha iyi bir haritalama için d(x,y) minimize etmek gerekir. (Gerso, A. and Gray, R.M,1992)

Vektör Kuantalama Örneği:

İki boyutlu kodlayıcı ne polytopal ne de düzgün olup çünkü hücrelerin yüzeyleri (tek boyutlu sınırları) düz çizgi şeklinde ve hücreler convex yapıdan oluşur. Noktalar iki boyutlu uzayda kod vektörleri ve alanın içindeki her kod vektör parçalanmış hücrelerde bulunmaktadır.

Şekil 3.4' de iki boyutlu düzenli kuantalayıcıdan oluşmaktadır. Bu kuantalayıcı polygon hüzreleriyle (iki boyutlu kapalı polytopelarla) sınırlandırılmıştır.



Şekil 3.4 Düzgün Kuantalayıcı

Vektör kuantalamanın genel bir gösterimi olarak, x1 ve x2 rasgale değişkenler olup bunların herbiri skaler kuantalayıcı tarafından kuantalanmıştır.

$x= (x_1,x_2)$

Vektör kuantalayıcısı da,

$Q(x) = (Q_1(x_1), Q_2(x_2))$

Q_1 ve Q_2, x_1 ve x_2'nin skaler kuantalayıcısıdır. (Gerso, A. and Gray, R.M,1992)

3.2 Bozulma Ölçümleri :

Literatürde çok sayıda bozulmayı ölçme yöntemleri vardır. Bozulma ölçümü kolayca uygulanabilir olmak zorundadır. Bugün kullanılan bozulma ölçümleri kolayca uygulanabilir olmasına rağmen biraz uzun sürmektedir. Kodlamada matematiğin kolay olmasından dolayı en çok kullanılan yöntem hataların karesi (square error) yöntemidir. Formül 3.3' de açılımı görülmektedir.

$$d(x,\hat{x}) = \| x - \hat{x} \|^2 = \sum_{i=0}^{k-1} | x - \hat{x} |^2 \tag{3.3}$$

Diğer yaygın olarak kullanılan bozulma ölçümü ise I_v veya Holder normu'dur. Formül 3.4' deki gibidir.

$$d(x,\hat{x}) = \left\{ \sum_{i=0}^{k-1} | x_i - \hat{x}_i |^v \right\}^{1/v} = \| x - \hat{x} \|_v \tag{3.4}$$

Bir diğeri ise v^{th} – law bozulma ölçümüdür. Formül 3.5' de gösterilmiştir.

$$d(x,\hat{x}) = \sum_{i=0}^{k-1} |x_i - \hat{x}_i|^v = \|x - \hat{x}\|_v^v \tag{3.5}$$

Ağırlaştırılmış hataların karelerin toplamı Formül 3.6;

$$d_W(x,\hat{x}) = (x-\hat{x})^T W (x-\hat{x}) \tag{3.6}$$

şeklindedir.

Burada W, pozitif tanımlı (positive-definite) ağırlıklı matristir. $W = N^{-1}$ I alındığında $d_w = d_2$ olmakta, bu da hataların karesi yöntemine eşittir. $W = \Gamma^{-1}$ olarak seçilmektedir. Γ, x vektörünün ortak değişinti (cavariance) matrisidir ve şu şekilde hesaplanmaktadır. Formül 3.7' de ifade edilmiştir.

$$\Gamma = B\left[(x-\hat{x})(x-\hat{x})^T \right] \tag{3.7}$$

Burada;

B = Beklenen değer

Bu durumda d_w Formül 3.8' deki gibidir.

$$d_W(x,\hat{x}) = (x-\hat{x})^T \Gamma^{-1} (x-\hat{x}) \tag{3.8}$$

Bu ölçüm Mahalonobis mesafesi olarak bilinmektedir. (Özkan, K.,2000)

Vektör kuantalamanın en büyük problemi ise giriş vektörlerinin kod kitabındaki en yakın vektöre eşleyebilmek için kod kitabındaki bütün vektörlerle bozulma

ölçümünün hesaplanarak minimum bozulma ölçümüne sahip olan vektörün araştırılmasıdır. Bu araştırma işlemi Buzo tarafından önerilmiş olan ağaç araştırmalı vektör kuantalama (tree search vector quantization) yöntemi ile azaltılmıştır. Ağaç araştırmalı vektör kuantalama da kod kitabının hepsinde araştırmak yerine küçük kod kitabı dizilerinde araştırma diye tanımlanabilir. Bu yöntemin dezavantajı ise tam araştırmalı (full search) vektör kuantalamaya göre daha fazla hafıza istemesidir. Diğer vektör kuantalama yöntemleri ise hafızalama problemlerini azaltmaya çalışmışlardır. Bunlara Fisher tarafından dizayn edilen piramit vektör kuantalama (Pyramid VQ) örnek verilebilir.

PSNR (Doruk Sinyal Gürültü Oranı):

Sinyalin gürültüye oranı, tekrar elde edilen görüntünün kalitesi ile orijinal görüntünün hesaplanmasıdır. Geri elde edilen görüntünün kalitesi oranın yüksek çıkması ile doğru orantılıdır. Doruk sinyal gürültü Oranı ise gerçek bir hesaplama yöntemidir. Öyle ki;

F(i,j) tekrar elde edilen görüntü (reconstructed image)
f(i,j) orijinal görüntü
N x N piksel

$$MSE = \frac{\Sigma [f(i,j) - F(i,j)]^2}{N^2} \qquad (3.9)$$

RMSE (Root Mean Square Error) ise MSE'nin kareköküdür.
PSNR ise desibel (dB) cinsinden şöyle hesaplanır;
(http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html)

$$PSNR = 20 \log_{10} (255 / RMSE) \qquad (3.10)$$

3.3 Vektör Kuantalama İçin Optimum Olma Şartları :

Vektör kuantalayıcının temel amacı optimum kod kitabını bulmaktır. Toplam performansı maksimum yapacak kodlayıcı ve kod çözücünün dizaynının yapılması gerekmektedir. Toplam performansın ölçülmesi ise Formül 3.11' deki gibidir.

$$D = B\{d(x, K(x))\} = \sum_i d(x_i, K(x_i)) p_x(x_i) \qquad (3.11)$$

Burada ;

p_x(x)	olasılık kütle fonksiyonu (probabilty mass function)
d(x, K(x))	x vektörü ile K(x) vektörü arasındaki bozulma miktarı

Büyüklüğü N olan bir kod kitabı, k-boyutlu giriş vektörleri verildiğini ve bozulma ölçümü d(. , .) için herhangi bir yöntemin seçildiğini düşünelim. Bu şartlar altında minimum ortalama bozulmayı sağlayacak bir kuantalayıcı tasarlamak isteyelim. Kodlayıcı, R^k uzayını R_1, R_2, ..., R_N gibi hücrelere böler ve her hücre için o hücreyi temsil edecek olan kod vektörünü belirler. Böyle bir kuantalama tasarımında verilen kod kitabı için optimum bölmeyi ve verilen bölünmüş hücreler için de optimum kod vektörünün bulunma şartlarının belirlenmesi gerekir.

İlk önce sabit kod çözücü için optimum kodlayıcı şartını ele alalım. Verilen kod kitabı için, optimum bölmeyi sağlama koşulu en yakın komşu (nearest neighbor condition) koşuludur. Bütün giriş vektörleri kod kitabındaki kendisine en yakın olan kod vektörü ile temsil edilmelidir. (Gerso, A. and Gray, R.M,1992)

3.3.1 En yakın komşu şartı:

Verilen çıkış setinin optimum bölünme hücreleri Formül 3.12' de gösterilmiştir.

$$R_i \subset \{ x : d(x,y_i) \leq d(x,y_j) \text{ ; bütün j'ler için } \} \quad (3.12)$$

sağlamalıdır. Bu da ;

K (x) = y_i yalnız ve yanlız d (x,y_i) ≤ d (x,y_j) bütün j değerleri için şartını sağlar. Buradan da kodlayıcı da minimum bozulmayı sağlayacak veya en yakın komşuya haritalanır.

Formül 3.13' de ifade edilmiştir.

$$d(x, K(x)) = \min_{y_i \in C} d(x, y_i) \quad (3.13)$$

İkinci durum ise merkezleme şartıdır. Bu da her bölme hücresi için optimum kod vektörünün belirlenmesidir. Merkezlemenin tanımı ise herhangi bir setteki, R ∈ R^k bütün vektörlerin ortalamasıdır. Formül 3.14 ile hesaplanır.

$$Merkez(R) = \frac{1}{\| R \|} \sum_{i=1}^{\|R\|} X_i \quad (3.14)$$

R = { x_i ; i= 1,2,3, ... , || R ||}

|| R || R setindeki eleman sayısıdır. (Gerso, A. and Gray, R.M,1992)

3.3.2 Merkezleme Şartı:

Verilen hücreler için optimum kod vektörü için Formül 3.15 kullanılır.

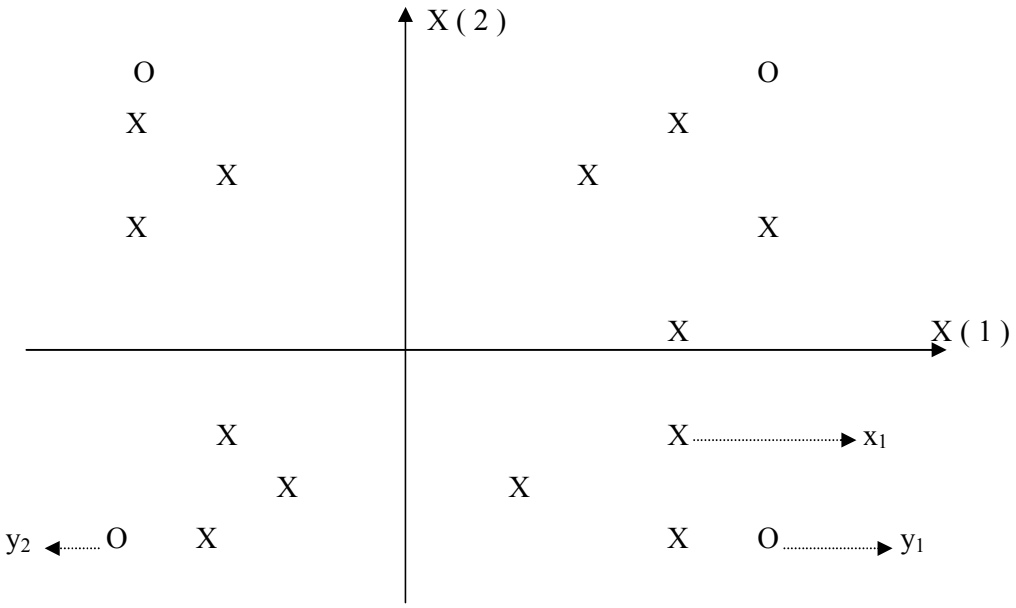
$$y_i = merkez\ (R_i) \tag{3.15}$$

eşitliğinin sağlanması gerekir.

Minimum bozulmayı sağlayan kod vektörünün seçilmesi aşağıdaki örnek ile açıklanabilir. Örnekte 2 – boyutlu 12 eğitme dizisi alınmış ve 4 tane kod vektörü üretilerek 4 bölgeye yerleştirilmiştir. Burada 12 eğitme vektörü kendisine en yakın kod vektör ile temsil edilerek minimum bozulma elde edilmiş vektörler Şekil 3.5' de gösterilmiştir.

