T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÖRÜNTÜ SIKIŞTIRMA ALGORİTMALARI

Mustafa TÜRK

YÜKSEK LİSANS TEZİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

1998
ELAZIĞ

T.C.

FIRAT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GÖRÜNTÜ SIKIŞTIRMA ALGORİTMALARI

MUSTAFA TÜRK

YÜKSEK LİSANS TEZİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

ANABİLİM DALI

Bu tez, ...25.03.1998....Tarihinde Aşağıda Belirtilen Jüri Tarafından Oybirliği/~~Oyçokluğu~~ ile Başarılı/~~Başarısız~~ Olarak Değerlendirilmiştir.

(imza) (imza) (imza)

Danışman:

Yrd. Doç. Dr. Fikret ATA

ÖZET

Yüksek Lisans Tezi
Görüntü Sıkıştırma Algoritmaları

Mustafa TÜRK

Fırat Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik Elektronik Anabilim Dalı

1998, Sayfa:95

Bu çalışmada, değişik görüntü sıkıştırma algoritmaları ve wavelet dönüşümünün görüntü sıkıştırma alanındaki uygulamaları gerçekleştirilmeye çalışılmıştır.

Görüntü sıkıştırma, görüntünün iletilmesi ve kaydedilmesi sırasındaki zaman ve kapladığı hafıza bölgesinin azaltılması için yapılan bir işlemdir. Aynı zamanda da görüntü, veri fazlalıklarından arındırıldığı için görüntü kalitesinin artması da söz konusudur.

Çok çözünürlülük analizi olarak adlandırılan ve görüntüyü farklı alt görüntülere ayırarak görüntüdeki piksel fazlalıklarından kurtulmamızı sağlayan yöntemin bilgisayar simülasyonu Matlab programlama dili kullanılarak yapılmış ve farklı sıkıştırma oranları için uygulamalar gerçekleştirilmiştir. Bu çalışmada kayıplı görüntü sıkıştırma tekniği ele alınmış ve sıkıştırma oranı ile kayıpların nasıl değiştiği incelenmiştir.

Ele alınan yöntemin nasıl daha iyi sonuçlar vereceği tartışılmıştır.

ANAHTAR KELİMELER: Görüntü Sıkıştırma, Algoritmalar, Wavelet, HFD, JPEG, MPEG2, H.261, Hufmann, Aritmetik Kodlama.

SUMMARY

Master Thesis

Image Compression Algorithms

Mustafa TÜRK

Fırat University
Graduate School of Natural and Applied Sciences
Department of Electrical and Electronics Engineering

1998, Pages:95

This thesis covers, application of various image compression techniques and wavelet transformation on image compression.

Image compression is applied to reduce time and memory consumption during image recording and transmission. Compression also results with an image of better quality because of eliminating data redundancies of image.

Computer simulation of a method called multiresolution analysis that eliminates pixel redundancies dividing an image into different sub-images was carried out using matlab programming language and applications for different compression ratios were realized. In this study, loosly image compression method and the loose variation versus compression ratio was examined.

It was also discussed how to obtain better result with this method.

KEY WORDS: Image Compression, Algorithms, Wavelets, FFT, JPEG, MPEG2, H.261, Hufmann Coding, Arithmetic Coding.

TEŞEKKÜR

Tez çalışması sırasında bilgi ve tecrübelerinden yararlandığım Yrd. Doç. Dr. Fikret ATA ve Yrd. Doç. Dr. Ahmet ARSLAN 'a teşekkür ederim.

İÇİNDEKİLER

Sayfa

ŞEKİLLER LİSTESİ

Şekil no **Sayfa**

TABLOLAR LİSTESİ

Tablo no **Sayfa**

SİMGELER

Cr: Sıkıştırma Oranı

N: Doğal Sayılar

Ψ: Ana Wavelet Fonksiyonu

Φ: Baba Wavelet Fonksiyonu

C_k: Orijinal Dizi

I: Tanım Aralığı

$\mathbf{Z}$: Tam sayılar

$\mathfrak{R}$: Reel Sayılar

$\cup$: Birleşim Kümesi

$\cap$: Kesişim Kümesi

h_n: Alçak Geçiren Filtre Katsayıları

g_n: Yüksek Geçiren Filtre Katsayıları

Σ: Toplam Fonksiyonu

Π: Çarpım Fonksiyonu

d^k: Fark Dizisi

~: Kompleks Eşlenek

e_n: Fark Görüntüsü

δ: Fark Bileşeni

<,>:Konvolüsyon Çarpımı

1. GİRİŞ

1.1. Sıkıştırmaya Genel Bir Bakış

Sıkıştırma bir sinyalin dijital temsilinden ibaret olup literatürde veri sıkıştırma, kaynak kodlama veya sinyal kodlama olarak geçer. Sıkıştırma olmadan bir çok iletim uygulaması gerçekleştirilebilir olamaz, olsa da hem zaman ve hem de veri alanı yönünden ciddi problemler ortaya çıkartacaktır.

Örneğin kopya edilerek alınmış bir görüntünün iletimini düşünelim. Genellikle 8.5 x 11 inç 'lik sayfalar inç başına 200 piksel içerirler ve dolayısıyla 3.74 Mbit yer tutarlar. 14400 kbit/s 'lik bir modem ile bu görüntünün iletimi 5.62 dakika sürer. Sıkıştırma ile bu süre 17 sn 'ye kadar düşebilir(Bhaskaran, 1996).

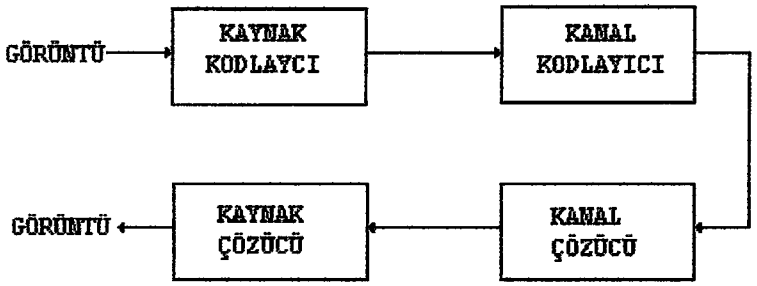
Görüntünün sıkıştırılmasında kullanılan dönüşüm oldukça önemlidir. Çünkü filtreleme etkisi göstererek piksellerdeki veri fazlalılıklarından bizi kurtaracaktır. Bu çalışmada bu dönüşümlerden wavelet dönüşümü incelenecek ve diğer dönüşümlere göre avantajları ele alınmaya çalışılacaktır.

Tablo 1.1 'de değişik sinyaller üzerinde yapılan sıkıştırma işleminin avantajları görülmektedir.

Tablo 1.1. Görüntü, video ve ses sıkıştırma uygulamaları

Uygulama	Veri oranı	
	Sıkıştırılmamış	Sıkıştırılmış
Ses sinyali	64 kbps	2-4 kbps
Video konferansı	30.41 Mbps	64-768 kbps
HDTV	1.33 Gbps	20 Mbps
Ağır çekim video	5.07 Mbps	8-16 kbps
CD 'den dijital video	60.83 Mbps	1.5-4 Mbps

Şekil 1.1 'de bir sıkıştırma sisteminin blok diyagramı verilmiştir. Kodlayıcının çekirdeği kaynak kodlayıcıdır. Kaynak kodlayıcı, giriş veri seviyesini iletim ve kayıt için makul bir seviyeye indirme işlemini gerçekleştirir.



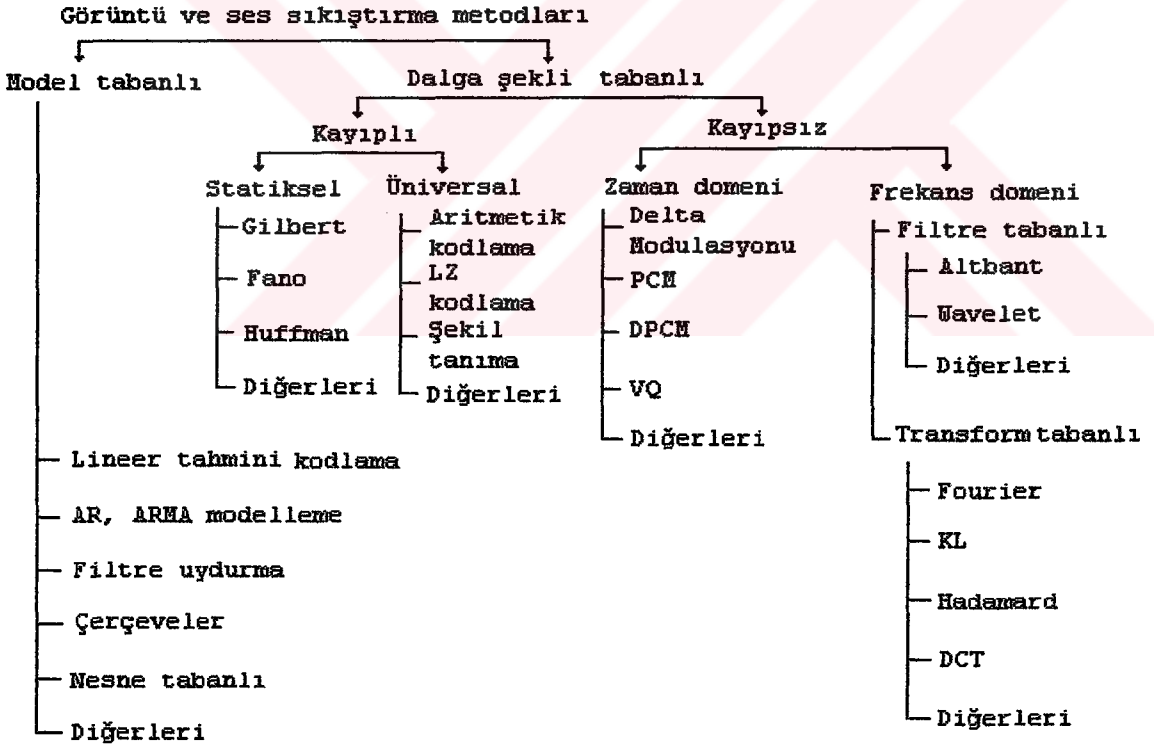
Şekil 1.1. Genel bir sıkıştırma sistemi

Literatürde sıkıştırma oranı cr ile verilir ve aynı zamanda bit oranı yerine de kullanılır.

$$cr = \frac{\text{kaynak kodlayacı giriş boyutu}}{\text{kaynak kodlayacı çıkış boyutu}}$$

1.2. Sıkıştırmanın Sınıflandırılması

Şekil 1.2 'de sıkıştırma metotlarının dağılımı görülmektedir.



Şekil 1.2. Sıkıştırma işleminin sınıflandırılması

2. AYRILABİLİR DÖNÜŞÜM METOTLARI

Görüntü işlemede kullanılan başlıca dönüşüm metodları Hızlı Fourier Dönüşümü(HFD), Haar, Walsh, Hadamard ve Ayrık Kosinüs Dönüşümüdür(AKD). Bunlardan en çok kullanılanları HFD ve AKD 'dir. HFD 'nin avantajı normalde gerekenden daha az çarpma ve toplamaya ihtiyaç duyması, AKD 'nin avantajı ise dönüşüme uğramış katsayıların çoğunun bir katsayıda yoğunlaşması ki bu da kodlamadan sonra iletimdeki fazlalıklardan kurtulmamızı sağlaması olarak gösterilebilir. Bu özelliğine enerji sıkıştırma özelliği denir. Diğer bir avantajı HFD algoritması kullanılarak gerçekleştirilebilir olmasıdır(Kayran, 1989).

2.1. Hızlı Fourier Dönüşümü Ve Temel Özellikleri

Normalde fourier dönüşümü için N^2 'lik toplam işleme ihtiyaç duyulurken HFD 'de ise $N \log_2 N$ 'lik toplam işleme ihtiyaç duyulur. En büyük avantajı da buradan gelmektedir. Tablo 2.1. 'de N 'in değerlerine göre direkt ve hızlı fourier dönüşümlerinin karşılaştırılması yapılmıştır.