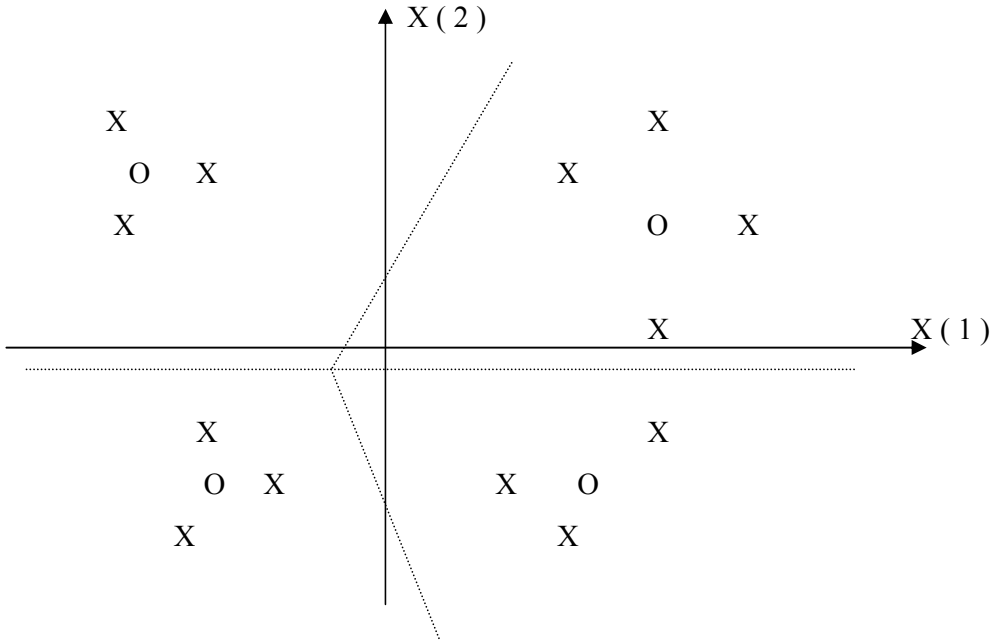
X : eğitme vektörü

O : kod vektörleri



Şekil 3.5 Vektör kuantalama örneği

d (x_1 , y_2) > d (x_1,y_1) olduğundan x_1 vektörü y_1 vektörü ile temsil edilmesi gerekir. Benzer şekilde her eğitme vektörü için bozulma ölçümleri hesaplandıktan sonra aynı kod vektörünü kullanılan eğitme verileri kullandıkları kod vektörünü kendi ağırlık merkezlerine çekerek, Şekil 3.6' da optimum kodlamayı gerçekleştirilmiş olur. (Özkan, K., 2000)



Şekil 3.6 Vektör kuantalama örneği

3.4 Vektör Kuantalama Dizaynı:

Araştırmaların amacı optimum vektör kuantalama elde etmektir. Optimumun anlamı minimum bozulmayı sağlayacak başka bir kuantalayıcı olmamasıdır. Lloyd öz

yinelemeli teknikle skalar kuantalayıcı dizayn etmiştir. Linde Lloyd 'un kullandığı tekniği temel alarak vektör kuantalayacı dizayn etmiştir.

Linde 'nin dizayn ettiği vektör kuantalayıcı algoritması ise;

Adım 1) Başlangıç N seviyeli tekrar üretilmiş alfabe, A_0 , ve eğitme dizisi { x_j ; j = 0, 1, ..., n-1 } verildiğini kabul edelim. Seviye sayısı, N, bozulma eşik değeri, $\varepsilon \geq 0$, belirlenir. Öz yineleme sayısı , m, sıfıra eşitlenir.

Adım 2) Verilen A_m = (y_i ; i = 1, 2, ... , N) için eğitme dizisinde minimum bozulmayı sağlayacak bölümleme P (A_m) = (S_i ; i = 1, 2, ... , N) bulunur. x_j ε S_i eğer d (x_j, y_i) ≤ d (x_j,y_l) bütün değerleri için ortalama bozulma hesaplanır.

$$D_m = D\,[\,(A_m, P(A_m)\,] = (n-1)\sum_{j=0}^{n-1} \min_{y \hat{\in} A_m} d(x_j, y)$$

Adım 3) Eğer ($D_{m-1} - D_m$) / $D_m \leq \varepsilon$ öz yineleme durdurulur ve A_m sonuç tekrar üretilen alfabesi olur. Diğer şarta öz yinelemeye devam edilir.

Adım 4) Optimum tekrar üretilen alfabe $X(P(\hat{A}_m)) = (\hat{X}(S_i); i = 1,...., N)$ bulunur

$$\hat{x}(S_i) = \frac{1}{\| S_i \|} \sum_{j, x_j \in S_i}^{m} x_j$$

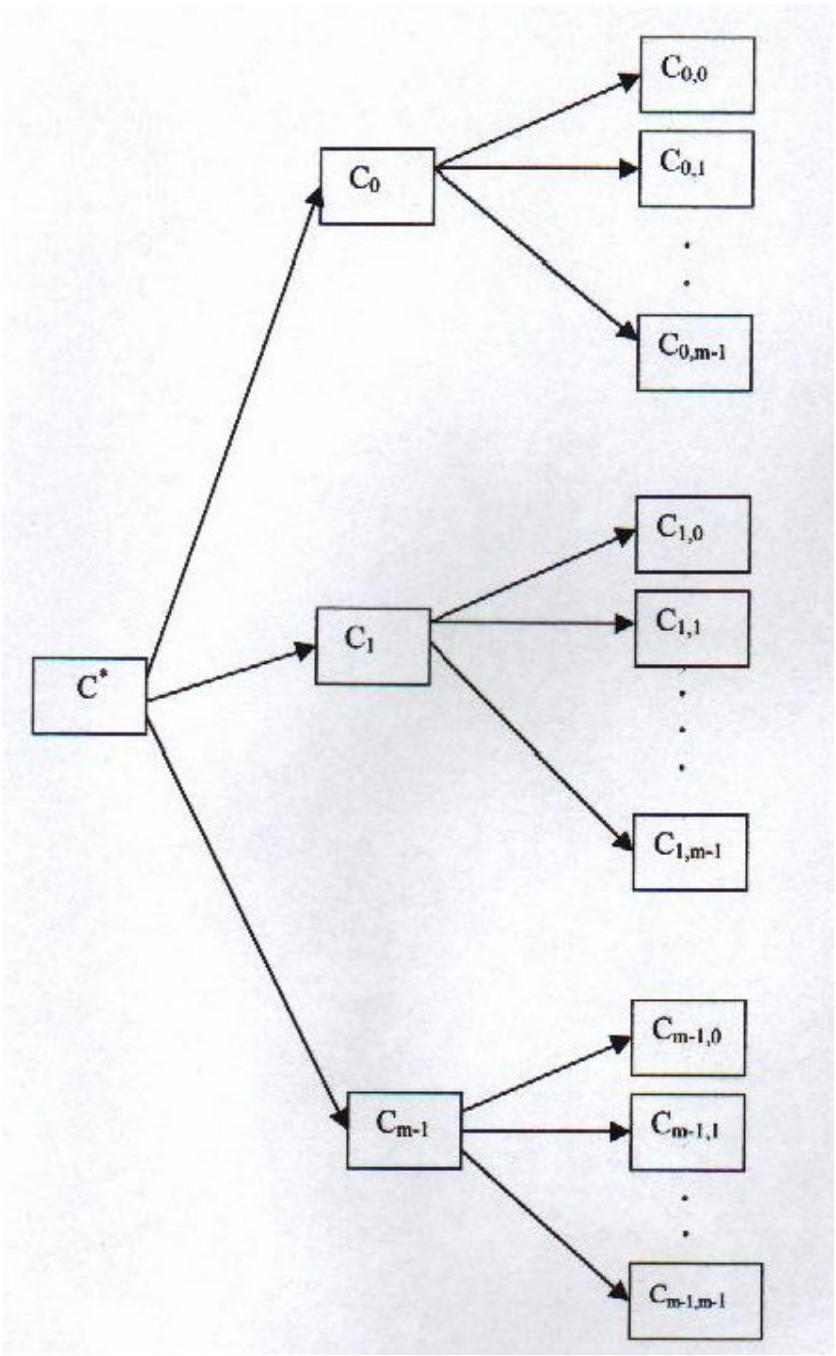
Adım 5) $\hat{x}(S_i)$, $\hat{A}_{m+1}$ set edilir, $\hat{A}_{m+1} = \hat{x}(S_i)$ ve öz yineleme sayısı artırılır , m $\rightarrow$ m+1 ve adım 2' ye dönülür.

Yukarıdaki öz yinelemeli algoritmada başlangıç alfabesi $\hat{A}_0$ (kod kitabı) algoritmanın başladığı yerdir. Başlangıç kod kitabının belirlenmesi için çok sayıda teknik vardır. En basit teknik eğitim dizisinden kod kitabı boyutu kadar vektör almaktır. Linde bölümleme (splitting) tekniğini kullanır. Bölümleme tekniğinde dizinin merkezi (ortalaması) hesaplanır ve iki kod vektöre bölünür. Bu vektörler tekrar ikiye bölünür. Bu bölme işlemi N seviyeli başlangıç kod kitabına kadar devam eder. Bölme işlemi her vektöre ε değeri eklenmesi ve çıkarılması ile bulunur. Yani $y_i + \varepsilon$ ve $y_i - \varepsilon$. ε değeri ise eğitim dizisindeki vektörlerin büyüklüğü ile belirlenir. (Lu, Zheming, Xu Runsheng)

3.5 Vektör Kuantalama Çeşitleri:

3.5.1 Ağaç yapılı vektör kuantalama:

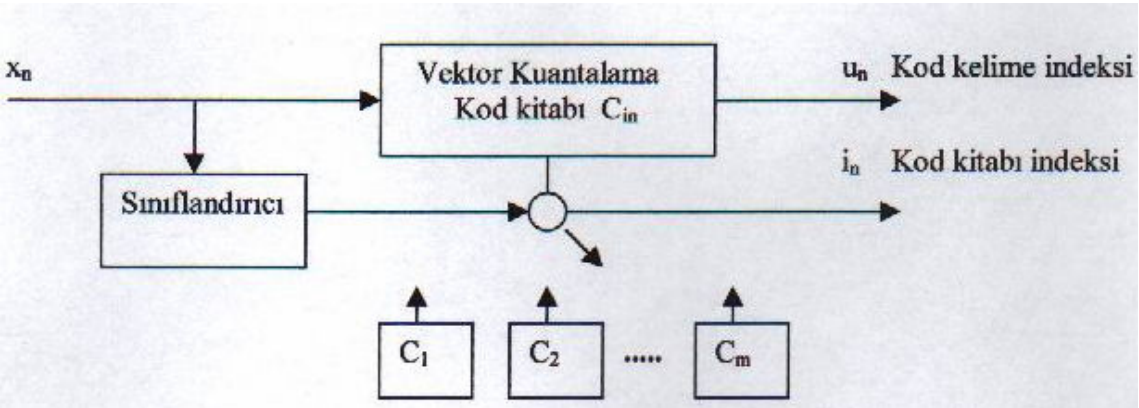
Vektör kuantalamadaki araştırma karmaşıklığını azaltmak için geniş ölçüde kullanılan ve etkili olan bir teknikdir. Ağaç yapılı vektör kuantalama (AYVK) araştırma işini katmanlara bölerek yapar. Her katmanda giriş kod vektörü daha önceden dizayn edilmiş m adet test vektörü (düğüm vektörü) ile karşılaştırarak alt kod vektörlere böler. Giriş kod vektörü en yakın(minimum bozulma) test vektörünün bulunduğu dala koyar. Her katmanda vektörler bir önceki katmana göre 1/m oranında azalır. Ağaç araştırmanın temel yapısı Şekil 3.7 'da görülmektedir. Araştırma işlemi uç düğüm vektörüne kadar devam eder. Kodlayıcıdaki ilk araştırma kod kitabı, C^* , ile başlar ve minumum bozulmaya sahip düğüm kod vektörü bulunur. Eğer düğüm kod vektörünün indeksi i ise, ilk sembol i olur ve araştırma işlemi ağacın i. dalında devam eder. C_i kod vektörü ağacın i. dalındaki düğüm kod vektörleri ile karşılaştırarak minimum bozulmaya sahip düğüm kod vektörünün bulunduğu alt dala gider. Araştırma işlemi böyle devam eder. (Özkan, K., 2000)



Şekil 3.7 Ağaç yapılı vektör kuantalama kodlayıcının temel yapısı

3.5.2 Sınıflandırılmış Vektör Kuantalama:

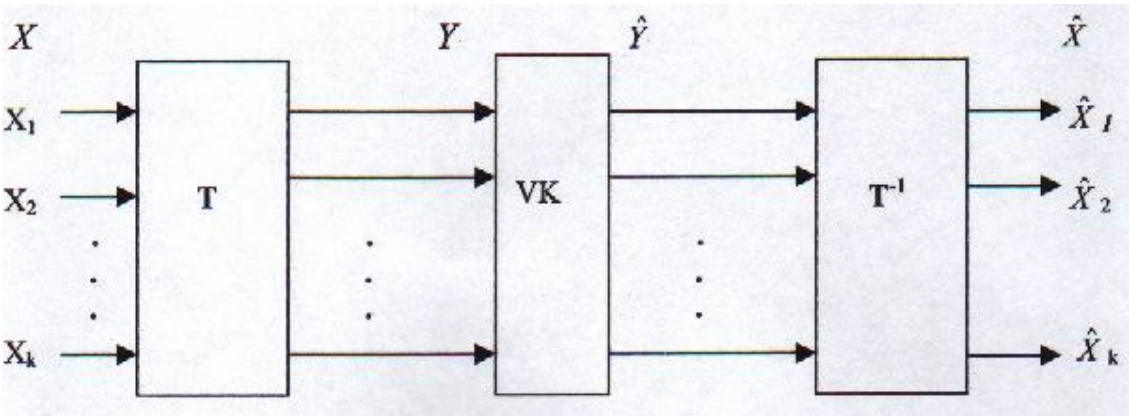
Sınıflandırılmış vektör kuantalama (SVK) tek katlamalı ağaç yapılı vektör kuantalamaya benzemektedir. Sınıflandırılmış vektör kuantalama alt kod kitaplarına bölmek için test vektörleri yerine sınıflandırıcılar kullanmaktadır. Alt küme sayısı ve her kümedeki kod kitabı büyüklükleri değişken olabilmektedir. Şekil 3.8'de sınıflandırılmış vektör kuantalamanın temel yapısı görülmektedir. Sınıflandırıcının seçilmesi için birçok olasılık vardır. En basit olarak giriş uzayı m bölgeye bölünür. Bu durumda sınıflandırılmış vektör kuantalama ikinci katmanındaki kod kitabı büyüklükleri farklı olan iki katmanlı ağaç yapılı vektör kuantalamadır. Sınıflandırıcı giriş vektörlerinin ortalama (örnek değerlerinin ortalaması), enerjisi (örneklerin kareleri toplamı) ve aralık (örneklerin maksimum ve minimum değerleri arasındaki fark) gibi bir veya daha fazla özelliğine göre sınıflandırır. (Özkan, K., 2000)



Şekil 3.8 Sınıflandırılmış vektör kuantalama

3.5.3 Dönüşüm Vektör Kuantalama:

Dönüşüm vektör kuantalama direk giriş vektörünü, ***X***, kuantalamak yerine onu birim dik lineer bir dönüşümden, ***T***, geçirir. Dönüşmüş vektör, ***Y***, kuantalanır. Kuantalama çıkışına ters dönüşüm uygulanarak orijinal giriş vektörüne benzetilen ***X*** vektörü elde edilir. Dönüşüm vektör kuantalama Şekil 3.9'da görülmektedir. Lineer dönüşümün ana fikri orijinal vektör bilgilerini alt vektör kümelerine sıkıştırmaktır. Örneğin, kesikli kosinüs dönüşümü giriş vektörlerini frekans bölgesine haritalar. Genellikle izgen banttın yüksek frekans bölgesinde daha az enerji vardır. Bu özellikten dolayı giriş vektörlerine kesikli kosinüs dönüşümü uygulandığında katsayılar sıfır frekans etrafında yoğunlaşır ve yüksek frekanstaki bilgiler önemsenmeyebilir. Buda kodlanacak vektör boyutlarını azaltmakta ve dolayısıyla vektör kuantalama karmaşıklığı azalmaktadır. (Özkan, K., 2000)



Şekil 3.9 Dönüşüm vektör kuantalama

3.5.4 Bölümlemeli Vektör Kuantalama:

Yüksek boyutlu vektörlerin araştırma ve hafızalama problemini azaltmanın en etkili yöntemi giriş vektörünü alt vektörlere bölünmesidir.

Giriş vektörü,

$X = (X_1, X_2, ..., X_k)$ ise
$X_a = (X_1, X_2, ..., X_m)$ k < m olmak üzere m-boyutlu
$X_b = (X_{m+1}, X_{m+2}, ...X_k)$ k – m boyutlu vektörlere bölünür.

Kodlayıcı X_a giriş vektörü için C_a ve X_b giriş vektörü içinde C_b kod kitabı oluşturur. Optimum kod kitabı dizaynı için eğitme seti de T_a (m-boyutlu) ve T_b (k-m boyutlu) bölünür. Bölümleme işleme 2 veya daha fazla bölümlemede yapılabilir. (Özkan, K., 2000)

3.5.5 Ortalaması Çıkartılmış Vektör Kuantalama:

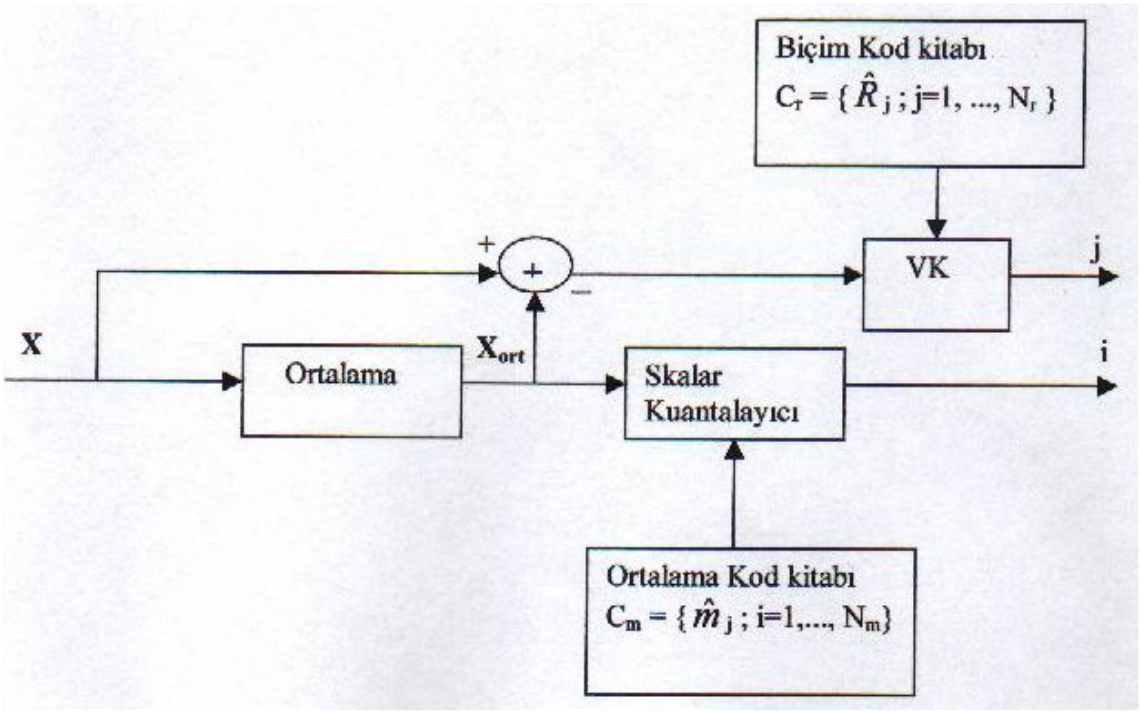
Her elemanın beklenen değeri sıfıra eşit olan vektöre istatistiksel sıfır ortalamaya sahip vektör denir. Bununla birlikte, vektör elemanlarının ortalama değerleri vektörden vektöre çeşitlilik göstermektedir. Ayrıca görüntüden elde edilen vektör elemanlarının değeri pozitif olmakla birlikte ortalaması sıfırdan farklı çıkmaktadır. Her bloğun lokal ortalamasıda görüntü boyunca değişmektedir. Bir vektörün ortalama vektörüde Formül 3.16 ile bulunur.

$$\text{Xort} = \frac{1}{\text{k}} \sum_{i=1}^{k} Xi \qquad (3.16)$$

Ortalaması çıkarılan artık vektörde Formül 3.17 ile bulunmaktadır.

$$\boldsymbol{R=X-X_{ort}} \tag{3.17}$$

Artık vektör sıfır ortalamaya sahiptir. Ortalama çıkarma işlemi görüntü vektörlerini iki özelliğini ayırma işlemidir. Ortalama görüntüdeki arka plan seviyesini, artık vektörde görüntünün biçimini göstermektedir. İki özelliğin ayrı olarak kuantalanmasına da ortalaması çıkartılmış vektör kuantalama denir ve Şekil 3.10'da görülmektedir. Ortalama skalar, artık vektörde vektörel kuantalanır. (Özkan, K., 2000)



Şekil 3.10 Ortalaması çıkartılmış vektör kuantalama

3.5.6 Öngörücülü Vektör Kuantalama:

Hafızasız vektör kuantalamada her giriş vektörü önceki (veya sonraki) vektörle bağıntısına bakılmaksızın kuantalanır. Bu vektör sinyalin sınırlı bir bölümünde veya birine yakın örnek kümelerinden oluşur. Her vektör aynı olasılık yoğunluk fonksiyonuna sahip olduğu kabul edilir, ancak gerçek görüntülerdeki vektörler birbirleriyle istatistiksel olarak bağımlıdır. Vektörler veya bloklar arası bağımlılığın kullanılması ile aynı bit oranında daha iyi performans elde edilmekle birlikte araştırma karmaşıklığını azaltır. Öngörücülü vektör kuantalamada skalar öngörücülü kuantalamadaki skalar kuantalama ve skalar öngörücü yerine vektör kuantalama ve vektör öngörücü kullanılır. Skalar kuantalamada olduğu gibi giriş vektörü ile öngörülen vektör arasındaki fark kuantalanır. (Gerso, A. and Gray, R.M,1992)

3.5.7 Adres Vektör Kuantalama:

Hafızasız vektör kuantalama komşu pikseller arasındaki ilintiyi azaltır, ancak komşu bloklar arası ilintiyi dikkate almaz. Adres vektör kuantalamanın temeli komşu blokları gruplayıp büyük grup blokları oluşturmaktır. Pratik uygulamalar için 4 blok gruplanır. Adres kod kitabı ve kod vektörlerin adresi (indeksi), kod kitabının ilk kısmında ufak bloklara LBG algoritmasıyla üretilmiş olan kod kitapları bulunmaktadır. Kod kitabının ikinci kısmında ise bu kod kitabındaki kod vektörlerinin adresleri bulunur. İkinci kısmında eğitme sırasında bulunan bütün olası adres gruplamaları bulunur. Komşu bloklar birbirleri ile yüksek ilintili olduğu için eğitme sırasında olası blok kombinasyonları çok fazladır, ancak sınırlıdır. Kodlayıcıda her ufak bloğa bit atamak yerine 4'ü bir araya getirilmiş büyük bloklara bit ataması yapılarak gerekli olan bir sayısını azaltır.

Adres kod kitabını bulmak için eğitme setindeki görüntüler küçük bloklara bölünür. Her blok LBG algoritması ile kuantalanır, ve 4 komşu bloğun bütün olası adres kombinasyonları bulunur. Adres kod kitabı genellikle çok büyüktür, bundan dolayı iki bölgeye ayrılır; aktif ve aktif olmayan bölge. Eğitme sırasında yan yana gelme olasılığı fazla olan blokların adresi aktif bölgede, yan yana gelme olsaılığı az olan blokların adresi aktif olmayan bölgede tutulur. Yalnızca aktif bölge kodlayıcı ve kod çözücüde bulunur. Eğer blok kombinasyonu aktif bölgede ise yalnızca indeksi, değilse o bloktaki kod vektörlerde indeksle beraber gönderilir. (Gerso, A. and Gray, R.M,1992)

BÖLÜM 4

BİT-DÜZLEM DİLİMİ

4.1 Bit-düzlem Kodlama (Bit-plane Coding):

Gereksiz kodlamayı atmak için uygulanan başlıca metodlardan, pekçok kayıpsız sıkıştırma tekniklerinden bir tanesi görüntünün karşılıklı ilgisiz piksel bilgilerini ayırmaktır. Bu tekniğe bit-düzlem kodlama denir. Bu teknik bir tek renkli veya renkli görüntüyü binary görüntüler serisine ayırır ve her binary görüntüyü sıkıştırır. Aşağıda en popüler ayrıştırılmış yaklaşımı inceleyeceğiz. (Wang, Z. and Bovik A.C, 2002)

4.2 Bit-düzlem Analizi (Bit-plane decomposition):

M bit gri kademeli (gray-scale) görüntünün gri seviyeleri 2 tabanlı polinom şeklinde ifade edilir. Formül 4.1' de gösterilmiştir.