Denklem (2.1) 'de bir boyutlu Fourier dönüşümünün ifadesi verilmiştir:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) W_N^{ux} \qquad (2.1)$$

Denklem (2.1) 'deki W_N değeri aşağıdaki gibi bulunur.

$$W_N = \exp\left[\frac{-j2\pi}{N}\right]$$

Denklem (2.1) 'deki W_N değeri aşağıdaki gibi bulunur.

$$W_N = \exp\left[\frac{-j2\pi}{N}\right]$$

Tablo 2.1. HFD 'nin hesaplama avantajı

N	N^2 (DİREKT FD)	N LOG2 N (HFD)	HESAP AVANTAJI
2	4	2	2.00
4	16	8	2.00
8	64	24	2.67
16	256	64	4.00
32	1024	160	6.40
64	4096	384	10.67
128	16384	896	18.29
256	65536	2048	32.00
512	262144	4608	56.89
1024	1048576	10240	102.40
2048	4194304	22528	186.18
4096	16777216	49152	341.33
8192	67108864	106496	630.15

Eğer $N=2^n$ olarak kabul edilirse ki burada n pozitif bir tam sayıdır $N=2M$ yazılabilir burada M de bir tamsayıdır. Bu değişkenler denklem (2.1) 'de yerine konursa

$$F(u) = \frac{1}{2M}\sum_{x=0}^{2M-1} f(x)W_{2M}^{ux}$$

$W_{2M}^{ux} = W_M^{ux}$ olduğuna göre denklemler yeniden düzenlenecek

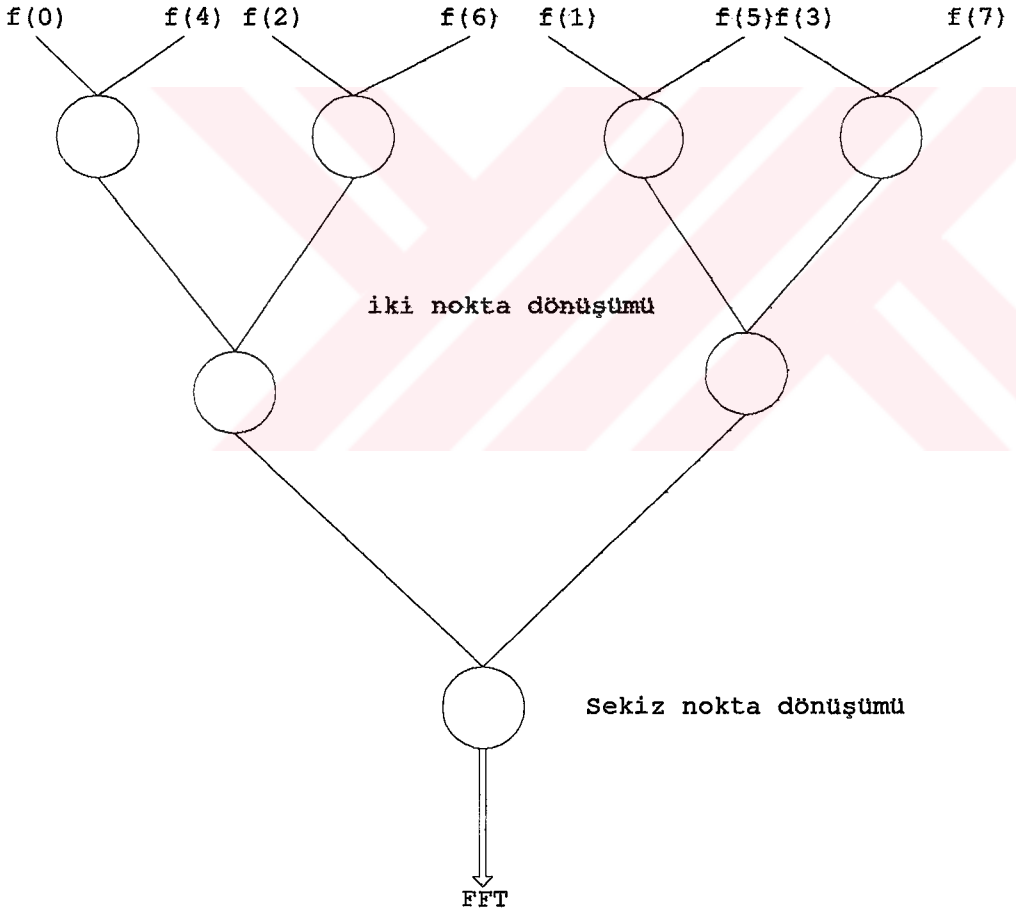
olursa aşağıdaki ifade elde edilir.

$$F(u) = \frac{1}{2}\left[\frac{1}{M}\sum_{x=0}^{M-1} f(2x)W_M^{ux} + \frac{1}{M}\sum_{x=0}^{M-1} f(2x+1)W_M^{ux}W_{2M}^{u}\right]$$

Yukarıdaki denklem tek ve çift olarak ayrılacak olursa (2.2) 'deki hali alacaktır.

$$F(u) = \frac{1}{2}[F_{çift}(u) + F_{tek}(u)W_{2M}] \qquad (2.2)$$

Şekil 2.1. 'de sekiz elemanlı bir dizinin ardışıl ikileme metodu kullanılarak üç nokta dönüşümü ile HFD 'sinin alınması gösterilmiştir.



Şekil 2.1. Sıralanmış giriş verileri ve ardışıl ikileme metodunun kullanımı

Tablo 2.2. n=8 için sıralanmış giriş değerleri

BİT SIRASI			ORIJINAL DİZİ	BİTLERI DÖNDÜRÜLMÜŞ ARGUMAN			SIRALANMIŞ DİZİ
0	0	0	f(0)	0	0	0	f(0)
0	0	1	f(1)	1	0	0	f(4)
0	1	0	f(2)	0	1	0	f(2)
0	1	1	f(3)	1	1	0	f(6)
1	0	0	f(4)	0	0	1	f(1)
1	0	1	f(5)	1	0	1	f(5)
1	1	0	f(6)	0	1	1	f(3)
1	1	1	f(7)	1	1	1	f(7)

Tablo 2.2.'de ardışıl ikileme metodu kullanılarak gelen bitlerin nasıl değiştirildiği ve yeni bit sıraları verilmiştir.

3. WAVELET VE WAVELET'IN TARİHÇESİ

Wavelet, verileri değişik frekans bölgelerine ayıran matematiksel bir fonksiyondur. Sinyalin sürekli olmayışı ve ani değişimler göstermesi durumunda analiz safhasında fourier tabanlı metotlar üzerinde belirgin bir avantajı vardır. Waveletlar kesin matematiksel belirlemelere ihtiyaç gösteren fonksiyonlar olup veriyi ya da fonksiyonları temsil etmekte kullanılırlar. Bu görüş yeni olmayıp Joseph Fourier tarafından 1800 'lü yıllarda fonksiyonların sinüs ve cosinüs fonksiyonları ile temsil edilebileceğini göstermiş idi. Bununla beraber ölçek wavelet dönüşümünde veriyi görmek için önemlidir. Wavelet algoritmaları veriyi değişik ölçek veya çözünürlülüklerde işler. Eğer sinyale geniş bir pencereden bakarsak verinin özelliklerini kabaca ancak görebiliriz. Eğer küçük bir pencereden veriye bakarsak özelliklerini yine tam olarak göremeyiz. Sonuç olarak üzerinde konuşmak için hem ormanı hem de ağaçları görmek gerekecektir. İşte bu wavelet 'ı ilginç ve faydalı yapar(Graphs, 1995).

3.1. Wavelet'ın Tarihçesi

3.1.1. 1930'ların öncesi

Waveletların gelişiminde Joseph Fourier(1807) ve onun fourier analizi önemli bir rol oynar. Her 2π periyodik fonksiyonun toplamının onun fourier dönüşümü olduğunu göstermiştir.

$$a_0 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx) \qquad (3.1)$$

a_0, a_k ve b_k katsayılarını ise denklem (3.2) deki gibi göstermiştir.

$$a_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)dx \,,\; a_0 = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(kx)dx \,,\; b_0 = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(kx)dx \quad (3.2)$$

1807 'den sonra fonksiyonların ortalamasının, fourier serisi yaklaşımı ve ortogonal sistemlerin bulunması ile matematikçiler frekans analizinden ölçek analizine doğru yöneldiler. Bu işlem, analiz ölçeği değişen matematiksel yapılar oluşturarak analiz etmektir. Acaba nasıl? Bir fonksiyonu oluşturmak, bir miktar kaydırmak ve ölçeği değiştirmekle olur. Şimdi bu işlemi tekrarlayalım. Temel bir yapı alın ve bir miktar kaydırın ve ölçeğini değiştirin. Onu aynı sinyale uygulayıp yeni bir yaklaşım elde edin. Böylece daha az duyarlı bir sinyal elde edilmiş olunur.

İlk kez wavelet terimi A.Haar(1909)'ın tez çalışmasında geçmiştir. Haar waveletının önemli bir özelliği belirli bir aralığın dışında zayıflayan olmasıdır. Maalesef Haar waveletlarının sürekli differansiyellenebilir olmamasından dolayı uygulama alanları oldukça dardır.

3.1.2. 1930'lar

1930'larda birkaç grup bağımsız olarak ölçek değişim temel fonksiyonlarını kullanarak fonksiyonların temsili üzerinde araştırma yaptılar. Temel fonksiyon kavramı nedir? Her iki boyutlu (x,y) vektörü (1,0) ve (0,1) vektörlerinin bir kombinasyonudur. x, (1,0) vektörü ile çarpılırsa (x,0) aynı şekilde (0,y) elde edilerek toplanırsa (x,y) vektörü elde edilir. Bu (1,0) ve (0,1) vektörleri temel fonksiyonlardır.

3.1.3. 1980 'ler

Stephane Mallat(1985) qmf, piramit algoritmalar ve ortonormal wavelet tabanları arasında bazı ilişkiler kurmuştur. Bu Haar 'ın tersine sürekli olarak türevleri alınabilir. Daha sonraları Ingrid Daubechies bugünkü waveletın temellerini atmıştır.

3.2. Wavelet ve Fourier Dönüşümü

3.2.1. Fourier dönüşümü ile olan benzerlikleri

Wavelet dönüşümü veriyi ya da fonksiyonu ya da operatörleri değişik frekans bileşenlerine parçalayan ve ona uyan bir çözünürlülük skalası ile her bir bileşen ile çalışan bir araçtır.
Standart fourier dönüşümü

$$(Ff)(w) = \frac{1}{\sqrt{2\pi}} \int e^{-iwt} f(t)dt$$

ile verilir. Bu dönüşüm frekans bileşenlerini verir ama yüksek frekans bileşenlerini kolayca veremez. İyi bir zaman belirleme (Time-localization) işlemi ancak ve ancak önce f sinyalini pencereleme ve daha sonra fourier dönüşümünü almakla olur(Graps, 1995).

$$(T^{win}f)(w, t) = \frac{1}{\sqrt{2\pi}} \int e^{-iws} f(s)g(s - t)ds \qquad (3.3)$$

Bu pencereleme işlemi zaman-sınırlama için tamamen standart bir işlemdir. Bu tam anlamı ile sinyal analizinin ayrık versiyonu ile benzerlikler gösterir. m ve n tam sayılar ve t_0 ile w_0 sabitler olmak üzere;

$$T_{m,n}^{win}(f) = \frac{1}{\sqrt{2\pi}} \int e^{-imw_0 s} f(s) g(s - nt_0) ds \qquad (3.4)$$

elde edilebilir.

En önemli benzerlik her iki dönüşümde de giriş verisinin boyutu 2^n olmasıdır. Ayrıca her ikisi de güç kayıplarını hesaplanmasında oldukça etkilidir.

3.2.2. Pencereli fourier dönüşümü ile olan ilişkisi

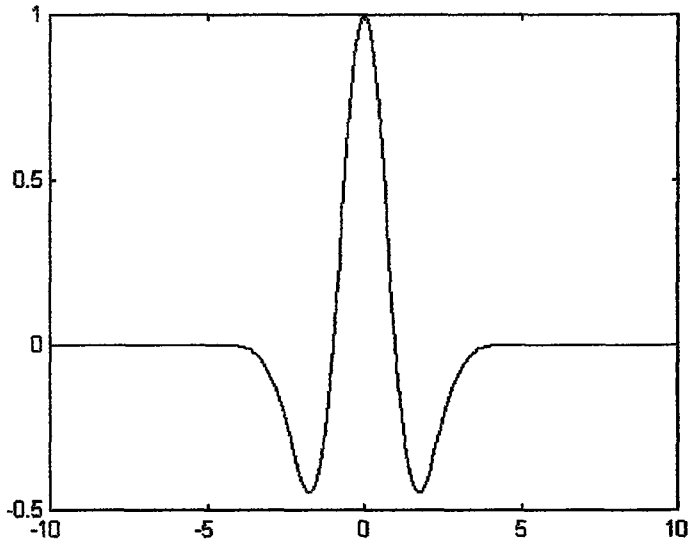
Wavelet dönüşümü bir kaç önemli farklılıkla beraber benzer özellikler gösterir. Aşağıda verilen wavelet dönüşüm formülleri (3.3) ve (3.4) nolu denklemlerle benzerlik gösterir.

$$(T^{wav} f)(a, b) = |a|^{-1/2} \int f(t) \Psi\left(\frac{t - b}{a}\right) dt \qquad (3.5)$$

$$T_{m,n}^{wav}(f) = a_0^{-m/2} \int f(t) \Psi(a_0^{-m} t - nb_0) dt \qquad (3.6)$$

Benzerliklerden birisi; hem denklem (3.3) hem de denklem (3.5) 'de iki değişken ile indekslenmiş bir fonksiyon ailesi f fonksiyonunun iç çarpımları alınmaktadır. Denklem (3.5) 'e geçen Ψ fonksiyonu tipik olarak $(1-t^2)\exp(-t^2/2)$ ile verilir. Şekil (3.1) 'deki benzerlik dolayısı ile Meksikalı şapkası da denir.

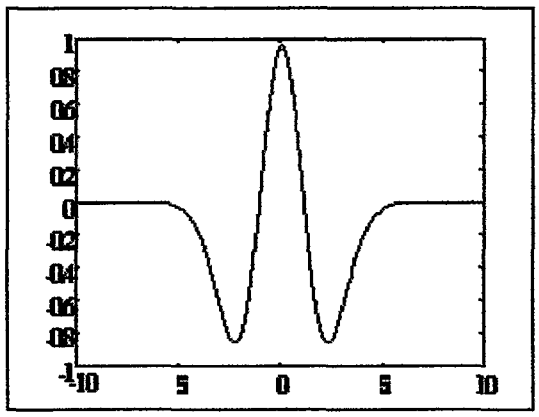
Bu fonksiyonun en önemli özelliği; a değişirken $\Psi^{a,0}(s) = |a|^{-1/2} \Psi(s/a)$ değişik frekansları kontrol eder. a 'nın büyük değerlerindeki Ψ küçük frekanslara, a 'nın küçük değerleri ise yüksek frekanslara tekabül eder. Değişen b değeri ise zaman-sınırlama olayının merkezini yani Meksikalı şapkasının merkezini belirler.