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + ...+ a_1 2^1 + a_0 2^0 \qquad (4.1)$$

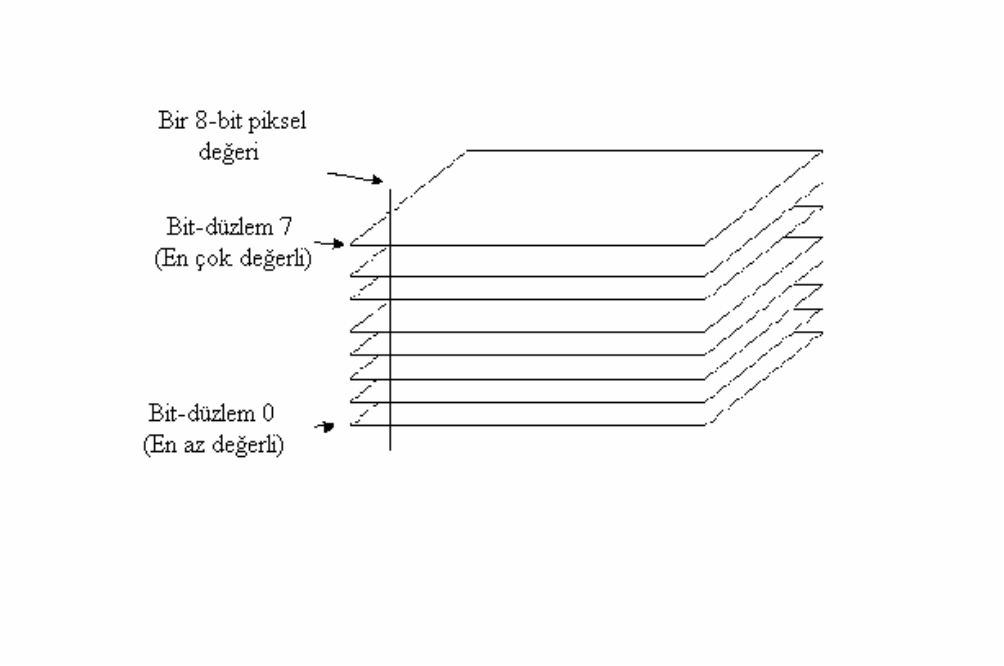
Bu özelliğe bağlı olarak, bir görüntüyü binary görüntülere ve m katsayılı polinomu m tane 1-bit bit düzlemlerine ayırır. (Gonzalez,R.C and Woods,R.E.,2001)

4.3 Bit-düzlem Dilimlemesi (Bit-plane Slicing):

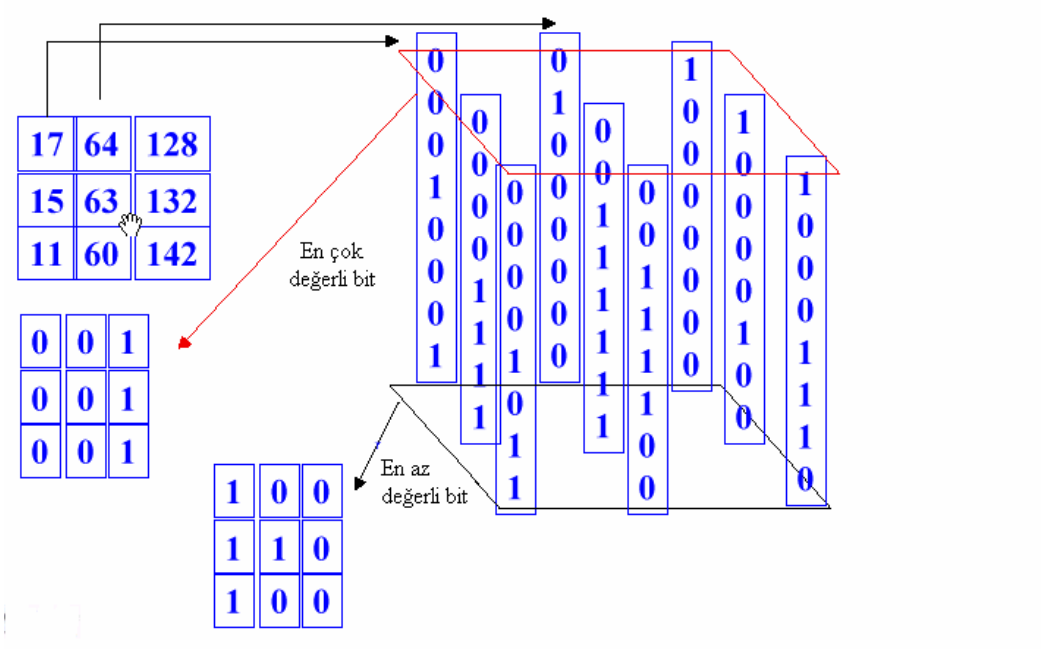
Her pikseldeki görüntüsü 8-bit verilmiş olup, bu görüntüyü farklı düzlemlere (bit-düzlemlerine) ayırmak görüntü işlemede önemli rol oynamaktadır. Uygulanan bu teknik veri sıkıştırmaktadır.

Toplam görüntünün dikkate alınacak tarafı, istenen belirli bitlerle ifade edilmiş olmasıdır. Bir görüntünün her pikseli 8 bit ile gösterildiğini farz edelim. Görüntü 8 tane 1-bit düzlemlerinden oluşur ve düzlem 0'ın en az değerli bit ve düzlem 7 ise en çok değerli bit olarak sıralanır. Düzlem 0 düşük sıralı bitleri düzlem 7 ise yüksek sıralı bitleri içerir. (Gonzalez,R.C and Woods,R.E.,2001) Şekil 4.1' de gösterilmiştir.

Gerçekte, sadece en yüksek 5 bit en önemli verileri içerir. Geri kalan bitler ince detayları içerir. Düzlem 7 gerçek görüntünün gri seviyesi 128'deki değerinin aynısıdır. Şekil 4.2 Bit Düzlem Dilimine başka bir örnektir.



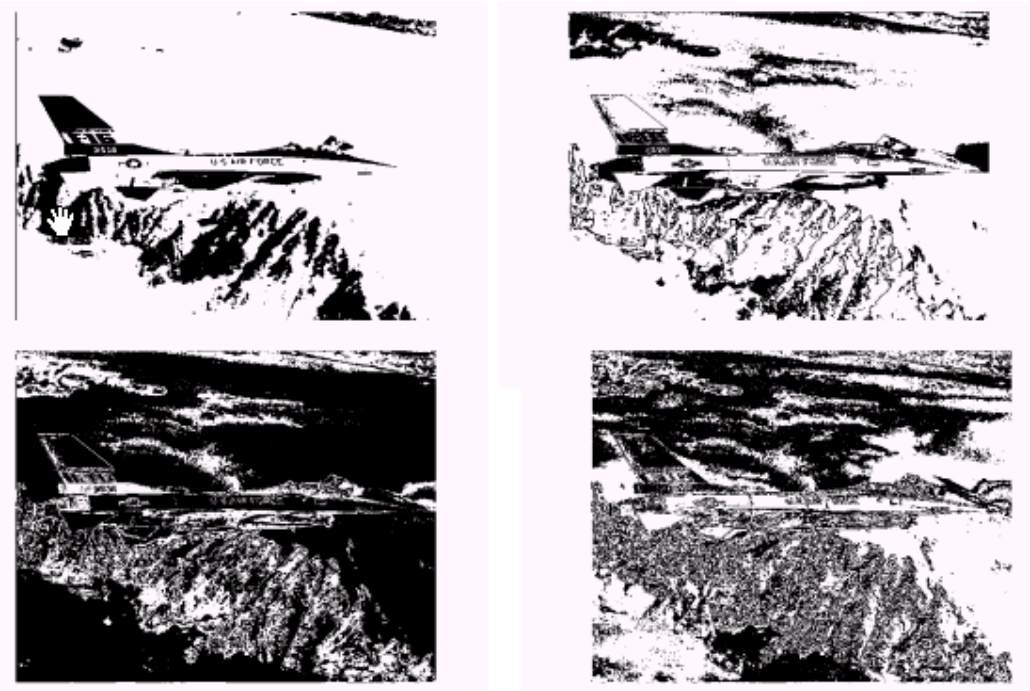
Şekil 4.1 Bit-düzlem Dilimi



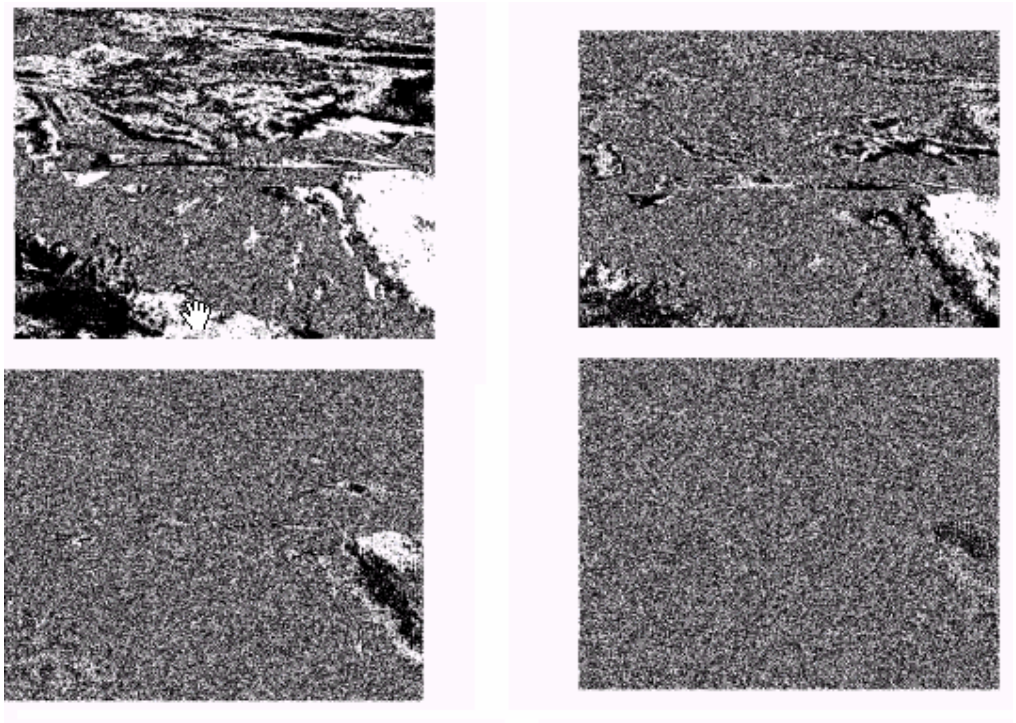
Şekil 4.2 Bit düzlem Dilimi

Şekil 4.3 orijinal görüntü olup, Şekil 4.4 görüntünün yüksek değerli (7., 6., 5., 4.) bitleri, Şekil 4.5' de görüntünün (3., 2., 1., 0.) bitlerini içermektedir.

Şekil 4.3 Orijinal Görüntü



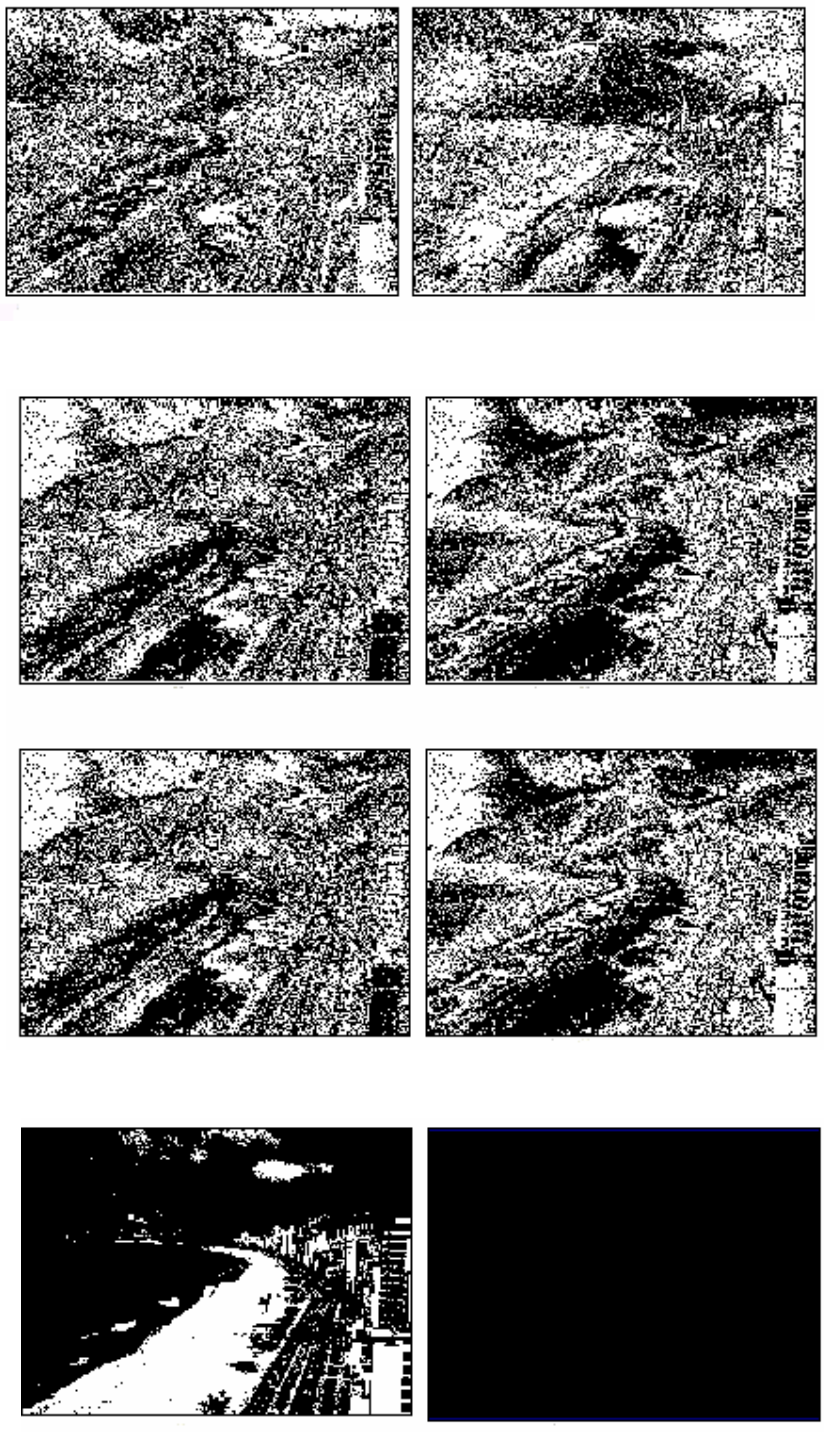
Şekil 4.4 a) Sol-üstteki görüntü bit-düzlem 7
b) Sağ-üstteki görüntü bit-düzlem 6
c) Sol-alttaki görüntü bit-düzlem 5
d) Sağ-alttaki görüntü bit-düzlem 4



Şekil 4.5 a) Sol-üstteki görüntü bit-düzlem 3
b) Sağ-üstteki görüntü bit-düzlem 2
c) Sol-alttaki görüntü bit-düzlem 1
d) Sağ-alttaki görüntü bit-düzlem 0

Şekil 4.6 orijinal görüntü olup, Şekil 4.7' de bit düzlemindeki görüntüleri içermektedir.

Şekil 4.6 Orijinal Görüntü



Şekil 4.7 Bit-düzlem dilimleme yapıldıktan sonraki görüntüler. Bit-düzlem 7'den Bit-düzlem 0'a sıralanmıştır.

BÖLÜM 5

UYGULAMALAR

5.1 Bit-düzlem Dilimi kullanılmış uygulamalar:

1.Program:

Verilen değerlerin (115 86 34 209; 48 75 100 231; 185 18 45 11; 238 15 150 191) 2 bit-düzlem diliminde işlenerek sonra 1. bit düzlemi görüntünün düşük değerli bitlerini (LSB) 2.bit düzleminde görüntünün yüksek değerli bitlerini (MSB) oluşturur. 8 bitlik görüntülerde son dört bit görüntünün düşük değerli bitlerini ve ilk dört bit yüksek değerli bitleri oluşturmaktadır.

Sonuçlar:

Resim =

115	86	34	209
48	75	100	231
185	18	45	11
238	15	150	191

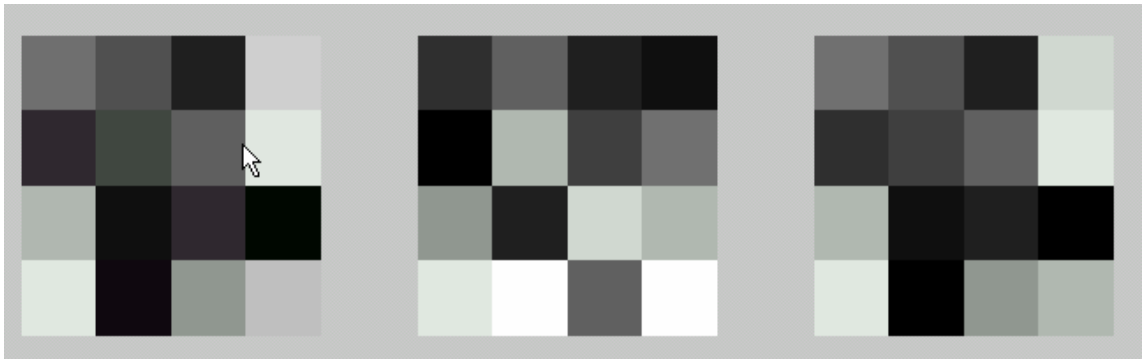
Low_Resim =

3	6	2	1
0	11	4	7
9	2	13	11
14	15	6	15

High_Resim =

7	5	2	13
3	4	6	14
11	1	2	0
14	0	9	11

Şekil 5.1.a orijinal görüntü, Şekil 5.1.b Low_Resim (Düşük değerli bitleri), Şekil 5.1.c ise High_Resim (Yüksek değerli bitlerin) görüntülerini içermektedir.



Şekil 5.1.a Orijinal görüntü

Şekil 5.1.b Düşük değerli bitlerin görüntüsü (LSB) Bit-düzlem 0.

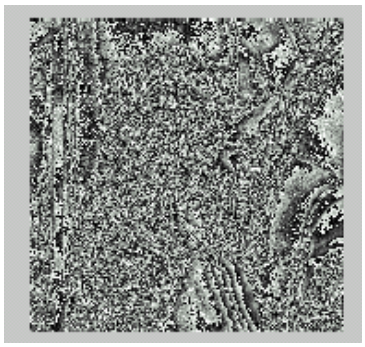
Şekil 5.1.c Yüksek değerli bitlerin görüntüsü (MSB) Bit-düzlem 1.

2. Program

Verilen resimler (Lena ve Kameraman) 2 bit-düzlem diliminde işlenerek sonra 1. bit düzlemi görüntünün düşük değerli bitlerini (LSB) 2.bit düzlemi de görüntünün yüksek değerli bitlerini (MSB) oluşturur. 8 bitlik görüntülerde son dört bit görüntünün düşük değerli bitlerini ve ilk dört bit yüksek değerli bitleri oluşturmaktadır.

Şekil 5.2.a ve Şekil 5.3.a orijinal görüntüleri, Şekil 5.2.b ve Şekil 5.3.b Düşük değerli bitlerin görüntülerini, Şekil 5.2.c ve Şekil 5.3.c ise Yüksek değerli bitlerin görüntülerini içerir. (Kesme Metodu)

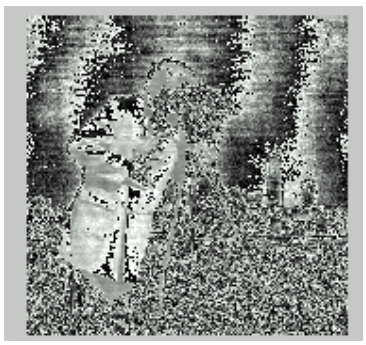
Şekil 5.2.a Orijinal görüntü



Şekil 5.2.b Düşük değerli bitlerin görüntüsü (LSB) Bit-düzlem 0.

Şekil 5.2.c Yüksek değerli bitlerin görüntüsü (MSB) Bit-düzlem 1.

Şekil 5.3.a Orijinal görüntü



Şekil 5.3.b Düşük değerli bitlerin görüntüsü (LSB) Bit-düzlem 0.

Şekil 5.3.c Yüksek değerli bitlerin görüntüsü (MSB) Bit-düzlem 1.

3.Program:

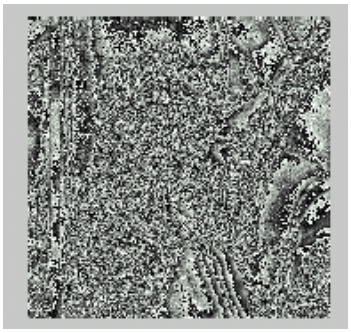
Verilen resimler (Lena ve Kameraman):

a) 2 bit-düzlem diliminde işlenerek sonra 1. bit düzlemi görüntünün düşük değerli bitlerini (LSB) 2.bit düzlemi de görüntünün yüksek değerli bitlerini (MSB) oluşturur. 8 bitlik görüntülerde son dört bit görüntünün düşük değerli bitlerini ve ilk dört bit yüksek değerli bitleri oluşturmaktadır.

b) Programın ikinci kısmında ise yine 2-bit düzleminden oluşmaktadır. 1. bit düzlemi görüntünün düşük değerli bitlerini (LSB) 2.bit düzlemi de görüntünün yüksek değerli bitlerini (MSB) oluşturur. 8 bitlik görüntülerde 1.bitten başlayıp birer atlayarak 3. ,5., 7. bitler görüntünün düşük değerli bitlerini ve diğerleri (2. , 4. ,6., ve 8., bitler)'de yüksek değerli bitleri oluşturmaktadır.

Şekil 5.4.a, Şekil 5.4.d, Şekil 5.5.a ve Şekil 5.5.d orijinal görüntü olup, Şekil 5.4.b ve Şekil 5.5.b Kesme Metodu ile görüntünün düşük değerli bitlerini, Şekil 5.4.e ve Şekil 5.5.e ise Taraklama Metodu ile görüntünün düşük değerli bitlerini içerir. Şekil 5.4.c ve Şekil 5.5.c Kesme Metodu ile görüntünün yüksek değerli bitlerini, Şekil 5.4.f ile Şekil 5.5.f Taraklama Metodu ile görüntünün yüksek değerli bitlerini içerir.

Şekil 5.4.a Orijinal görüntü



Şekil 5.4.b Düşük değerli bitlerin görüntüsü (LSB) Bit-düzlem 0.

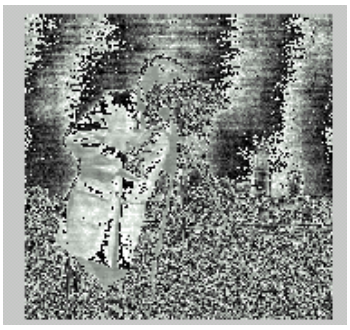
Şekil 5.4.c Yüksek değerli bitlerin görüntüsü (MSB) Bit-düzlem 1.

Şekil 5.4.d Orijinal görüntü

Şekil 5.4.e Düşük değerli bitlerin görüntüsü (LSB) Bit-düzlem 0 Görüntünün 0.,2.,4.,6 bitlerini kullanarak

Şekil 5.4.f Yüksek değerli bitlerin görüntüsü (MSB) Bit-düzlem 1. Görüntünün 1.,3.,5.,7 bitlerini kullanarak

Şekil 5.5.a Orijinal görüntü



Şekil 5.5.b Düşük değerli bitlerin görüntüsü (LSB) Bit-düzlem 0.

Şekil 5.5.c Yüksek değerli bitlerin görüntüsü (MSB) Bit-düzlem 1.

Şekil 5.5.d Orijinal görüntü

Şekil 5.5.e Düşük değerli bitlerin görüntüsü (LSB) Bit-düzlem 0 Görüntünün 0.,2.,4.,6 bitlerini kullanarak

Şekil 5.5.f Yüksek değerli bitlerin görüntüsü (MSB) Bit-düzlem 1. Görüntünün 1.,3.,5.,7 bitlerini kullanarak

5.2 Vektör Kuantalama Programı Kullanılmış Uygulamalar:

5.2.1 Direk Vektör Kuantalama Uygulaması:

Kod Kitabı büyüklüğü*; 64, 128, 256, 512, 1024* (4 x 4 bloklar) seçilip 8 bit 512 x 512 "lena" , 8 bit 256 x 256 "cameraman", 8 bit 512 x 512 "boats" üzerinde uygulanmıştır. Bütün uygulamalarda resimler üzerinde bit / piksel = 1 alınmıştır.

Şekil 5.6.a, Şekil 5.7.a ve Şekil 5.8.a orijinal görüntüler olup, Şekil 5.6.b, Şekil 5.7.b ve Şekil 5.8.b 64 kod kitabı kullanarak, Şekil 5.6.c, Şekil 5.7.c ve Şekil 5.8.c 128 kod kitabı kullanarak, Şekil 5.6.d, Şekil 5.7.d ve Şekil 5.8.d 256 kod kitabı kullanarak, Şekil 5.6.e, Şekil 5.7.e ve Şekil 5.8.e 512 kod kitabı kullanarak, Şekil 5.6.f, Şekil 5.7.f ve Şekil 5.8.f 1024 kod kitabı kullanarak direk VQ uygulanan görüntüleri, sıkıştırma oranlarını ve hesaplanan PSNR değerlerini içerir.