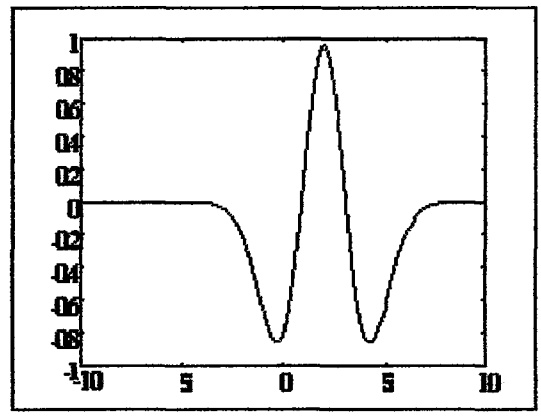
Şekil 3.1. Meksikalı Şapkası Fonksiyonu

Pencereli fourier dönüşümü ile wavelet dönüşümü arasındaki farklardan biri $g^{w,t}$ ve $\Psi^{a,b}$ 'nın şekilleri arasındaki farktır. g fonksiyonunda w değeri ne olursa olsun fonksiyonunun genişliği aynıdır. $\Psi^{a,b}$ yüksek frekanslarda dar, düşük frekanslarda geniştir. Sonuç olarak wavelet dönüşümü pencereli fourier dönüşümüne göre kısa süreli işaretlerde büyültme (zoom in) yapmada daha iyidir.

Şekil 3.2. ve şekil 3.3. 'de farklı frekans ve zaman merkezlerine karşılık gelen Meksikalı şapkası fonksiyonları görülmektedir. Görüldüğü üzere farklı a ve b değerleri için şekil zaman ekseni üzerinde kaymakta ve değişik frekans bölgelerine karşılık gelmektedir.



Şekil 3.2. " Meksikalı şapkası" a=1.1 ve b=0.1

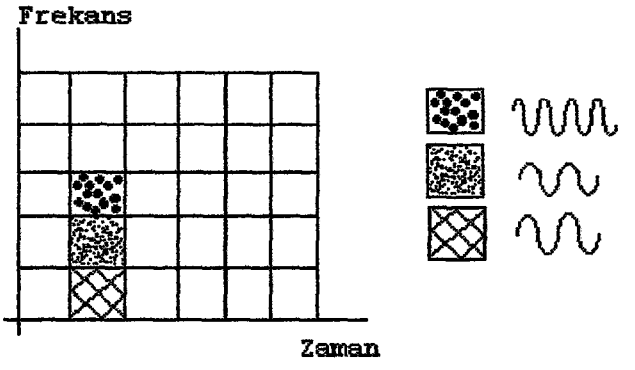


Şekil 3.3. " Meksikalı şapkası" a=1.1 ve b=2

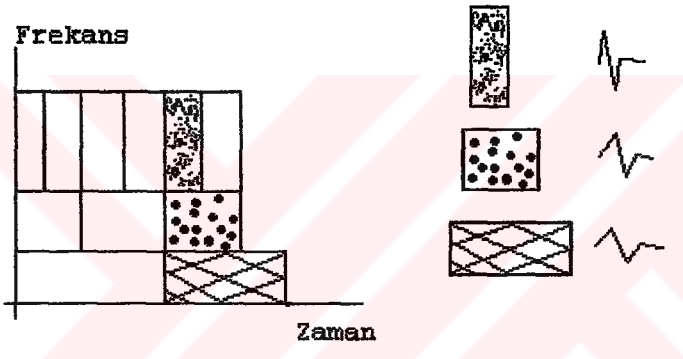
3.2.3. Fourier dönüşümü ile olan temel farklılıklar

En önemli özelliği uzayda yer belirlemedir(time-localization). Sinüs ve Kosinüs dönüşümünün böyle bir özelliği yoktur. Bu yer belirleme özelliği ile veride bir seyreklik oluşacak ve bu seyreklik ile veri sıkıştırma, görüntüde tarama yapma ve zaman serilerinden gürültü çıkartma mümkün olabilir. Pencereli fourier dönüşümü tüm frekans bölgeleri için aynı pencere genişliğini kullanır. Bu nedenle analizin çözünürlülüğü zaman-frekans domeninde her noktada aynıdır.

Wavelet dönüşümünün avantajı pencerenin değişebilir olmasıdır. Sinyaldeki devamsızlıklardan etkilenmemek için çok kısa temel fonksiyonlar kullanılmalıdır. Aynı zamanda ayrıntılı frekans analizi için çok uzun temel fonksiyonlar kullanılmalıdır. Bunu başarmak için yüksek frekansta kısa, düşük frekansta uzun temel fonksiyonlar kullanılmalıdır.



Şekil 3.4. Fourier temel fonksiyonları, zaman-frekans değişimi.



Şekil 3.5. Daubechies temel fonksiyonu, zaman-frekans değişimi.

Şekil 3.4. ve şekil 3.5. 'de sırasıyla fourier ve wavelet dönüşümlerinin değişik frekanstaki işaretler için nasıl davrandıkları görülmektedir(Graps, 1995).

3.3. Wavelet Dönüşümünün Değişik Tipleri

Wavelet dönüşümünün değişik tiplerinin hepsi denklem (3.3) ve (3.4) 'den çıkartılır.

a- Sürekli wavelet dönüşümü

b- Ayrık wavelet dönüşümü.

a1- Fazla bilgi içeren ayrık (çerçeveler) sistemler.

b2- Çok çözünürlülük analizi.

3.3.1. Sürekli wavelet dönüşümü

Burada kaydırma ve oran operatörleri süreklidir. Herhangi bir fonksiyon onun wavelet katsayılarından tekrar oluşturulabilinir:

$$f = C_{\Psi}^{-1} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{dadb}{a^2} < f, \Psi^{a,b} > \Psi^{a,b}$$

Burada $\Psi^{a,b}(x) = |a|^{-1/2} \Psi(\frac{x - b}{a})$ ile tanımlı olup <,> ibaresi L^2 iç çarpımları belirtir. İntegrallerin başındaki C_{Ψ} sabiti sadece Ψ 'e bağlı olup aşağıdaki ifade ile verilir:

$$C_{\Psi} = 2\pi \int_{-\infty}^{\infty} |\hat{\Psi}(\xi)|^2 |\xi|^{-1} d\xi$$

Ayrıca burada $C_{\Psi} < \infty$ olduğu farz edilecektir.

3.3.2. Ayrık wavelet dönüşümü

Bu dönüşüm türünde a ve b parametreleri şu şekilde $a = a_0^m$ ve $b = nb_0a_0^m$ ayrık değerler alır. Bu durumda ayrık wavelet dönüşümü;

$$\Psi_{m,n}(x) = a_0^{-m/2} \Psi(a_0^{-m}(x - nb_0a_0^m))$$
$$= a_0^{-m/2} \Psi(a_0^{-m}x - nb_0)$$

Ayrık Wavelet dönüşümü dönüşüm sonrası gereğinden fazla veri içerir. Çok çözünürlülük analizi ayrık dönüşümün

özel bir halidir. Eğer $a_0=2$ ve $b_0=1$ alınırsa iyi zaman-belirleme özelliğine sahip Ψ elde edilebilir. Ψ aşağıdaki ifade ile verilir.

$$\Psi_{m,n}(x) = 2^{-m/2}\Psi(2^{-m}x - n)$$

4. ALTBANT KODLAMA

$f(\theta) = \sum_{-\infty}^{\infty} c_k e^{ik\theta}$ ile verilmiş ve I, $[0, 2\pi]$ aralığını belirtmekle birlikte bu aralığın dışındaki tüm $f(\theta)$ değerleri sıfır kabul edilir. l_I^2'nın Hilbert alt uzaylarını gösterdiğini kabul edersek bu uzaylara ait $(c_k)_{k\in Z}$ 'nin alt dizileri olarak $(c_{mk})_{k\in Z}$ en uygun ve orijinalinin oldukça benzerini temsil eder(Ruskai, 1992).

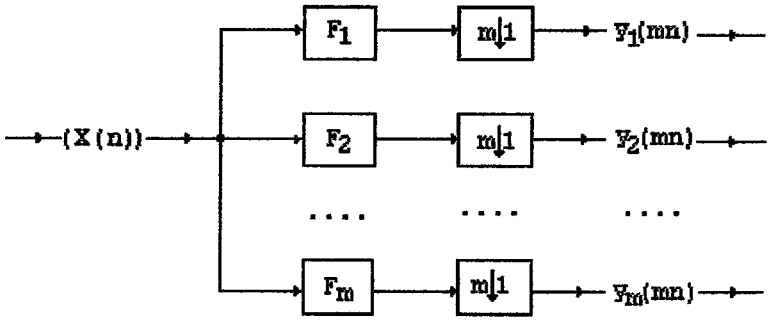
$$\frac{1}{m} f(\theta) = \frac{1}{m}\left(f(\theta) + f\left(\theta + \frac{2\pi}{m}\right) + \ldots + f\left(\theta + \frac{2\pi(m-1)}{m}\right)\right) = \sum c_{km} e^{im\theta}$$

Buradan şu bağıntıyı elde ederiz: $\sum_{-\infty}^{\infty} |c_{km}|^2 = \frac{1}{m} \sum_{-\infty}^{\infty} |c_k|^2$

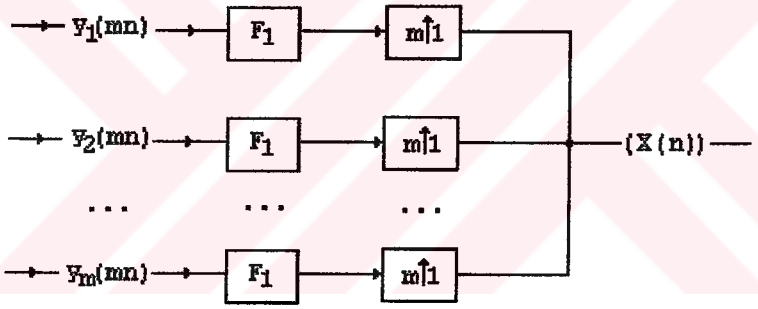
Bu da demektir ki orijinal dizi $f(\theta)$ 'yı oluşturmak için m tane sayısal veriye ihtiyaç duyulur.

İdeal altbant kodlama ilk önce gelen sinyali filtreleme ve daha sonra alt örnekleme ile sinyalin boyutunun küçültülmesi ile olur. Bu örnekleme işlemi gelen sinyal dizisinden belirli aralıklarla değer almaktır. Örnek alma işleminden sonra gelen sinyalin boyutu bir miktar düşecektir. Şekil 4.1. 'de bu kodlama işleminin nasıl olabileceği blok diyagramı şeklinde verilmiştir.

Şekil 4.2. 'deki sinyali tekrar oluşturma safhası ise ayrıştırma safhasının bir ikizidir. Önce gelen sinyal üst örnekleme ile boyutu artırılır. Bu artırma işleminde gelen sinyal dizisinin elemanları arasına sıfırlar eklemekle yapılır. Sonra ise daha önce kullanılan filtre sinyalinin kompleks eşleniği ile bu genişletilmiş sinyal çarpılarak orijinal sinyale ulaşılır.



Şekil 4.1. Ayrıştırma safhası

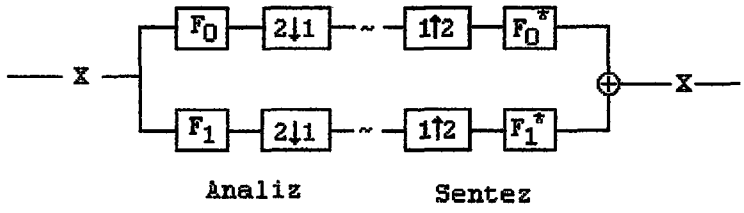


Şekil 4.2. Tekrar oluşturma safhası

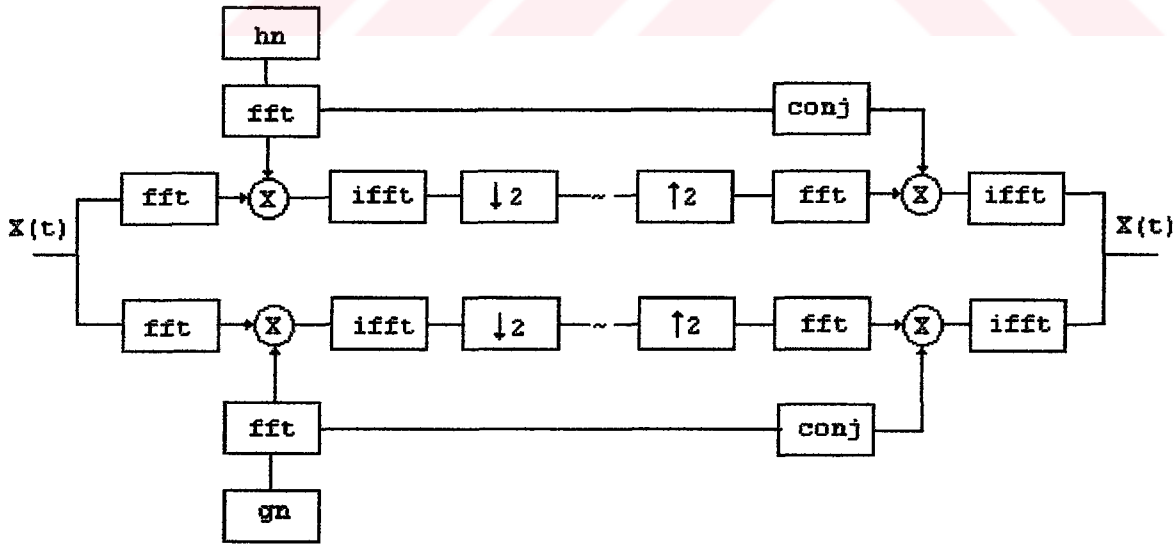
2π periyodik $F_0(\theta)$ fonksiyonunun fourier katsayılarının bir dizisi F_0 filtresinin impulse cevabıdır ve aynısı $F_0(\theta)$ ile F_0 için de geçerlidir.