Şekil 5.6.a. Orijinal Görüntü

Şekil 5.6.b. Kod Kitabı: 64
1:256 (8 bit) Sıkıştırma Oranı
PSNR: 28.172

Şekil 5.6.c. Kod Kitabı: 128
1:128 (8 bit) Sıkıştırma Oranı
PSNR: 28.819

Şekil 5.6.d. Kod Kitabı: 256
1:64 (8 bit) Sıkıştırma Oranı
PSNR: 29.388

Şekil 5.6.e. Kod Kitabı: 512
1:32 (8 bit) Sıkıştırma Oranı
PSNR: 29.933

Şekil 5.6.f. Kod Kitabı: 1024
1:16 (8 bit) Sıkıştırma Oranı
PSNR: 30.617

Şekil 5.7.a. Orijinal Görüntü

Şekil 5.7.b. Kod Kitabı: 64
1:64 (8 bit) Sıkıştırma Oranı
PSNR: 21.565

Şekil 5.7.c. Kod Kitabı: 128
1:32 (8 bit) Sıkıştırma Oranı
PSNR: 21.755

Şekil 5.7.d. Kod Kitabı: 256
1:16 (8 bit) Sıkıştırma Oranı
PSNR: 21.813

Şekil 5.7.e. Kod Kitabı: 512
1: 8 (8 bit) Sıkıştırma Oranı
PSNR: 22.958

Şekil 5.7.f. Kod Kitabı: 1024
1:4 (8 bit) Sıkıştırma Oranı
PSNR: 24.965

Şekil 5.8.a. Orijinal Görüntü

Şekil 5.8.b. Kod Kitabı: 64
1:256 (8 bit) Sıkıştırma Oranı
PSNR: 27.809

Şekil 5.8.c. Kod Kitabı: 128
1:128 (8 bit) Sıkıştırma Oranı
PSNR: 28.671

Şekil 5.8.d. Kod Kitabı: 256
1:64 (8 bit) Sıkıştırma Oranı
PSNR: 29.378

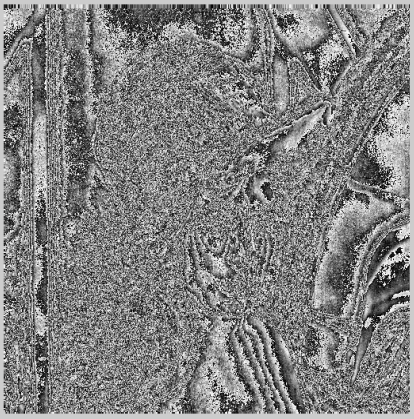
Şekil 5.8.e. Kod Kitabı: 512
1:32 (8 bit) Sıkıştırma Oranı
PSNR: 30.226

Şekil 5.8.f. Kod Kitabı: 1024
1:16 (8 bit) Sıkıştırma Oranı
PSNR: 30.994

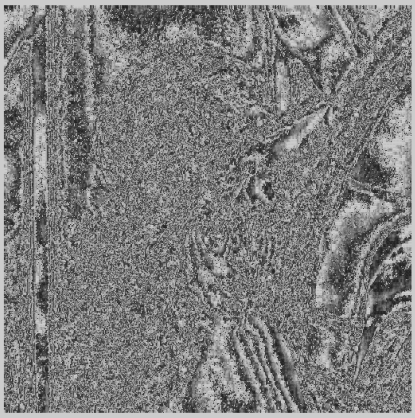
5.2.2 Bit Düzlem Diliminde Vektör Kuantalama Uygulamaları:

5.2.2.1 8 bitlik görüntülerde son dört bit görüntünün düşük değerli bitlerini ve ilk dört bit yüksek değerli bitlerini ayırıp, 512 kod kitabı, (4 x 4) bloklar halinde, ayrı ayrı uygulanan vektör kuantalama sonucunda elde edilen görüntüler;

Şekil 5.9.a ve Şekil 5.10.a 4 bit düşük değerli bitler ve Şekil 5.9.b ve Şekil 5.10.b 512 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.9.c ve Şekil 5.10.c 4 bit yüksek değerli bitleri ve Şekil 5.9.d ve Şekil 5.10..d 512 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.9.e orijinal görüntü ile Şekil 5.9.f, Şekil 5.9.b ve Şekil 5.9.d' nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır. Şekil 5.10.e orijinal görüntü ile Şekil 5.10.f, Şekil 5.10.b ve Şekil 5.10.d'nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır.(Kesme Metodu)



Şekil 5.9.a. 4 bit Lena LSB



Şekil 5.9.b. Kod Kitabı: 512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 37.986

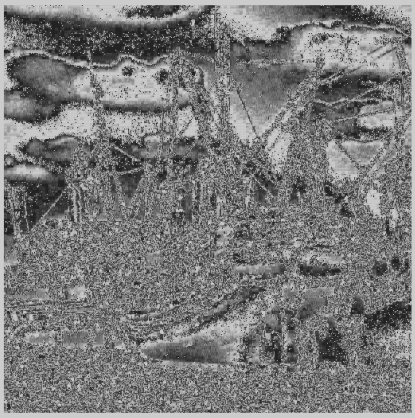
Şekil 5.9.c. 4 bit Lena MSB

Şekil 5.9.d. Kod Kitabı: 512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 52.276

Şekil 5.9.e. Orijinal Lena

Şekil 5.9.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 28.487

Şekil 5.10.a. 4 bit Boats LSB



Şekil 5.10.b. Kod Kitabı: 512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 38.433

Şekil 5.10.c. 4 bit Boats MSB

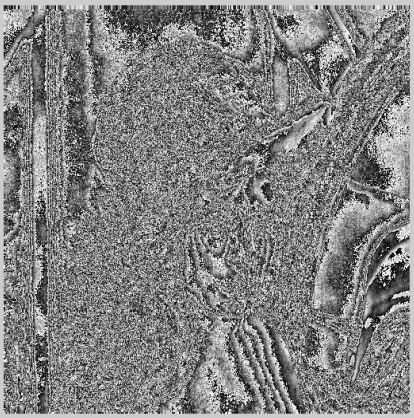
Şekil 5.10.d. Kod Kitabı: 512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 52.192

Şekil 5.10.e. Orijinal Boats

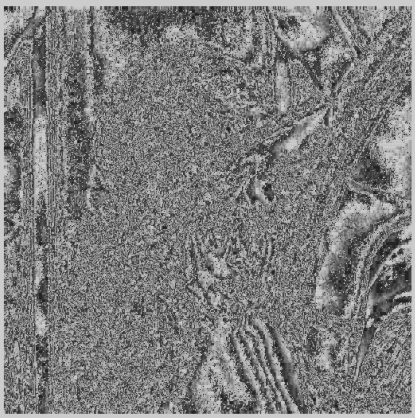
Şekil 5.10.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü PSNR: 28.361

5.2.2.2 8 bitlik görüntülerde son dört bit görüntünün düşük değerli bitlerini ve ilk dört bit yüksek değerli bitlerini ayırıp, yüksek değerli bitler için kullanılan 512 kod kitabının, (4 x 4) bloklar halinde, düşük değerli bitlere uygulanması sonucunda elde edilen görüntüler;

Şekil 5.11.a ve Şekil 5.12.a 4 bit düşük değerli bitler ve Şekil 5.11.b ve Şekil 5.12.b yüksek değerli bitler için en uygun bulunan 512 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.11.c ve Şekil 5.12.c 4 bit yüksek değerli bitleri ve Şekil 5.11.d ve Şekil 5.12..d 512 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.11.e orijinal görüntü ile Şekil 5.11.f, Şekil 5.11.b ve Şekil 5.11.d' nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır. Şekil 5.12.e orijinal görüntü ile Şekil 5.12.f, Şekil 5.12.b ve Şekil 5.12.d' nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır. (Kesme Metodu)



Şekil 5.11.a. 4 bit Lena LSB



Şekil 5.11.b. Kod Kitabı: 512
MSB için kullandığımız kod kitabı kullanarak
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 37.848

Şekil 5.11.c. 4 bit Lena MSB

Şekil 5.11.d. Kod Kitabı: 512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 52.276

Şekil 5.11.e. Orijinal Lena

Şekil 5.11.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 28.542

Şekil 5.12.a. 4 bit Boats LSB

Şekil 5.12.b. Kod Kitabı: 512
MSB için kullandığımız kod kitabı kullanarak
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 38.306

Şekil 5.12.c. 4 bit Boats MSB

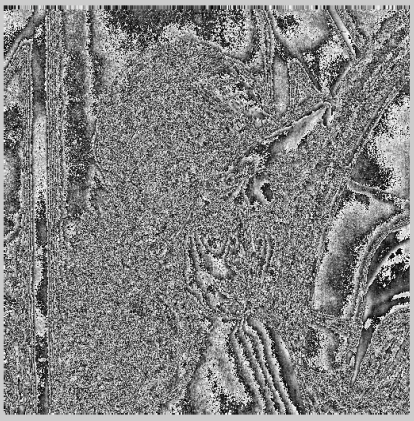
Şekil 5.12.d. Kod Kitabı: 512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 52.192

Şekil 5.12.e. Orijinal Boats

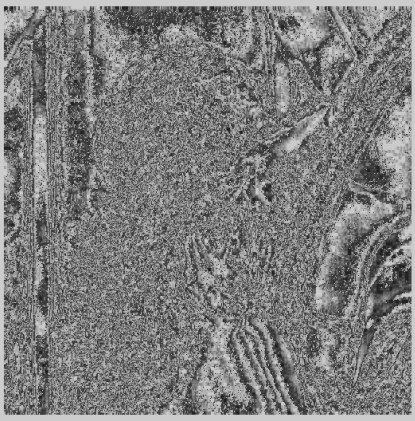
Şekil 5.12.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 28.362

5.2.2.3 8 bitlik görüntülerde son dört bit görüntünün düşük değerli bitlerini ve ilk dört bit yüksek değerli bitlerini ayırıp, yüksek değerli bitler için (4 x 4) bloklar halinde, en uygun 512 kod kitabı bulunup, düşük değerli bitler içinde 64 kod kitabı bulunup, toplam (512 + 64) = 576 kod kitabı ile ayrı ayrı uygulanan vektör kuantalama sonucunda elde edilen görüntüler;

Şekil 5.13.a ve Şekil 5.14.a 4 bit düşük değerli bitler ve Şekil 5.13.b ve Şekil 5.14.b 576 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.13.c ve Şekil 5.14.c 4 bit yüksek değerli bitleri ve Şekil 5.13.d ve Şekil 5.14..d 576 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.13.e orijinal görüntü ile Şekil 5.13.f, Şekil 5.13.b ve Şekil 5.13.d' nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır. Şekil 5.14.e orijinal görüntü ile Şekil 5.14.f, Şekil 5.14.b ve Şekil 5.14.d' nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır.



Şekil 5.13.a. 4 bit Lena LSB



Şekil 5.13.b. Kod Kitabı: 576
1:28 (4 bit) Sıkıştırma Oranı
PSNR: 37.968

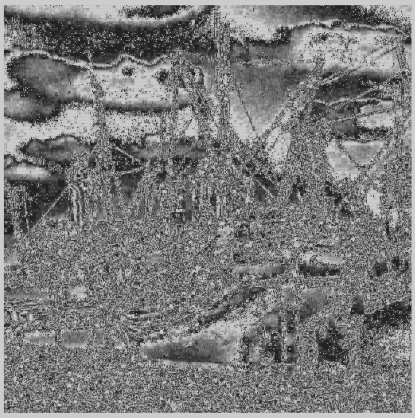
Şekil 5.13.c. 4 bit Lena MSB

Şekil 5.13.d. Kod Kitabı: 576
1:28 (4 bit) Sıkıştırma Oranı
PSNR: 52.971

Şekil 5.13.e. Orijinal Lena

Şekil 5.13.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 29.179

Şekil 5.14.a. 4 bit Boats LSB



Şekil 5.14.b. Kod Kitabı: 576
1:28 (4 bit) Sıkıştırma Oranı
PSNR: 38.410

Şekil 5.14.c. 4 bit Boats MSB

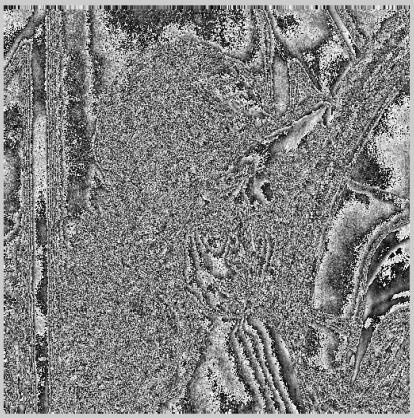
Şekil 5.14.d. Kod Kitabı: 576
1:28 (4 bit) Sıkıştırma Oranı
PSNR: 52.874

Şekil 5.14.e. Orijinal Boats

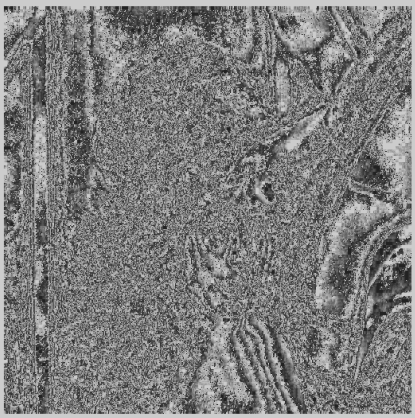
Şekil 5.14.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü PSNR: 29.039

5.2.2.4 8 bitlik görüntülerde son dört bit görüntünün düşük değerli bitlerini ve ilk dört bit yüksek değerli bitlerini ayırıp, yüksek değerli bitler için (4 x 4) bloklar halinde, en uygun 512 kod kitabı bulunup, düşük değerli bitler içinde 128 kod kitabı bulunup, toplam (512 + 128) = 640 kod kitabı ile ayrı ayrı uygulanan vektör kuantalama sonucunda elde edilen görüntüler;

Şekil 5.15.a ve Şekil 5.16.a 4 bit düşük değerli bitler ve Şekil 5.15.b ve Şekil 5.16.b 640 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.15.c ve Şekil 5.16.c 4 bit yüksek değerli bitleri ve Şekil 5.15.d ve Şekil 5.16..d 640 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.15.e orijinal görüntü ile Şekil 5.15.f, Şekil 5.15.b ve Şekil 5.15.d' nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır. Şekil 5.16.e orijinal görüntü ile Şekil 5.16.f, Şekil 5.16.b ve Şekil 5.16.d' nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır.



Şekil 5.15.a. 4 bit Lena LSB



Şekil 5.15.b. Kod Kitabı: 640
1:26 (4 bit) Sıkıştırma Oranı
PSNR: 38.041

Şekil 5.15.c. 4 bit Lena MSB

Şekil 5.15.d. Kod Kitabı:640
1:26 (4 bit) Sıkıştırma Oranı
PSNR: 53.139

Şekil 5.15.e. Orijinal Lena

Şekil 5.15.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 29.336

Şekil 5.16.a. 4 bit Boats LSB

Şekil 5.16.b. Kod Kitabı: 640
1:26 (4 bit) Sıkıştırma Oranı
PSNR: 38.563

Şekil 5.16.c. 4 bit Boats MSB

Şekil 5.16.d. Kod Kitabı:640
1:26 (4 bit) Sıkıştırma Oranı
PSNR: 52.831

Şekil 5.16.e. Orijinal Boats

Şekil 5.16.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 29.331

5.2.2.5 8 bitlik görüntülerde 1.bitten başlayıp birer atlayarak giden bitler görüntünün düşük değerli bitlerini ve diğerleri de yüksek değerli bitlerini oluşturup, (4 x 4) bloklar halinde 512 kod kitabı ayrı ayrı uygulanan vektör kuantalama sonucunda elde edilen görüntüler;

Şekil 5.17.a ve Şekil 5.18.a 4 bit düşük değerli bitler ve Şekil 5.17.b ve Şekil 5.18.b 512 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.17.c ve Şekil 5.18.c 4 bit yüksek değerli bitleri ve Şekil 5.17.d ve Şekil 5.18..d 512 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.17.e orijinal görüntü ile Şekil 5.17.f, Şekil 5.17.b ve Şekil 5.17.d' nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır. Şekil 5.18.e orijinal görüntü ile Şekil 5.18.f, Şekil 5.18.b ve Şekil 5.18.d'nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır.(Taraklama Metodu)

Şekil 5.17.a. 4 bit Lena LSB

Şekil 5.17.b. Kod Kitabı: 512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 41.746

Şekil 5.17.c. 4 bit Lena MSB

Şekil 5.17.d. Kod Kitabı:512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 45.893

Şekil 5.17.e. Orijinal Lena

Şekil 5.17.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 23.605

Şekil 5.18.a. 4 bit Boats LSB

Şekil 5.18.b. Kod Kitabı: 512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 41.911

Şekil 5.18.c. 4 bit Boats MSB

Şekil 5.18.d. Kod Kitabı:512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 45.963

Şekil 5.18.e. Orijinal Boats

Şekil 5.18.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü PSNR: 23.549

5.2.2.6 8 bitlik görüntülerde 1.bitten başlayıp birer atlayarak giden bitler görüntünün düşük değerli bitlerini ve diğerleri de yüksek değerli bitlerini oluşturup, (4 x 4) bloklar halinde, yüksek değerli bitler için kullanılan 512 ve 1024 kod kitabı düşük değerli bitlerede uygulanması sonucunda elde edilen görüntüler;

Şekil 5.19.a ve Şekil 5.20.a 4 bit düşük değerli bitler ve Şekil 5.19.b ve Şekil 5.20.b yüksek değerli bitler için en uygun bulunan 512 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.21.a ve Şekil 5.22.a 4 bit düşük değerli bitler ve Şekil 5.21.b ve Şekil 5.22.b yüksek değerli bitler için en uygun bulunan 1024 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.19.c ve Şekil 5.20.c 4 bit yüksek değerli bitleri ve Şekil 5.19.d ve Şekil 5.20..d 512 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.21.c ve Şekil 5.22.c 4 bit yüksek değerli bitleri ve Şekil 5.21.d ve Şekil 5.22..d 1024 kod kitabı ile VQ uygulandıktan sonraki görüntüleri içerir. Şekil 5.19.e, Şekil 5.20.e, Şekil 5.21.e, Şekil 5.22.e orijinal görüntüleri ile Şekil 5.19.f, Şekil 5.20.f, Şekil 5.21.f, Şekil 5.22.f, Şekil 5.19.b ve Şekil 5.19.d' nin, Şekil 5.20.b ve Şekil 5.20.d' nin, Şekil 5.21.b ve Şekil 5.21.d' nin, Şekil 5.22.b ve Şekil 5.22.d' nin birleştirilerek 8 bit oluşturulmuş hali ile karşılaştırılmaktadır. (Taraklama Metodu)

Şekil 5.19.a. 4 bit Lena LSB

Şekil 5.19.b. Kod Kitabı: 512
MSB için kullandığımız kod kitabı kullanarak
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 41.673

Şekil 5.19.c. 4 bit Lena MSB

Şekil 5.19.d. Kod Kitabı:512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 45.893

Şekil 5.19.e. Orijinal Lena

Şekil 5.19.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 23.713

Şekil 5.20.a. 4 bit Boats LSB

Şekil 5.20.b. Kod Kitabı: 512
MSB için kullandığımız kod kitabı kullanarak
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 41.807

Şekil 5.20.c. 4 bit Boats MSB

Şekil 5.20.d. Kod Kitabı:512
1:32 (4 bit) Sıkıştırma Oranı
PSNR: 45.963

Şekil 5.20.e. Orijinal Boats

Şekil 5.20.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 23.502

Şekil 5.21.a. 4 bit Lena LSB

Şekil 5.21.b. Kod Kitabı: 1024
MSB için kullandığımız kod kitabı kullanarak
1:16 (4 bit) Sıkıştırma Oranı
PSNR: 42.485

Şekil 5.21.c. 4 bit Lena MSB

Şekil 5.21.d. Kod Kitabı:1024
1:16 (4 bit) Sıkıştırma Oranı
PSNR: 46.539

Şekil 5.21.e. Orijinal Lena

Şekil 5.21.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 23.677

Şekil 5.22.a. 4 bit Boats LSB

Şekil 5.22.b. Kod Kitabı:1024
MSB için kullandığımız kod kitabı kullanarak
1:16 (4 bit) Sıkıştırma Oranı
PSNR: 42.647

Şekil 5.22.c. 4 bit Boats MSB

Şekil 5.22.d. Kod Kitabı:1024
1:16 (4 bit) Sıkıştırma Oranı
PSNR: 46.564

Şekil 5.22.e. Orijinal Boats

Şekil 5.22.f. 8 bit vektör kuantalamadan sonra elde edilmiş görüntü
PSNR: 23.614

EKLER

Ek 1: Bit-Dilimi Programları:

A.1.

```
clear;
clc;
a=[115 86 34 209; 48 75 100 231; 185 18 45 11; 238 15 150 191];
fid=fopen('dnm4.bmp','w');
fwrite(fid,a,'char');
fid=fopen('dnm4.bmp','r');
b=fread(fid);
Resim=reshape(b,4,4);
Resim=uint8(Resim)
Size_Resim=size(Resim);
%Resmin her elamanini and islemine tabi tutabilmek icin
%Resim boyutunda elemanlari 15 olan matrix olusturuluyor
% 15 decimal rakami 00001111 binary rakamina karsilik geliyor
%Ayni islem 240 decimal 11110000 binary islemine tabi tutuluyor
% bitand islemi sadece unsigned integerde tanimli oldugu icin
%bu matrix uint8 yardimiyla unsigned integere cevriliyor
Low_Decimal=uint8(15*ones(Size_Resim));
High_Decimal=uint8(240*ones(Size_Resim));
%Bit wise and leme islemi yapiliyor
Low_Resim=(bitand(Resim,Low_Decimal))
High_Resim=bitand(Resim,High_Decimal);
High_Resim=(bitshift(High_Resim,-4))

figure
subplot(1,3,1)
imshow(Resim)
```

```
subplot(1,3,2)
imshow(uint8(17*double(Low_Resim)))
subplot(1,3,3)
imshow(uint8(17*double(High_Resim)))

pause
close all
```

A.2.

```
clear;
clc;
%Resim okunuyor
Resim=imread('lena.tif');
%Resim=Resim(1:240,:);
%Resmin buyuklugu bulunuyor
Size_Resim=size(Resim);
%Resmin her elamanini and islemine tabi tutabilmek icin
%Resim boyutunda elemanlari 15 olan matrix olusturuluyor
% 15 decimal rakami 00001111 binary rakamina karsilik geliyor
%Ayni islem 240 decimal 11110000 binary islemine tabi tutuluyor
% bitand islemi sadece unsigned integerde tanimli oldugu icin
%bu matrix uint8 yardimiyla unsigned integere cevriliyor

Low_Decimal=uint8(15*ones(Size_Resim));
High_Decimal=uint8(240*ones(Size_Resim));
%Bit wise and leme islemi yapiliyor
Low_Resim=(bitand(Resim,Low_Decimal));
High_Resim=bitand(Resim,High_Decimal);
High_Resim=(bitshift(High_Resim,-4));

figure
subplot(1,3,1)
imshow(Resim)
subplot(1,3,2)
imshow(uint8(17*double(Low_Resim)))
subplot(1,3,3)
imshow(uint8(17*double(High_Resim)))

pause
close all
```

A.3.

```
clear;
clc;

Resim=imread('lena.tif');
Size_Resim=size(Resim);

%Resmin her elamanini and islemine tabi tutabilmek icin
%Resim boyutunda elemanlari 15 olan matrix olusturuluyor
% 15 decimal rakami 00001111 binary rakamina karsilik geliyor
%Ayni islem 240 decimal 11110000 binary islemine tabi tutuluyor
% bitand islemi sadece unsigned integerde tanimli oldugu icin
%bu matrix uint8 yardimiyla unsigned integere cevriliyor
Low_Decimal=uint8(15*ones(Size_Resim));
High_Decimal=uint8(240*ones(Size_Resim));
%Bit wise and leme islemi yapiliyor
Low_Resim=(bitand(Resim,Low_Decimal));
High_Resim=bitand(Resim,High_Decimal);
High_Resim=(bitshift(High_Resim,-4));

figure
subplot(2,3,1)
imshow(Resim)
subplot(2,3,2)
imshow(uint8(17*double(Low_Resim)))
subplot(2,3,3)
imshow(uint8(17*double(High_Resim)))

Low_Decimal=uint8(1*ones(Size_Resim));
Low_Resim=(bitand(Resim,Low_Decimal));
bit1=Low_Resim;
Low_Decimal=uint8(4*ones(Size_Resim));
```

```
Low_Resim=(bitand(Resim,Low_Decimal));
bit2=bitshift(Low_Resim,-1);
Low_Decimal=uint8(16*ones(Size_Resim));
Low_Resim=(bitand(Resim,Low_Decimal));
bit3=bitshift(Low_Resim,-2);
Low_Decimal=uint8(64*ones(Size_Resim));
Low_Resim=(bitand(Resim,Low_Decimal));
bit4=bitshift(Low_Resim,-3);
Low_Resim=double(bit1)+double(bit2)+double(bit3)+double(bit4);

High_Decimal=uint8(2*ones(Size_Resim));
High_Resim=(bitand(Resim,High_Decimal));
bit1=bitshift(High_Resim,-1);
High_Decimal=uint8(8*ones(Size_Resim));
High_Resim=(bitand(Resim,High_Decimal));
bit2=bitshift(High_Resim,-2);
High_Decimal=uint8(32*ones(Size_Resim));
High_Resim=(bitand(Resim,High_Decimal));
bit3=bitshift(High_Resim,-3);
High_Decimal=uint8(128*ones(Size_Resim));
High_Resim=(bitand(Resim,High_Decimal));
bit4=bitshift(High_Resim,-4);
High_Resim=double(bit1)+double(bit2)+double(bit3)+double(bit4);

subplot(2,3,4)
imshow(Resim)
subplot(2,3,5)
imshow(uint8(17*double(Low_Resim)))
subplot(2,3,6)
imshow(uint8(17*double(High_Resim)))
pause
close all
```