Altbant kodlamada m=2 alınırsa en iyi sonuca ulaşılabilir. Bu durumda iki tane frekans kanalı olacağından bunlar F_0 ve F_1 olarak adlandırılacaktır.

Burada F_0 alçak geçiren filtre F_1 ise yüksek geçiren filtre olmak zorundadır. Tüm işlem analiz ve sentez olarak iki kısımda incelenebilir. Şekil 4.3. 'de blok diyagramı olarak bu işlem gösterilmiştir. Şekil 4.4. 'de ise hem sinyalin analiz edilmesi ve hem de sinyalin sentez edilmesi safhalarının matematiksel karşılığı verilmiştir.



Şekil 4.3. m=2 için analiz ve sentez safhaları



Şekil 4.4. Sistemin matematiksel modeli

5. ÇOK ÇÖZÜNÜRLÜK ANALİZİ

Çok çözünürlülük analizi Vj uzaylarının bir dizisinden oluşur. Ve bu uzaylar aşağıdaki özellikleri taşırlar(Meyer, 1992).

$\ldots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \ldots$

Bu uzaylar şu şartları da sağlamalıdır.

$$\overline{\bigcup_{J\in>Z} V_j} = L^2(\Re)$$

ve

$$\bigcap_{j\in Z} V_j = \{0\}$$

$$f \in V_j \Leftrightarrow f(2^j .) \in V_0$$

$$f \in V_0 \Rightarrow f(. - n) \in V_0 \forall n \in Z$$

$\psi_{j,k}$ $\{\psi_{j,k}; k\in Z\}$ W_j için ortonormal bir taban oluşturur. Bu fonksiyonun oluşturulması için aşağıdaki Φ ve W_0 için aşağıdaki özellikleri incelemek gerekir.

$\Phi\in V_0\subset V_{-1}$ ve $\Phi_{-1,n}$, V_{-1} 'de bir ortonormal bir taban oluşturduğundan dolayı şunu yazabiliriz.

$$\Phi = \sum_n h_n \Phi_{-1,n} \qquad (5.1)$$

$$h_n =< \Phi, \Phi_{-1,n} > \text{ ve } \sum_n |h_n|^2 = 1$$

Denklem (5.1) 'i ayrıca aşağıdaki gibi de yazabiliriz:

$$\Phi(x) = \sqrt{2}\sum_n h_n \Phi(2x - n)$$

ya da fourier dönüşümü alınmış hali olarak

$$\hat{\Phi}(\xi) = \frac{1}{\sqrt{2}} \sum_n h_n e^{-in\xi/2} \hat{\Phi}(\xi/2)$$

$$m_0(\xi/2) = \frac{1}{\sqrt{2}} \sum_n h_n e^{-in\xi/2}$$

Şeklinde yazılacak olursa m_0 denklemini denklem (5.2) 'deki gibi yazılabiliriz.

$$\hat{\Phi}(\xi) = m_0(\xi/2)\hat{\Phi}(\xi/2) \qquad (5.2)$$

Burada m_0 fonksiyonunun 2π periyotlu olduğu görülmektedir.

$$\delta_{k,0} = \int dx\Phi(x)\overline{\Phi(x - n)} = \int d\xi\left|\hat{\Phi}(\xi)\right|^2 e^{ik\xi}$$

$$= \int_0^{2\pi} d\xi e^{ik\xi}\sum_{l\in Z}\left|\hat{\Phi}(\xi + 2\pi l)\right|^2$$

$\zeta=\xi/2$ konarak ve Φ yerine de denklem (5.2)'deki değeri konursa

$$\sum_l \left|m_0(\zeta + \pi l)\right|^2\left|\hat{\Phi}(\zeta + \pi l)\right|^2 = 1$$

elde edilir. Burada toplam l 'nin tek ve çift olması durumuna göre ikiye ayrılabilir. Ve buradan da şu eşitliği çıkartabiliriz:

$$\left|m_0(\zeta)\right|^2 + \left|m_0(\zeta / 2)\right|^2 = 1$$

Wavelet fonksiyonu Ψ fonksiyonuna ise denklem (5.3) 'deki gibi karar verilir.

$$\Psi(x) = \sqrt{2}\sum_n (-1)^{n-1}h_{-n-1}\Phi(2x - n) \qquad (5.3)$$

5.1. Altband Filtreleme İle Birleşimi

Çok çözünürlülük analizi, verilen bir fonksiyonun wavelet katsayılarının hesaplanmasında oldukça hızlıdır. Denklem (5.3) 'den hareket ederek $\Psi = \sum_n g_n\Phi_{-1,n}$ yazılabilir. Burada $g_n = <\Psi,\Phi_{-1,n}>$ ile tanımlı olup bilindiği gibi $g_n = (-1)^n h_{1-n}$ verilmiş idi. Buradan gerekli düzenlemeleri yaparak denklem (5.4) 'deki eşitliği elde etmek mümkündür.

$$\Psi_{j,k} = \sum_n g_{n-2k}\Phi_{j-1,n}(x) \qquad (5.4)$$

$$\Psi_{j,k} = \sum_n g_{n-2k}\Phi_{j-1,n}(x) \qquad (5.4)$$

Buradan şu eşitlik çıkartılabilir:

$$< f, \Psi_{j,k} >= \sum_n \overline{g_{n,2k}} < f, \Phi_{0,n} >$$

Eğer bu ifade genelleştirilecek olursa;

$$< f, \Psi_{j,k} >= \sum_n \overline{g_{n,2k}} < f, \Phi_{j-1,n} > \qquad (5.5)$$

Buradan da görülmektedir ki $\bar{g}$ ile $< f, \Phi_{j-1,n} >$ 'nin konvolüsyonundan $< f, \Psi_{j,k} >$ hesaplanabilir. Aynı zamanda

$$< f, \Phi_{j,k} >= \sum_n \overline{h_{n,2k}} < f, \Phi_{j-1,n} > \qquad (5.6)$$

ile de $< f, \Phi_{j,k} >$ hesaplanabilir.

Buradaki işlem açıkça görülmektedir ki $< f, \Phi_{0,k} >$ ile başlanırsa denklem (5.5) yardımı ile $< f, \Psi_{j,k} >$ hesaplanır. Denklem (5.6) yardımı ile de $< f, \Phi_{j,k} >$ hesaplanır ve bu şekilde devam edilerek işlem sürdürülür. Biz her adımda sadece j. seviyeye tekabül eden $< f, \Psi_{j,k} >$ wavelet katsayılarını hesaplamıyoruz aynı zamanda j. seviyeden bir sonraki wavelet katsayılarının hesaplanmasında faydalı olan $< f, \Phi_{j,k} >$ fonksiyonlarını da hesaplıyoruz.

Tam bir işlem iki ardışık seviye arasındaki bilgi farklılığı ile birlikte f 'in daha kabaca yaklaşımlarının hesaplanması olarak da değerlendirilebilir. Bu görüşle f 'e daha iyi bir yaklaşımla, $f^0=P_0f$ ve $f^0 \in V_0 = V_1 \oplus W_1$ 'ı $f^0=f^1+\delta^1$ ayıracak olursak ki f^1 burada $P_1f^0=P^1f$ dır, bir sonraki çok çözünürlülük analizindeki f 'e daha kabaca bir yaklaşım olmuş olur ve $\delta^1 = f^0-f^1=Q_1f^0=Q1f$ 'de $f^0 \rightarrow f^1$ dönüşümündeki kayıptır. Bu V_j ve W_j uzaylarının her birinde

sırası ile $(\Phi_{j,k})_{k\in Z}$, $(\Psi_{j,k})_{k\in Z}$ ortonormal temellere sahibiz öyle ki denklem (5.7)'deki ifadeleri yazabiliriz.

$$f^0 = \sum_n C_n^0 \Phi_{1,n}\,, \quad f^1 = \sum_c C_n^1 \Phi_{1,n}\,, \quad \delta^1 = \sum_n d_n^1 \Psi_{1,n} \tag{5.7}$$

Denklem (5.7) 'deki bazı ifadeleri denklem (5.8) 'deki gibi yazabiliriz.

$$C_k^1 = \sum_n \overline{h_{n-2k}} C_n^0 \quad , \quad d_k^1 = \sum_n \overline{g_{n-2k}} C_n^0 \tag{5.8}$$

Denklem (5.8) 'deki gibi bir notasyon ile olaya yaklaşacak olursak $a = (a_n)_{n\in Z}$, $\bar{a} = (\overline{a_{-n}})_{n\in Z}$ ve $(Ab)_k = \sum_n a_{2k-n} b_n$ aşağıdaki ifadeleri yazabiliriz.

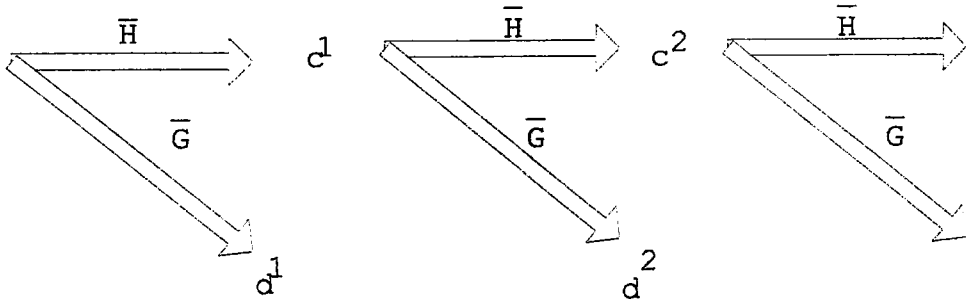
$C^1 = \bar{H}C^0$, $d^1 = \bar{G}C^0$

Kabaca bir yaklaşımla $f^1 \in V_1 = V_2 \oplus W_2$ olduğundan şu şekilde bir ayrıştırma yapmak mümkündür $f^1 = f^2 + \delta^2$, $f^2 \in V_2$, $\delta^2 \in W_2$. Böylece $f^2 = \sum_c C_n^2 \Phi_{2,n}\,, \quad \delta^2 = \sum_n d_n^2 \Psi_{2,n}$ elde edilir. Bu işlem yinelemeli bir şekilde devam edecek olursa şekil 5.1. 'deki şematik diyagramı çıkartabiliriz. Ters işlem ise adjoint matrisler ile verilir. Daha açık bir şekilde denklem (5.9) 'daki gibi verilir.

$$\begin{aligned} f^{j-1} &= f^j + \delta^j \\ &= \sum_k C_k^j \Phi_{j,k} + \sum_k d_k^j \Psi_{j,k} \end{aligned} \tag{5.9}$$

Denklem (5.9) 'a dayanarak aşağıdaki ifadeyi yazılabilir.

$$C_n^{j-1} = \langle f^{j-1}, \Phi_{j-1,n} \rangle = \sum_k C_k^j \langle \Phi_{j,k}, \Phi_{j-1,n} \rangle + \sum_k d_k^j \langle \Psi_{j,k}, \Phi_{j-1,n} \rangle$$
$$= \sum_k \left[h_{n-2k} C_k^j + g_{n-2k} d_k^j \right]$$



Şekil 5.1. Denklem (5.8)'in şematik gösterimi

Denklem (5.7) ve (5.9) elektrik mühendisliğinde sırası ile tam bir oluşum için alt bant filtreleme planının sentez ve analiz adımlarıdır. İki kanallı alt bant filtrelemede, gelen bir $(C_n^0)_{n\in Z}$ dizisi biri alçak geçiren filtre biri de yüksek geçiren filtre olmak üzere iki farklı filtre ile ayrıştırılır. Bu iki sonuç dizi ya tek girişler ya da çift girişler kalacak şekilde örneklenir. Şimdi filtre terminolojisini inceleyelim. Her karesel olarak toplanabilen $(C_n^0)_{n\in Z}$ dizisinin örneklenmiş değerleri $[-\pi,\pi]$ aralığında bant sınırlı olan $\gamma(n)$ fonksiyonu ile yorumlanabilir.

$$\gamma(x) = \sum_n C_n \frac{\sin \pi(x - n)}{\pi(x - n)}$$

ya da

$\hat{\gamma}(\xi) = \frac{1}{\sqrt{2\pi}} \sum_{n\in Z} C_n e^{-in\xi}$ olarak yazılabilinir.

Filtreleme işlemi $\hat{\gamma}$ ile 2π periyotlu bir fonksiyonun çarpımına tekabül eder.

$$\hat{\alpha}(\xi) = \sum_{n\in Z} a_n e^{-in\xi}$$

$\alpha * \gamma$ çarpımı başka bir bant sınırlı fonksiyona karşılık gelir.