Ek 2: Direct VQ (LBG) Programları:

```
function v=lvq(x,para,p)
%initialize
etaz=para.etaz;
N=para.it;
i=0;
n=size(x,1);
%c=512;
c=para.clusters;
fno=para.fno;
if(p==0)
s=para.s(3);
else
  s=para.s(p);
end
%------------------------

%initialize codebook

v=rand(c,s*s);
v=v*max(max(double(x)));
imwrite(uint8(v),'codebook.bmp');
v=imread('codebook.bmp');
v=double(v);

%------------------------
```

```
N1=1;
if(para.cit~=0)
   N1=para.cit;
   N=10;%+para.step;
end

 for i=N1:N
    eta=etaz*(1-i/N);

   for j=1:n
    %find winning prototype by euclidean
    %mini=(norm(x(j,:)-v(1,:)));
     %dif(1)=mini;
     %win=1;
      %for k=2:c
       % dif(k)=(norm(x(j,:)-v(k,:)));
        %   if(dif(k)<mini)
        %   mini=dif(k);
        %   win=k;
        %end

        %end

        mini=(((repmat(x(j,:),[c 1])-v(:,:)).^2))';
         %--------------------
        %t=find(mini==0);
        %mini(t(1:end))=.01;
        %t=find(mini==inf);
        %mini(t(1:end))=10000;
        %---------------------
if(not (size(x,2)==1))
   mini=(sum(mini))';
```

```
end
mini=sqrt(mini);

m=min(mini);
temp=find(mini==m);
if(length(temp)==1)
   win=temp;
else
win=temp(1);
end
dif=mini;
    %-------------------------------------
        %calculate u,w,n
    %--------------------
    [u ,w ,nw]=findweight(dif,win,c,fno,para.alpha,para.beta,para.gama);
    %--,---------------------
        %update prototype
        vnew=v(win,:)+eta*(x(j,:)-v(win,:)).*(1+sum(w));
      for k=1:c
        v(k,:) = v(k,:) + eta.*(x(j,:)-v(k,:)).*nw(k);
     end
    v(win,:)=vnew;
       clear temp;
 end
 end
v=v(:,1:s*s);


function y=lvqdecode(x,v)
xorig=x;
x=double(x);
nx=size(x,1);
```

```
ny=size(x,2);
s=sqrt(size(v,2));
c=size(v,1);

%y=cell(size(x,1),1);
%y(i,:)=v(index(i),:);
%if(s==4)
%y{i}=[v(index(i),1:s:s*s);v(index(i),2:s:s*s);v(index(i),3:s:s*s);v(index(i),4:s:s*s)];
%elseif(s==2)
 %   y{i}=[v(index(i),1:s:s*s);v(index(i),2:s:s*s)];
 %end
%clear temp;
%end

t=v(x(:),:)';
%--------change structure of the y--------------------------
%y1=reshape(temp,nx*s,ny*s);

y=col2im(t,[s s],[nx*s ny*s],'distinct');



function [y]=testlvq1(x)
tic; %start watch
global para;
s=para.s(1);
x=double(x);

%preprocess----------------------
rn=size(x,1);
cn=size(x,2);
rem1=mod(rn, s);
rem2= mod(cn,s);
```

```
rn=rn-rem1;
cn=cn-rem2;
x=x(1:rn,1:cn);
x1=im2col(x,[s s],'distinct');
x1=x1';

%compress and decompress
for i=1:1
   if(i==1)
   v=imread('codebook.bmp');
   s1='Reconstructed';
   else
%        s1='Reconstructed- proposed';
%    v=imread('codebookp.bmp');
   end

   c=size(v,1);

yc=lvqcode(x,v);
yp=lvqdecode(yc,v);
%------------------------
if(i==1)
   y=yp;
end
y=uint8(yp);
end
toc; % stop watch


function [u, w, nw]=findweight(diff,win,c,fno,alpha,beta,gamma)
%       diff is the ||x-v||^2
%       win is the index of winning prototype
```

```
%        c->no.of prototypes
% alpha=1;
% beta=1;
% gamma=.5;
diff=diff+.1;
z=diff(win)./diff;
z(win)=0;
if(fno==1)

u=z./(1+alpha.*z);
u(win)=1;
w=1./(1+alpha.*z).^2;
nw=alpha.*u.^2;
elseif(fno==2)
   e=exp(-beta.*z);
   u=z.*(e);
w=(1-beta.*z).*e;
nw=beta.*u.^2.*e;
else
   u=z.*(1-gamma.*z);
w=(1-2*gamma.*z);
nw=gamma.*z.^2;
end


u=u';
nw=nw';
w=w';

w(win)=0;
n(win)=0;
```

```
function y=lvqcode(x,v)
v=double(v);
xorig=x;
nx=size(x,1);
ny=size(x,2);
x=double(x);
s=sqrt(size(v,2));
%take blockwise
x1=im2col(x,[s s],'distinct');
x=x1';

c=size(v,1);
n=size(x,1);

%-------------find nearest code----------------------
for i=1:n
mini=(((repmat(x(i,:),[c 1])-v(:,:)).^2))';
if(not (size(x,2)==1))
   mini=(sum(mini))';
end
mini=sqrt(mini);
m=min(mini);
temp=find(mini==m);
if(length(temp)==1)
   index(i)=temp;
else
index(i)=temp(1);
end
clear temp;
```

```
end
%y=col2im(index,[1 1],[nx/s ny/s],'distinct');
y=reshape(index,nx/s,ny/s);
% training vector quantization

function [v]=trainlvq(x,p)
global para;
p=0;
x=imread('tarakcamlsb.tif');
%c=128;
c=para.clusters;
if(p==0)
s=para.s(3);
else
   s=para.s(p)
end
x=double(x);
%preprocess----------------------
rn=size(x,1);
cn=size(x,2);
rem1=mod(rn,s);
rem2= mod(cn,s);
rn=rn-rem1;
cn=cn-rem2;
x=x(1:rn,1:cn);

x1=im2col(x,[s s],'distinct');
x1=x1';

v=lvq(x1,para,p);
%vp=lvqproposed(x1,para,p);
if(p==0)
```

```
 imwrite(uint8(v),'codebook.bmp');
%imwrite(uint8(vp),'codebookp.bmp');
end
% function for finding MSE and PSNR
function p=psnr(x,xc,g)
x=double(x);
xc=double(xc);
t=(x-xc).^2;
disp('s');
s=sum(sum(t));
disp(s);
%p=g*g*(size(x,1)*size(x,2))/(s);
n=size(x,1)*size(x,2);

disp('p');
p=(g*g)/(s/n);
p=10*log10(p);
disp(p);


%MAIN PROGRAM
load parameter;

global para

x=imread('tarakcamlsb.tif');

original=x;

% v=rand(256,16);
% imwrite(uint8(v),'codebook.bmp');
```

```
[v]=trainlvq(x,0);
compressed=v;

%imwrite(uint8(vp),'codebook.bmp');

[y]=testlvq1(x);

 [psnrvalue]=psnr(original,y,255);

% error=0;
%   for y=1:240
%      for x=1:291
%         MSE=(((Original(x,y))-(DecompressedImage(x,y)))^2);
%         error=MSE+error;
%     end
%  end
%  MSE=(1/(291*240))*error;
%  disp('PSNR');
%  PSNR=20*log10(255/sqrt(MSE));
%  disp(PSNR);
%
figure,imshow(original);
title('Original Image');

% figure,imshow(compressed);
% title('compressed Image');

figure,imshow(uint8(y));
title('Decompressed Image');
%imwrite(uint8(y),'tarakcamlsbsamevq.tif')
pause
```

Ek 3: Bit-Düzlem Diliminin Ayrıştırılması:

Bu çalışma süresinde 8-bit görüntüler üzerinde çalışılmıştır. Parçalı bit-düzlem dilimlemesini iki grup halinde 8-bitlik görüntüleri 4'er bit görüntüler halinde vektör kuantalama yapmak üzere ayrıştırdık.

a. Kesme:

8-bit bir görüntü 2 bit-düzlem diliminde işlenerek sonra 1. bit düzlemi görüntünün düşük değerli bitlerini (LSB) 2.bit düzlemi de görüntünün yüksek değerli bitlerini (MSB) oluşturur. 8 bitlik görüntülerde son dört bit görüntünün düşük değerli bitlerini ve ilk dört bit yüksek değerli bitleri oluşturmaktadır.

Örnek:

1 0 1 1 0 1 0 1 8-bit bir görüntümüz olsun,

1 0 1 1 <u>0 1 0 1</u> Son dört bit görüntünün düşük değerli bitlerini,

<u>1 0 1 1</u> 0 1 0 1 İlk dört bit görüntünün yüksek değerli bitlerini,

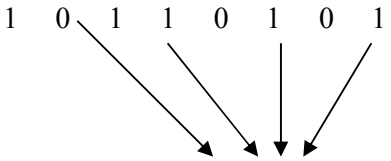
oluşturur.

b. Taraklama:

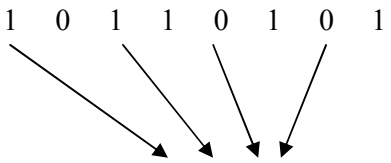
8 bitlik görüntülerde 1.bitten başlayıp birer atlayarak giden bitler görüntünün düşük değerli bitlerini ve diğerleri de yüksek değerli bitlerini oluşturur.

Örnek:

1 0 1 1 0 1 0 1 8-bit bir görüntümüz olsun,



0 1 1 1 görüntünün düşük değerli bitlerini,



1 1 0 0 görüntünün yüksek değerli bitlerini,

oluşturur.

KAYNAKLAR DİZİNİ

Cay,A., Li,W., Zhang Y., On the Optimal Transform for Vector Quantization of Images, IEEE, ISCAS'93, Chicago, 687-690.

Dedeakayoğulları, B., 1997, Yüksek Lisans Tezi, Osmangazi Üniversitesi Fen Bilimleri Enstitüsü.

Gerek, Ö.N and Çetin, A.E., Image Coding, Anadolu University, Departmant of Electrical and Electronics Engineering, 26 p.

Gerek, Ö.N and Çetin, A.E., Vector Quantization, Anadolu University, Departmant of Electrical and Electronics Engineering, 24 p.

Gerso, A. and Gray, R.M, 1992, Vector Quantization and Signal Processing, Kluwer Academic Publishers, 732 p.

Gonzalez, R.C and Gray, R.M., 2001, Digital Image Processing Second Edition, Printice Hall, 793 p.

Lu, Zheming, Xu Runsheng, Shenghe Sun, A Tabu Search Based Maximum Descent Algorithm for VQ Codebook Design, Intelligent Control Research Institute, Shanghai Jiaotong University, China

Nasrabadi, N. and Feng, Y., 1990, Image Compression Using Address-Vector Quantization, IEEE Transaction on Communications, Vol.38, No.12.

Özkan, K., 2000, Yüksek Lisans Tezi, LBG Algoritmasında Görüntü İstatistiklerinden Faydalanma, Osmangazi Üniversitesi Fen Bilimleri Enstitüsü, 113 s.

Wang, Z. and Bovik A.C, 2002, Bitplane-by-Bitplane Shift (BbB Shift)- A Suggestion for JPEG2000 Region of Intrest Image Coding, 160-162.

http://cobweb.ecn.purdue.edu/~ace/jpeg-tut/jpegtut1.html

http://bmrc.berkeley.edu/frame/research/mpeg

http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html

http://ocw.mit.edu/NR/rdanlyers/Electrical-Engineering-and-Computer-Science