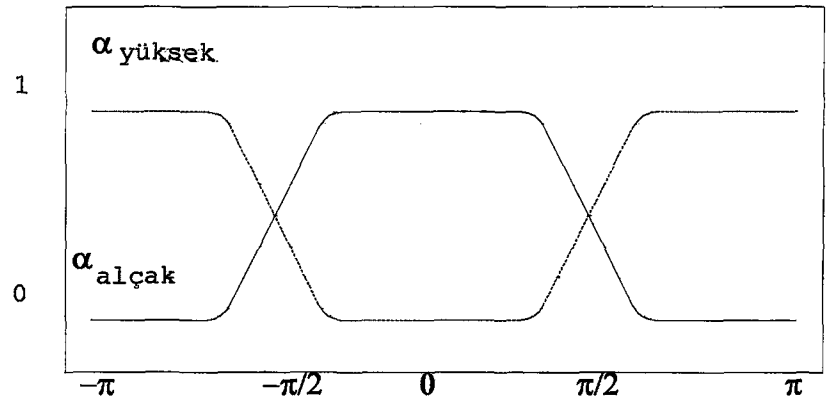
$$(\alpha * \gamma)(x) = \sum_n \left(\sum_m a_{a-m} C_m \right) \frac{\sin \pi(x - n)}{\pi(x - n)}$$

Eğer $\hat{\alpha}|_{[-\pi,\pi]}$, $[-\pi/2, \pi/2]$ aralığında yoğun ise filtre alçak geçiren, $\hat{\alpha}|_{[-\pi,\pi]}\{\xi \; ; \; \pi/2 \le |\xi| \le \pi\}$ aralığında toplanmış ise yüksek geçiren filtredir. İdeal alçak geçiren ve yüksek geçiren filtre $\hat{\alpha}_L(\xi) = 1$ *eğer* $|\xi| < \pi/2$, 0 eğer $\pi/2 < |\xi| < \pi$ ve $\hat{\alpha}_L(\xi) = 0$ eğer $|\xi| < \pi/2$, 1 eğer $\pi/2 < |\xi| < \pi$.

Denklem (5.8) 'e tekabül eden a_n katsayıları:

$$a_n^L = \begin{cases} \frac{1}{2} & n = 0 \\ 0 & n = 2k, \; k \neq 0 \\ \frac{(-1)^k}{(2k+1)\pi} & n = 2k + 1 \end{cases}$$

$$a_n^H = \begin{cases} \frac{1}{2} & n = 0 \\ 0 & n = 2k, \; k \neq 0 \\ \frac{(-1)^{k+1}}{(2k+1)\pi} & n = 2k + 1 \end{cases}$$



Şekil 5.2. Alçak geçiren(kesik çizgili) ve yüksek geçiren filtre karakteristikleri

Şekil 5.2. 'de ideal alçak ve yüksek geçiren filtre karakteristikleri verilmiştir. Eğer ideal alçak geçiren filtre karakteristiği örneklenmiş değerlere uygulanacak olursa sonucun tanım aralığı $\left[-\pi/2, \pi/2\right]$ kapalı aralığıdır. Böyle bir fonksiyona onun 2Z domeninde örneklenmiş değerleri ile karar verilir.

$$(\alpha_L * \gamma)(x) = \sum \left(\sum a^L_{2n-m} C_M \right) \frac{\sin\left[\pi(x - 2n)/2\right]}{\pi(x - 2n)/2}$$

Benzer şekilde yüksek geçiren filtreyi γ 'e uygulanmasının sonucu, $\left[-\pi/2, \pi/2\right]$ tanım aralığında bant sınırlı bir fonksiyonun frekans kaydırmalı versiyonudur.

$$(\alpha_H * \gamma)(x) = \frac{1}{\pi} \int_{\frac{\pi}{2} \le |\xi| \le \pi} d\xi e^{ix\xi} \sum_n \left(\sum_m a^L_{2n-m} C_M \right) e^{-2in\xi}$$

$$= \sum_n \left(\sum_m a^H_{2n-m} C_M \right) \frac{\sin[\pi(x - 2n)/2]}{\pi(x - 2n)/2} \{2\cos[\pi(x - 2n)/2] - 1\}$$

a^L_n, a^H_n'nın çift indeksli terimlerinin C_k ile olan konvolüsyonlarını $\alpha_L * \gamma$ ve $\alpha_H * \gamma$ tam anlamı ile karakterize

etmek için yeterlidir ve konvolüsyondan sonra onları yitirmemek için daha duyarlı yapar. Bu alt örnekleme (downsampling) denen işlemdir ve iki faktörü ile alt bant filtrelemede ondalık sayıya çevirme temeline dayanır. Orijinal C_m 'i iki filtrelenmiş ve desime edilmiş diziden tekrar oluşturabiliriz. Şekil 5.3. 'de genel olarak sistemin yapısı verilmiştir(Ruskai, 1992).

$$C_n^L = \sum_m a_{2n-m}^L C_m \,, \quad C_n^H = \sum_m a_{2n-m}^H C_m$$

$$C_m = \gamma(m) = (\alpha_L * \gamma)(m) + (\alpha_H * \gamma)(m)$$

Bu denklemi tek ve çift kısımlarına ayıracak olursak

$$C_{2m} = C_m^L + C_m^H$$

$$C_{2m+1} = \sum_l \frac{2(-1)^l}{\pi(2l+1)} (C_{m-l}^L - C_{m-l}^H)$$

Bunu şu şekilde de yazabiliriz.

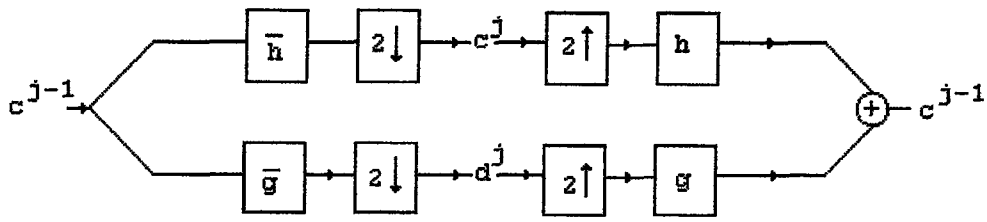
$$C_m = 2\sum_n (a_{m-2n}^l C_n^l + a_{m-2n}^H C_n^H)$$

Bu son işlem sonucunda aşağıdaki sonuçlara varabiliriz.

1- Hem C_n^L hem de C_n^H 'e sıfırlar ilave etmek.

2- Bu aralara sıfırlar ilave edilmiş (upsampled) diziler ile a^L ve a^H filtrelerini konvüle etme.

3- Bu iki sonucu toplama.



Şekil 5.3. Çok çözünürlülük analizinde alt bant filtreleme planının uygulanması. İlk konum ayrıştırma ikinci konum tekrar oluşturma içindir.

5.2. Haar Çözünürlülüğü Üzerine Bir Örnek

Bilindiği üzere Haar wavelet 'ı şu şekilde tanımlanır:

$$\Phi(x) = \begin{cases} 1 \text{ e ğ er } 0 \le x \le 1 \\ 0 \text{ di ğ er durumlar} \end{cases}$$

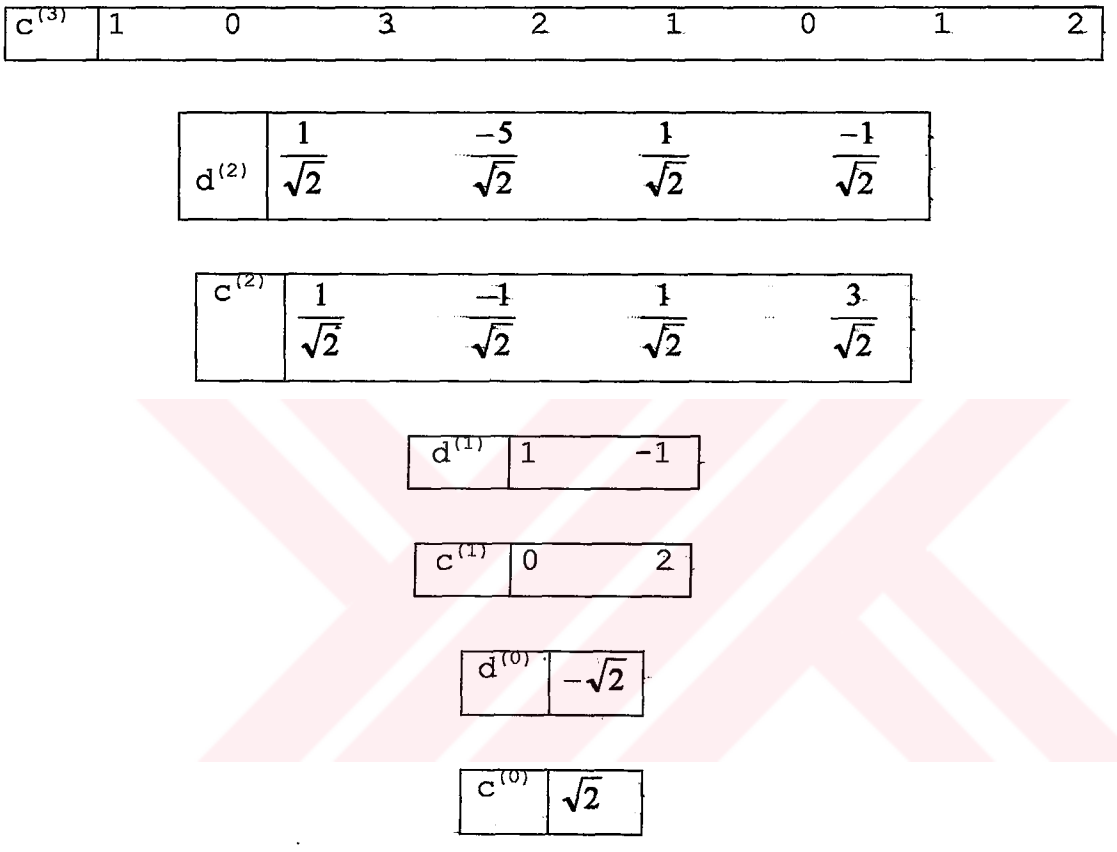
$$h_n = \sqrt{2}\int dx\Phi(x)\overline{\Phi(2x - n)} = \begin{cases} 1/\sqrt{2} n = 0,1 \\ 0 \text{ di ğ er durumlarda} \end{cases}$$

Ana wavelet olarak adlandırılan Ψ fonksiyonunu denklem (5.3) ile bulacak olursak

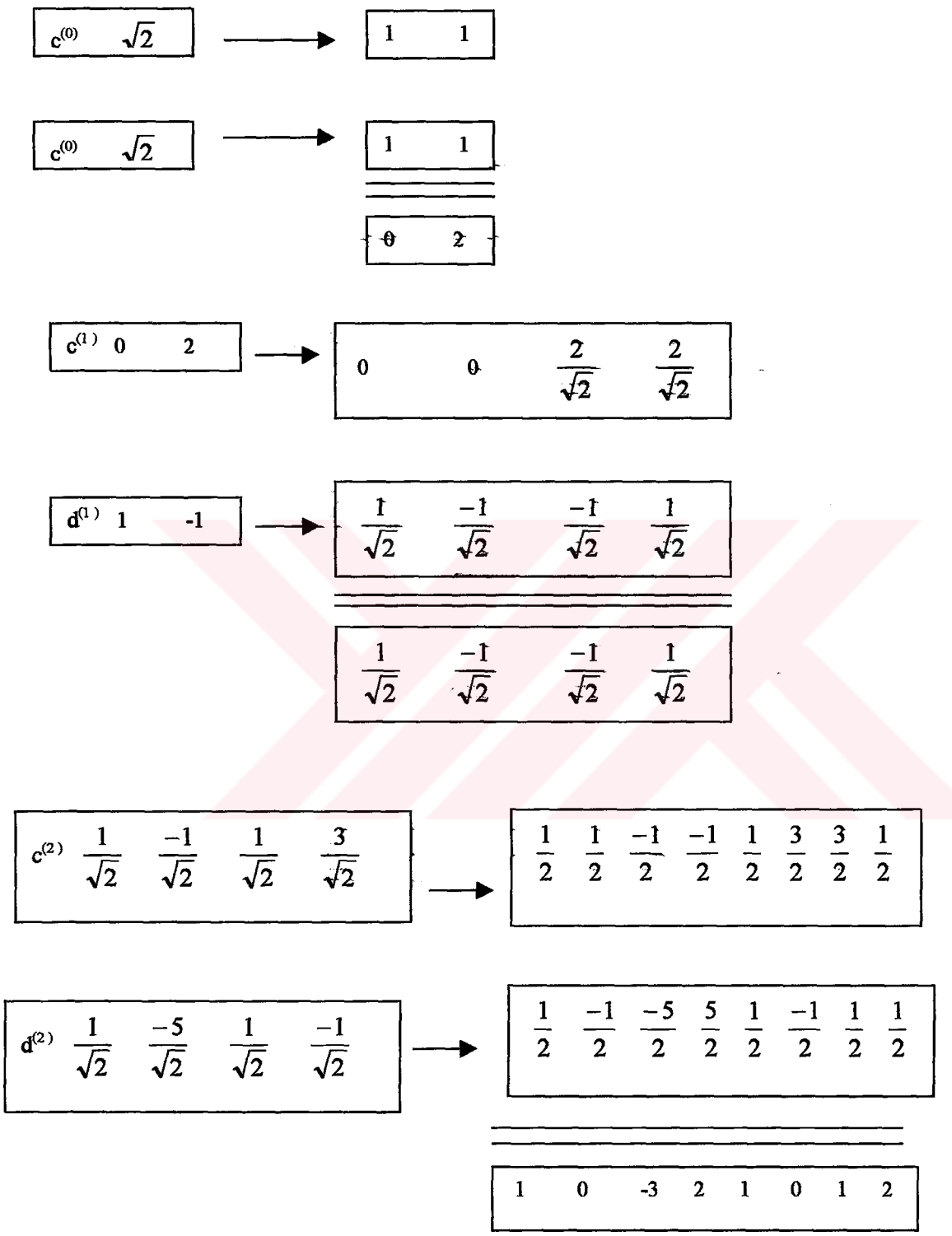
$$\Psi(x) = \begin{cases} 0 \le x \le 1/2 \\ -1/2 \le x \le 1 \\ 0 \text{ di ğ er durumlar} \end{cases}$$

Elimizde {1 0 -3 2 1 0 1 2} şeklinde bir veri dizi olduğunu varsayalım. Burada n=8 olduğuna göre $2^j=2^3$ olduğu görülmektedir. Dolayısıyla $c^{(3)}$=(1 0 -3 2 1 0 1 2) 'dir. Haar filtre katsayıları $h(0) = h(1) = \frac{1}{\sqrt{2}}$ ve $g(0) = -g(1) = \frac{1}{\sqrt{2}}$ olduğuna göre öz yinelemeli olarak hesaplanması gereken katsayıları ($d^{(j-1)}$, $d^{(j-2)}$, ...$d^{(1)}$, $d^{(0)}$, $c^{(0)}$) 'dir. Ayrıştırma işlemi şekil 5.4. 'deki gibidir. Elimizdeki dönüştürülmüş veri ($d^{(2)}$, $d^{(1)}$, $d^{(0)}$, $c^{(0)}$)=($\frac{1}{\sqrt{2}}$, $\frac{-5}{\sqrt{2}}$, $\frac{1}{\sqrt{2}}$, $\frac{-1}{\sqrt{2}}$, 1, -1, $-\sqrt{2}$, $\sqrt{2}$) şeklinde bir vektördür. Eğer 0.9 'dan küçük değerler sıfıra çekilerek bir eşik işlemi uygulanırsa vektör (0, $\frac{-5}{\sqrt{2}}$, 0, 0, 1, 1, $-\sqrt{2}$, $\sqrt{2}$) halini alır.

Orijinal veriye tekrar ulaşmak için şekil 5.5 'deki işlemleri takip etmek gerekecektir.



Şekil 5.4. Haar çözünürlülüğünde ayrıştırma safhası



Şekil 5.5. Haar çözünürlülüğünde sentez safhası

5.3. Görüntü Sıkıştırmada Çok Çözünürlülük Analizi

Tanımlamalar:

Φ çok çözünürlüklü wavelet temelinin üreteci olup $\Phi_{mn}(x) = 2^{-m/2}\Phi(2^{-m}x - n)$ ile verilmiş olsun.

V_m:=aralık{Φ_{mn}| n∈Z} uzayları değişik ayrıştırma uzaylarına tekabül eden uzaylar olsun. $\Psi_{mn}(x) = 2^{-m/2}\Psi(2^{-m}x - n)$ ile belirtilen ve W_m:=aralık{Ψ_{mn}| n∈Z} uzayında tanımlanan bir fonksiyon daha tanımlanabilir.

V_m ve W_m uzayları arasındaki ilişki şu şekilde verilir:

$V_{m-1}=V_m \oplus W_m$

P_m ve Q_m sırası ile V_m ve W_m üzerinde verilmiş olan ortogonal projektörler olsun.

Bu tanımlamalardan sonra sıkıştırmak istediğimiz sinyali belirtelim. $(C_n)_{n\in Z} \in L^2Z$ elimizdeki sinyal olsun. Ve $C_n^o = C_n$ dizisini belirtsin. Bu diziyi $f\in V_0$ gibi bir dizi ile birleştirecek olursak $f(x) = \sum_n c_n^0 \Phi_{0n}(x)$ elde edilir.

Şimdi çok çözünürlük analizini bu f fonksiyonuna uygulayalım. Burada f 'i şu şekilde belirtebiliriz:

$f=P_1f+Q_1f$

Burada P_1, V_1 uzayındaki düşük çözünürlülükleri, Q1 ise W1 uzayındaki fark sinyalini tanımlar. Ve P_1f aşağıdaki gibi tanımlanır:

$$P_1f = \sum_k C_k^1 \Phi_{1k}$$

O halde çok çözünürlülük analizine göre C_k^1 için aşağıdaki bağıntıları yazabiliriz:

$$C_k^1 =< P_1f, \Phi_{1k} >=< f, \Phi_{1k} >= \sum_n C_n^0 < \Phi_{0n}, \Phi_{1k} >= \sum C_n^0 h_{n-2k}$$

$$h_n = 2^{-1/2}\int \Phi\left(\frac{x}{2}\right)\Phi(x - n)dx$$

Ayrıca fark sinyali içinde şunu yazabiliriz:

$$Q_1 f = \sum_n d_k^1 \Psi_{1k}$$

O halde çok çözünürlülük analizine göre d_k^1 fark sinyali için aşağıdaki bağıntıları yazabiliriz:

$$d_k^1 = < Q_1 f, \Psi_{1k} > = \sum_n C_n^0 < \Phi_{0n}, \Psi_{1k} > = \sum C_n^0 g_{n-2k}$$

$$g_n = 2^{-1/2} \int \Psi\left(\frac{x}{2}\right) \Phi(x - n) dx$$

yukarıdaki bu ifadeye göre gn 'i şu şekilde de ifade edebiliriz.

$$g_n = (-1)^n \overline{h}_{1-n}$$

Eğer Φ 'yi reel değerli olarak seçecek olursak

$g_n = (-1)^n h_{1-n}$ olur. Burada hem g_n hem de h_n reel değerlidir.

Buraya kadar yapılan işlemler N defa daha tekrar edilirse f için şu ifadenin yazılabileceği açıktır:

$f = Q_1 f + Q_2 f + \ldots\ldots\ldots\ldots\ldots + Q_N f + P_N f$

Burada $P_N f$ ve $Q_N f$ 'in genel ifadeleri şu şekildedir.

$$P_N f = \sum_k C_k^n \Phi_{nk}$$

$$Q_N f = \sum_k d_k^n \Psi_{nk}$$

C_k^n ve d_k^n katsayılarını öz yinelemeli formada yazacak olursak;

$$C^j = HC^{j-1}$$

$$d^j = GC^{j-1}$$

Buradaki H ve G değerleri için aşağıdaki ifade verilebilir;

$$(H_a)_k = \sum_n h_{n-2k} a_n$$

$$(G_a)_k = \sum_n g_{n-2k} a_n$$

Bir kaç iterasyondan sonra orijinal dizi C^0, en düşük çözünürlülük sinyali C^N ve d^N, $d^{N-1}, \ldots d^1$

fark sinyallerine ayrışır. Orijinal dizi aşağıdaki bağıntının gerçekleştirilmesi ile tekrar elde edilir.

$$P_{j-1}f = P_j f + Q_{1f} = \sum C_k^j \Phi_{jk} + \sum d_k^j \Psi_{jk}$$

Burada C_n^{j-1} 'i aşağıdaki gibi yazabiliriz.

$$C_n^{j-1} = < P_{j-1}, \Phi_{j-1,n} >$$

$$= \sum_k C_k^j < Q_{jk}, \Phi_{j-1,n} > + \sum_k d_k^j < \Psi_{jk}, \Phi_{j-1,n} >$$

Tekrar oluşturma algoritması aşağıdaki formda verilir.

$$C^{j-1} = H * C^j + G * d^j$$

H ve G reel değerli olduklarından dolayı eşlenikleri de gerçel formda olup sonuç olarak orijinal C^0 dizisi aşağıdaki şekilde tekrar oluşturularak elde edilebilir(Meyer, 1992).

$$C^0 = \sum_{j=0}^{N} (H^*)^{j-1} G * d^j + (H^*)^N C^N$$

6. DAUBECHIES' WAVELETLARI

Daubechies wavelet'ı temel olarak teorem (6.1) 'i kullanarak sonuca gider.

Teorem 6.1:(daubechies') h_n aşağıdaki gibi özelliklere sahip bir dizi olsun(Daubechies, 1992).

(i) $\sum_n |h_n||n|^\varepsilon < \infty$ bazı $\varepsilon>0$ için

(ii) $\sum_n h_{n-2k}h_{n-2l} = \delta_{kl}$

(iii) $\sum_n h_n = \sqrt{2}$

Farz edelim ki $H(\xi) = 2^{\frac{-1}{2}}\sum_n h_n e^{-in\xi}$ şu şekilde yazılabilir:

$$H(\xi) = \left[\frac{1}{2}(1 + e^{-i\xi})\right]^N \left[\sum_n f_n e^{-in\xi}\right]$$

burada

(iv) $\sum_n |f_n||n|^\varepsilon < \infty$ bazı $\varepsilon>0$ için

(v) $\sum_n |f_n||n|^\varepsilon < 2^{N-1}$

tanım aralığı olup şu tanımlamaları yapabiliriz:

$g_n = (-1)^n h_{1-n}$

$$\hat{\Phi}(\xi) = \prod_{j=1}^{\infty} H(2^{-j}\xi)$$

$$\Psi(x) = \sqrt{2}\sum_n g_n \Phi(2x - n)$$

Teorem 6.2:(Daubechies) Belirli bir aralığın dışında zayıflayan waveletların oluşturulmasına H(ξ) 'un yazılması ile başlanır(Daubechies, 1992).

$$H(\xi) = \left[\frac{1}{2}(1 + e^{-i\xi})\right]^N Q(e^{-i\xi}) \qquad (6.1)$$

Burada Q reel katsayılı bir fonksiyondur. Teorem 6.1 'den aşağıdaki eşitliği çıkartmak mümkündür.

$$|H(\xi)|^2 + |H(\xi + \pi)|^2 = 1$$

Denklem (6.10) kullanılıp her iki yanın karesi alınarak aşağıdaki eşitliğe ulaşılabilir.

$$|H(\xi)|^2 = \left|\cos^2 \frac{1}{2}\xi\right|^N |Q(e^{-i\xi})|^2 \qquad (6.2)$$

elde edilir. Q(x) 'in katsayılarının gerçel değerli olmasından dolayı $\overline{Q(e^{-i\xi}} = Q(e^{-i\xi})$ yazılabilir ve gösterilebilir ki cosξ 'nin veya $\sin^2(1/2\xi)$ 'nin bir polinomu olarak yazılabilir. Bu durumda $P_N(y)$ şu şekilde ifade edilir.

$$P(\sin^2 \frac{1}{2}\xi) = |Q(e^{-i\xi})|^2$$

Burada

$$P(y) \geq 0 \quad \forall \in y[0,1]$$

Yukarıdaki polinom şu hale gelir.

$$y^N P(1 - y) + (1 - y)^N P(y) = 1$$

Eğer yukarıdaki eşitlik çözülecek olursa ve denklem (6.2) 'de yerine konursa

$$\left|Q(e^{-i\xi})\right|^2 = \sum_{j=0}^{N-1}\binom{N-1+j}{j}\sin^{2j}(\frac{1}{2}\xi) + \left[\sin^{2N}(\frac{1}{2}\xi)\right]R(\frac{1}{2}\cos\xi) \qquad (6.3)$$

elde edilir.

Daubechies'e göre(1992), Daubechies wavelet 'ını elde etmek için R=0 alınmalıdır. O halde denklem (6.3) şu hale gelir.

$$\left|Q(e^{-i\xi})\right|^2 = \sum_{j=0}^{N-1}\binom{N-1+j}{j}\sin^{2j}(\frac{1}{2}\xi)$$

Daha sonra bu Q^2 ifadesinden Q ifadesine karar verilir. Karar verme Q 'nun birim çember içindeki sıfırlarının bulunması ile gerçekleştirilir.

$$Q_N(e^{-i\xi}) = \sum_{n=0}^{N-1} q_n e^{-in\xi} \quad q_0 \neq \qquad (6.4)$$

Denklem (6.4) 'de belirtilen Q_N polinomunun sıfırlarına karar verilir. Buna tekabül eden H_0 ki onu ${}_NH_0$ ile göstereceğiz.

$${}_NH_0 = \left[\frac{1}{2}(1+e^{i\xi})\right]^N \sum_{n=0}^{N-1} q_N(n)e^{in\xi}$$

$$= 2^{-1/2}\sum_{n=0}^{2N-1} h_N(n)e^{in\xi}$$

N 'in küçük değerleri için h_n katsayıları analitik olarak hesaplanabilir. Ama büyük değerli N 'ler için analitik hesap zor olup ancak programlama ya da tablolardan faydalanarak bulunabilir. Tablo 6.1. 'de N=2,3,4,5 için bu katsayılar verilmiştir.

Tablo 6.1. N=2,3,4,5 için daubechies filtre katsayıları

FİLTRE DERECESİ	FİLTRE KAYSAYISI	FİLTRE DERECESİ	FİLTRE KAYSAYISI
N=2	0.482	N=4	0.230
	0.836		0.614
	0.224		0.630
	-.129		-.027
N=3	0.332		-.186
	0.806		0.030
	0.459		0.032
	-.135		-.001
	-.085	N=6	0.111
	0.035		0.494
N=5	0.160		0.651
	0.603		0.315
	0.624		-.226
	0.138		-.129
	-.242		0.096
	-.032		0.026
	0.066		-.031
	-.006		0.0005
	-.012		0.004
	0.003		-.001

6.1. Daubechies Waveletları İle Hesaplama

Bu bölümde herhangi bir fonksiyonun ayrık wavelet dönüşümünün nasıl hesaplanabileceği gösterilecektir.

Ayrık wavelet 'ın teorisinde aşağıdaki denklem oldukça önemli bir rol oynar ve baba wavelet olarak adlandırılır:

$$\Phi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} h_n \Phi(2x - n)$$

$$\Phi_{jk}(x) = 2^{-j/2} \Phi_N(2^{-j}x - k)$$

Bu fonksiyondan yola çıkılarak ana wavelet olarak adlandırılan ψ fonksiyonu elde edilebilir:

$$\Psi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} (-1)^n h_{1-n} \Phi(2x - n)$$

Burada daha öncede bilindiği üzere $(-1)^n h_{1-n}=g_n$ olarak tanımlıdır. $\bar{a} = (a_0, \ldots a_{M-1})^T$ gibi sonlu bir sinyal ile tanımlı bir fonksiyon olduğunu düşünelim. Ara bir ifade olarak $A = \sum_{j=0}^{M-1} a_j \Phi_{Kj}$ denklemini yazalım. Burada $M=2^K$ 'dır. O zaman a 'nın AWD 'si 2^K-2 tane katsayı içerir. N 'i sabit seçip Φ_N 'in filtre katsayılarına h_n diyecek olursak H_N ve G_N filtreleri şu şekilde tanımlanır.

$$(H_N \bar{a})_k = \sum_{l=-\infty}^{\infty} h_{l-2k} a_l$$

$$(G_N \bar{a})_k = \sum_{l=-\infty}^{\infty} g_{l-2k} a_l$$

Bu filtrelerle ve çift bir M ile $\bar{a}$ sonlu sinyali üzerinde çalışalım. H_N 'in matris formu aşağıdaki gibidir.

$$H_N = \begin{bmatrix} h_0 & h_1 & h_2 & & \cdots & h_{2N-1} & & \\ & h_0 & h_1 & & \cdots & h_{2N-1} & & \\ & & & \ddots & & & & \\ & & & & \ddots & & & \\ & & & & & & h_0 & h_1 \end{bmatrix}$$

G_N matrisi de H_N matrisi ile aynı olup sadece h_n yerine g_n konulması ile elde edilir. N>1 için bu 1/2(MxM) 'lik matrislerle filtreleme kenar etkilerine sebep olacaktır. Bu nedenle bu kenar etkilerini elimine etmek için matris aşağıdaki gibi periyodik bir hale getirilmelidir.

$$H_N = \begin{bmatrix} h_0 & h_1 & h_2 & & \cdots & h_{2N-1} & & \\ & \ddots & & & & & & \\ & & \ddots & & & & \ddots & \\ & & & \ddots & & & & h_{2N-1} \\ & & & & h_0 & & & \\ & & & & & h_0 & & \\ h_{2N-2} & h_{2N-1} & & & & & & h_{2N-3} \\ \vdots & & \ddots & & & & & \\ h_2 & & & h_{2N-1} & & & h_0 & h_1 \end{bmatrix}$$

Burada da G_N ile H_N aynı olup sadece h_n yerine g_n konulmalıdır. Bu filtreler $\bar{a}$ sonlu sinyaline uygulanırsa;

$$\begin{bmatrix} H_N \\ G_N \end{bmatrix}_M \begin{pmatrix} a_0 \\ \vdots \\ a_{M-1} \end{pmatrix} = \begin{pmatrix} H_N \begin{pmatrix} a_0 \\ \vdots \\ a_{M-1} \end{pmatrix} \\ G_N \begin{pmatrix} a_0 \\ \vdots \\ a_{M-1} \end{pmatrix} \end{pmatrix} : \mathfrak{R}^M \rightarrow \mathfrak{R}^M$$

$\begin{bmatrix} H_N \\ G_N \end{bmatrix}_M$ matrisi ortonormaldir.

$$\begin{bmatrix} H_N \\ G_N \end{bmatrix}_M \begin{bmatrix} H_N^T G_N^T \end{bmatrix}_M = \begin{bmatrix} H_N H_N^T & H_N G_N^T \\ G_N H_N^T & G_N G_N^T \end{bmatrix}_M = Id_M$$

Bu (6.5) ve (6.6) denklemlerden direkt olarak görülebilir(Strang, 1992).

$$\sum_{n=-\infty}^{\infty} h_{n-2k} h_{n-2l} = \delta_{kl} \qquad (6.5)$$

$$\sum_{n=-\infty}^{\infty} h_{n-2k} g_{n-2l} = 0 \qquad (6.6)$$

Burada denklem (6.5) teorem 6.1 'in (ii) şartını göstermektedir ve denklem (6.6) ise şu şekilde gösterilebilir:

$$\sum_{n=-\infty}^{\infty} h_{n-2k} g_{n-2l} = \sum_{n=-\infty}^{\infty} < \Phi_{1k}, \Phi_{0n} > < \Phi_{0n}, \Psi_{1l} > = < \Phi_{1k}, \Psi_{1l} > = 0$$

Buradan şu tanımlamaları yapabiliriz:

$$\Phi_{mn}(x) = 2^{\frac{-m}{2}} \Phi(2^{-m}x - n) \quad , \qquad \Psi_{mn}(x) = 2^{\frac{-m}{2}} \Phi(2^{-m}x - n)$$

ve $M=2^K$ ve $k \in N$ seçeriz. $A \in V_K$ olsun:

$$a_k = a_{k0} = < A, \Phi_{Kk} >$$

O zaman şu tanımlamaları da yapabiliriz:

$$a_{kj} = < A, \Phi_{K-k,j} > \qquad d_{kj} = < A, \Psi_{K-k,j} >$$

Buraya kadar söylenenler ile ayrık wavelet dönüşümünü örneğin K=4 için inceleyelim.

$$\begin{pmatrix} a_{0,0} \\ a_{1,0} \\ a_{2,0} \\ a_{3,0} \\ a_{4,0} \\ a_{5,0} \\ a_{6,0} \\ a_{7,0} \\ a_{8,0} \\ a_{9,0} \\ a_{10,0} \\ a_{11,0} \\ a_{12,0} \\ a_{13,0} \\ a_{14,0} \\ a_{15,0} \end{pmatrix} \xrightarrow{\begin{bmatrix} H_N \\ G_N \end{bmatrix}_{16}} \begin{pmatrix} a_{0,1} \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ a_{4,1} \\ a_{5,1} \\ a_{6,1} \\ a_{7,1} \\ d_{0,10} \\ d_{1,1} \\ d_{2,1} \\ d_{3,1} \\ d_{4,1} \\ d_{5,1} \\ d_{6,1} \\ d_{7,1} \end{pmatrix} \xrightarrow{\begin{bmatrix} H_N \\ G_N \end{bmatrix}_{8}} \begin{pmatrix} a_{0,2} \\ a_{1,2} \\ a_{2,2} \\ a_{3,2} \\ d_{0,2} \\ d_{1,2} \\ d_{2,2} \\ d_{3,2} \\ d_{0,1} \\ d_{1,1} \\ d_{2,1} \\ d_{3,1} \\ d_{4,1} \\ d_{5,1} \\ d_{6,1} \\ d_{7,1} \end{pmatrix} \xrightarrow{\begin{bmatrix} H_N \\ G_N \end{bmatrix}_{4}} \begin{pmatrix} a_{0,3} \\ a_{1,3} \\ d_{0,3} \\ a_{1,3} \\ d_{0,2} \\ d_{1,2} \\ d_{2,2} \\ d_{3,2} \\ d_{0,1} \\ d_{1,1} \\ d_{2,1} \\ d_{3,1} \\ d_{4,1} \\ d_{5,1} \\ d_{6,1} \\ d_{7,1} \end{pmatrix}$$

Burada $a_{0,K-1}$ ve $a_{1,K-1}$ katsayıları "mother fonksiyon katsayıları" olarak adlandırılır. Burada N 'in büyük değerlerinde Φ ve Ψ 'nin pürüzsüzlüğü artar.

6.2. Daubechies Waveletlarının Görüntü Sıkıştırmada Kullanılması

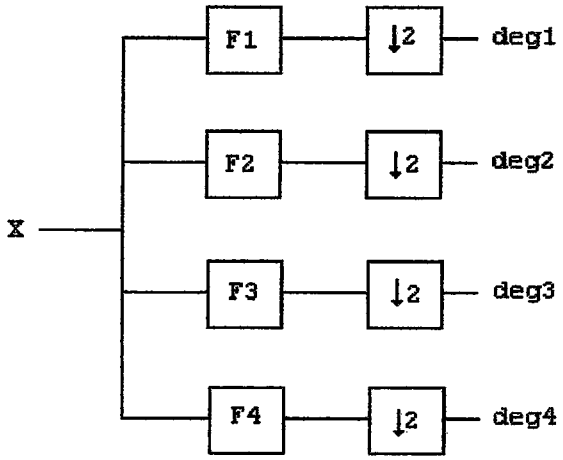
X, görüntünün matrissel gösterimi olup bir kare matris olmalıdır. Deg1 asıl matris olmak üzere deg2, deg3 ve deg4 en iyi oluşumu(perfect reconsruction) sağlamak için hesaplanmalıdırlar.

Bu değişkenler hesaplanırken aşağıdaki filtreler kullanılır.

F1=hn.hn {satır.sutun}
F2=hn.gn {satır.sutun}
F3=gn.hn {satır.sutun}
F4=gn.gn {satır.sutun}

Burada hn alçak geçiren, gn yüksek geçiren filtre katsayıları olup hn ile gn arasında $hn=(-1)^n$ gn şeklinde bir ilişki vardır.

Filtreler hesaplanırken Daubechies filtre katsayıları kullanılır. Şekil 6.1. 'de Daubechies filtreleri kullanılarak bir görüntünün ayrıştırılması görülmektedir.



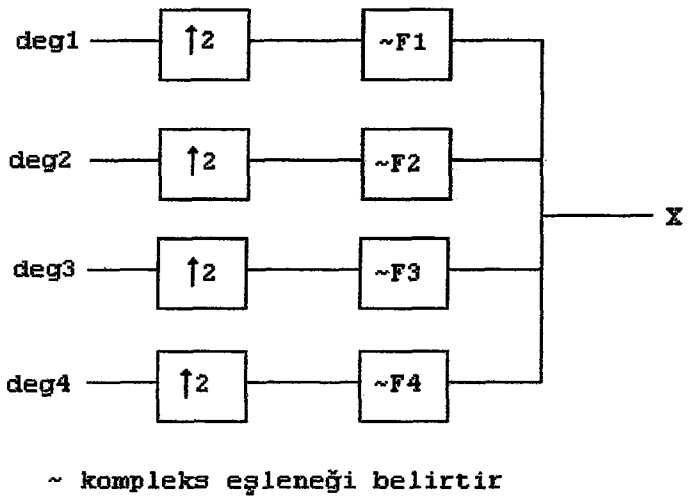
Şekil 6.1. Daubechies filtreleri ile hesaplamada sentez safhası

Dnsmp2 iki boyulu alt örnekleme fonksiyonu olup örnekleme oranı 2 'dir.

Görüntüyü tekrar elde etmek için daha önce uygulanan adımların tekrarlanması yeterli olacaktır. Eğer nicelikleştirme işlemi kullanılmaz ise kayıpsız bir sıkıştırma sağlayacaktır. Ama burada sağlanan sıkıştırma oranı çok çözünürlülük analizinden oldukça düşüktür.

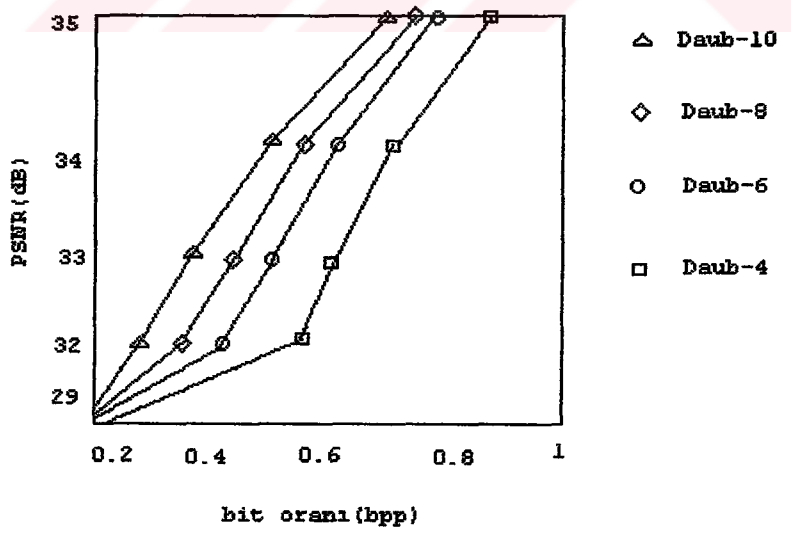
F1=hn.hn {satır.sutun}
F2=hn.gn {satır.sutun}
F3=gn.hn {satır.sutun}
F4=gn.gn {satır.sutun}

Blok diyagramı olarak sistemi tanımlayacak olursak şekil 6.2. 'deki gibi olur.



Şekil 6.2. Daubechies filtreleri ile hesaplamada analiz safhası

Şekil 6.3. 'de değişik uzunluktaki Daubechies filtreleri ile yapılmış olan sıkıştırma uygulamalarının sonuçları verilmiştir. Burada daha yüksek filtre derecesi için daha düşük sıkıştırma oranlarının elde edildiği görülmektedir(Mandal, 1993).



Şekil 6.3. Değişik uzunluktaki Daubechies filtreleri ile yapılmış olan sıkıştırma uygulamaları

7. GÖRÜNTÜ SIKIŞTIRMA STANDARTLARI

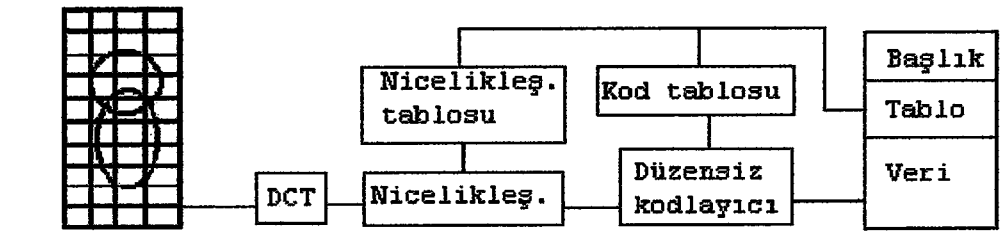
7.1 JPEG(Joint Photograph Expert Group)

JPEG oldukça iyi bir sıkıştırma oranı sağlar fakat sıkıştırma ve ayrıştırma safhasında oldukça yavaş olup tek dezavantajı budur.

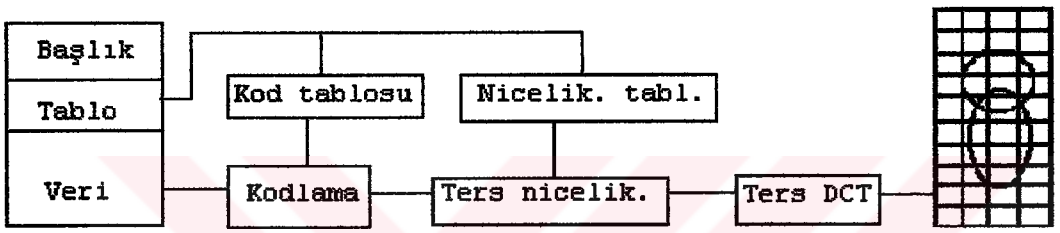
JPEG 'de görüntü önce küçük alt bloklara ayrılır. Yalnız bu bloklar 8x8 'lik olmalı ve görüntünün tamamı sekizin katı olmalıdır. Eğer görüntü sekizin katı değilse en son satır veya kolonun bir kopyası çıkartılarak bu bölgelere uygun bir şekilde yerleştirilir.

Bu alt bloklar iki boyutlu AKD 'ye maruz bırakılır ve AKD 'nin en büyük avantajı katsayıların çoğunun ilk satır ve sütun elamanını da toplanması ve diğer arta kalan elamanların çoğunun sıfır olmasıdır. Bu durum bize iletilecek olan veri miktarında avantaj sağlar. Bu katsayılar nicelikleştirme denilen bir nicelikleştirme işleminden sonra yuvarlama işlemine tabi tutulacak olursa katsayıların çoğu sıfır olur. Arta kalan katsayılar DC ve AC katsayılar olarak adlandırılır. İlk satır ve sütun elamanı DC, diğer elamanlar AC katsayılar olarak adlandırılır(Bhaskaran, 1996).

Ayrıştırma safhasında ise şifreleme işleminde yapılan işlemlerin tam tersi yapılarak orijinal görüntüye ulaşılmaya çalışılır. Önce kod tablosundan ve kodlayıcıdan geçen veriler ters nicelikleştiriciye girerek ters AKD 'den geçer ve orijinal piksel değerlerine ulaşılır. Yalnız dikkat edilmesi gereken nokta nicelikleştirme işleminin JPEG 'i kayıplı yapmasıdır. Şekil 7.1 a ve Şekil 7.1 b'de bu işlemlerin blok diyagramı verilmiştir.



(a)



(b)

Şekil 7.1.a-JPEG şifreleme blok diyagramı. **b-** JPEG ayrıştırma blok diyagramı

7.1.1 DC katsayıların kodlanması

DC katsayılar kodlanırken şu yol takip edilir:
DC katsayılar kodlanırken orijinal görüntüden elde edilen görüntü ki bu görüntü önce iki boyutlu AKD ve nicelikleştirme(quantization) işleminden geçmiş olup bunun ilk satır ve sütun elamanı DC_1 olarak adlandırılır. DC_2 olarak adlandırılan alt görüntünün ilk elamanı olan DC bileşen ki bu da önce pikseller yedi kere kaydırılarak elde edilmiş olup bu ikisinin farkı üzerinde işlem

yapılması ile gerçek DC bileşen bulunmuş olunur. Bunun kodlanmasında izlenecek yol aşağıdaki gibidir:

1- Önce tablo 7.1 'den DC katsayının hangi kategoriye girdiği bulunur.

Tablo 7.1. JPEG katsayı kodlama kategorisi

ARALIK	DC FARK KATEGORİSİ
0	0
-1,1	1
-3,-2,2,3	2
-7,..-4,4..7	3
-15,...-8,8...15	4
-31,....-16,16....31	5
-63,.....-32,32.....63	6
-127,.....64,64.....127	7
-255,.....-128,128.....255	8
-511,.....-256,256.....511	9
-1023,.....-512,512.....1023	A
-2047,.....-1024,1024.....2047	B
-4095,.....-2048,2048.....4095	C
-8191,.....-4096,4096.....8191	D
-16383,.....-8192,8192.....16383	E
-32767,.....-16384,16384.....32767	F

Tablo 7.2. JPEG şartsız kabul edilen DC Kod Tablosu

KATEGORI	TABAN KOD	KOD UZUNLUĞU
0	010	3
1	011	4
2	100	5
3	00	5
4	101	7
5	110	8
6	1110	10
7	11110	12
8	111110	14
9	1111110	16
A	11111110	18
B	111111110	20

2- Bulunan bu değerin tablo (7.2) 'den taban kodu ve toplam uzunluğu bulunur.

3- Toplam kod uzunluğu - taban kod uzunluğu = DC farkın en düşük byte 'ı(LSB) şeklinde düşünürsek DC farktan alınacak bit sayısı bulunur. Bu bitler en düşük anlamlı bitlerdir.

4-Önce taban kodu sonrada DC farkın LSB 'si olmak üzere kodlama tamamlanmış olur.

Örneğin DC fark -20 olsun. -20 'nin Tablo (7.1) 'den kaçıncı kategoriye girdiği bulunur. -20 kategori beşe girer. Kategori beş Tablo (7.2) 'den toplam sekiz bit ile ifade edilir ve bu sayının taban kod 'u 110 dır. Arta kalan beş bit ise -20 'den bir çıkartılarak bulunur. $-20-1=(1111101011)_b$ olduğundan en düşük beş biti 01011 olup sonuç olarak kodlanmış değer 11001011 olur.

7.1.2. AC katsayıların kodlanması

AC katsayılar sıralanırken zig-zag sırada taranarak sıralanırlar. Zig-zag sırada tarama kodlamada fazladan gelecek olan sıfırlardan kurtaracaktır.

Tablo 7.3. JPEG katsayı kodlama kategorisi

ARALIK	AC KATEGORİ
0	N/A
-1,1	1
-3,-2,2,3	2
-7,..-4,4..7	3
-15,...-8,8...15	4
-31,....-16,16....31	5
-63,.....-32,32.....63	6
-127,.....64,64.....127	7
-255,.....-128,128.....255	8
-511,.....-256,256.....511	9
-1023,.....-512,512.....1023	A
-2047,.....-1024,1024.....2047	B
-4095,.....-2048,2048.....4095	C
-8191,.....-4096,4096.....8191	D
-16383,.....-8192,8192.....16383	E
-32767,.....-16384,16384.....32767	N/A

AC katsayılar kodlanırken şu yollar takip edilir:

1- Önce tablo 7.3. 'den hangi kategoriye girdiği bulunur.

2- Run/kategori oranına bakılır. Burada run terimi AC katsayılar zig-zag sırada taranıp tekrar sıralanan

katsayıların önüne gelen sıfırların katsayıların toplamıdır.

3- Karar verilen bu oranla Tablo (7.3) 'den taban kodu ve toplam kaç bit ile ifade edileceği bulunur(Gonzalez, 1993).

Tablo 7.4. JPEG AC katsayı kodlama tablosu

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	1010 (= EOB)	4			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	1111111111000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111001	14	8/7	1111111110111011	23
0/8	11111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	11111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	11111110110	13	9/4	1111111111000001	20
1/5	11111111110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	11111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23
2/8	1111111110001101	24	A/8	1111111111001110	24
2/9	1111111110001110	25	A/9	1111111111001111	25
2/A	1111111110001111	26	A/A	1111111111010000	26
3/1	111010	7	B/1	11111010	10
3/2	111110111	11	B/2	1111111111010001	18
3/3	11111110111	14	B/3	1111111111010010	19
3/4	1111111110010000	20	B/4	1111111111010011	20
3/5	1111111110010001	21	B/5	1111111111010100	21
3/6	1111111110010010	22	B/6	1111111111010101	22
3/7	1111111110010011	23	B/7	1111111111010110	23
3/8	1111111110010100	24	B/8	1111111111010111	24
3/9	1111111110010101	25	B/9	1111111111011000	25
3/A	1111111110010110	26	B/A	1111111111011001	26

Tablo 7.4. 'ün devamı

Run/ Category	*Base Code*	*Length*	*Run/ Category*	*Base Code*	*Length*
4/1	111011	7	C/1	1111111010	11
4/2	1111111000	12	C/2	1111111111011010	18
4/3	1111111110010111	19	C/3	1111111111011011	19
4/4	1111111110011000	20	C/4	1111111111011100	20
4/5	1111111110011001	21	C/5	1111111111011101	21
4/6	1111111110011010	22	C/6	1111111111011110	22
4/7	1111111110011011	23	C/7	1111111111011111	23
4/8	1111111110011100	24	C/8	1111111111100000	24
4/9	1111111110011101	25	C/9	1111111111100001	25
4/A	1111111110011110	26	C/A	1111111111100010	26
5/1	1111010	8	D/1	1111111010	12
5/2	1111111001	12	D/2	1111111111100011	18
5/3	1111111110011111	19	D/3	1111111111100100	19
5/4	1111111110100000	20	D/4	1111111111100101	20
5/5	1111111110100001	21	D/5	1111111111100110	21
5/6	1111111110100010	22	D/6	1111111111100111	22
5/7	1111111110100011	23	D/7	1111111111101000	23
5/8	1111111110100100	24	D/8	1111111111101001	24
5/9	1111111110100101	25	D/9	1111111111101010	25
5/A	1111111110100110	26	D/A	1111111111101011	26
6/1	1111011	8	E/1	1111111010110	13
6/2	11111111000	13	E/2	1111111111101100	18
6/3	1111111110100111	19	E/3	1111111111101101	19
6/4	1111111110101000	20	E/4	1111111111101110	20
6/5	1111111110101001	21	E/5	1111111111101111	21
6/6	1111111110101010	22	E/6	1111111111110000	22
6/7	1111111110101011	23	E/7	1111111111110001	23
6/8	1111111110101100	24	E/8	1111111111110010	24
6/9	1111111110101101	25	E/9	1111111111110011	25
6/A	1111111110101110	26	E/A	1111111111110100	26
7/1	11111001	9	F/0	1111111110111	12
7/2	11111111001	13	F/1	1111111111110101	17
7/3	1111111110101111	19	F/2	1111111111110110	18
7/4	1111111110110000	20	F/3	1111111111110111	19
7/5	1111111110110001	21	F/4	1111111111111000	20
7/6	1111111110110010	22	F/5	1111111111111001	21
7/7	1111111110110011	23	F/6	1111111111111010	22
7/8	1111111110110100	24	F/7	1111111111111011	23
7/9	1111111110110101	25	F/8	1111111111111100	24
7/A	1111111110110110	26	F/9	1111111111111101	25
			F/A	1111111111111110	26

4-Bundan sonraki işlemler DC katsayının kodlanmasındaki ile aynıdır,

5-EOB göstergeci kodlama dizisinin bitiğini gösterir ve 1010 (0/0) ile verilir.

Mesela elimizde zig-zag sırada taranmış AC katsayı sırası aşağıdaki gibi olsun:

-26 -3 1 -3 -2 -6 2 -4 1 -4 1 1 5 0 2 0 0 -1 2 0 0 0 0 0 -1 -1 EOB

Şimdi 25. eleman olan -1 'i kodlamaya çalışalım:

1- Tablo 7.3 'den bakılırsa (-1) kategori 1 'e girer.

2- Önündeki sıfır sayısı 5 (run=5), kategori=1, run/kategori=5/1

3- Tablo 7.4 'den taban kod=1111010 , toplam uzunluk=8 olmalı

4- 11111110 'ın son biti işleme girer.

5- Sonuç kodlama 11110100 olarak bulunur.

7.2. MPEG2 Görüntü Formatı

Görüntünün oluşturulması için MPEG üç tane resim tipi kullanır:I(Ara kodlu-Intra), P(Tahmini-Predective) ve B(Çift yönlü-Bidirectional).

"I" resimleri tek bir resmin sıkıştırılmış bilgisini içerir. Ara resimler yaklaşık 0.5sn de bir veri akışının içerisine yerleştirilir. Bu durum sıkıştırılmış veri materyali üzerinde ileriye atlamayı sağlar. "I" resimleri, "B" ve "P" resimlerinin ortaya çıkartılması için yola çıkartılan resimlerdir(Bhaskaran, 1996).

"P" resimleri "I" ya da "P" tipi geçmiş resimlerden hareketin tahmini(motion estimation) ile türetilir. Hem "P" hem de "B" resimleri yer değişikliği vektörlerinin

hesaplanması için referans resimler olarak da işlev görürler.

Bir "B" resmi, önceki ve sonraki(bu yüzden Bidirectional denmektedir) bir "P" ya da "I" resminden türetilir.

Örneğin .....IBBPBBP biçiminde düzenlenmiş şifrenin çözülmesi şu şekilde olur. "I" resmi tam bir bilgi içerdiklerinden dolayı çözülmesi kolaydır. Daha sonra "I" çerçevesinin bilgisine başvurularak ilk "P" resmi oluşturulur ve bunların yardımıyla iki tane "B" resmi oluşturulur.

Veri akımında kodlamanın temel öğelerini birinden ayırmak ve uygun erişme mekanizmaları sağlamak amacıyla altı tabakalı yapılar kullanılır.

7.2.1 MPEG2 Kodlama Tabakaları

MPEG2 kodlama tabakaları aşağıdaki altı kısımdan oluşur:
Sıra tabakası
Resim grubu(GOP) tabakası
Resim tabakası
Zaman dilimi(slice) tabakası
Makroblok tabakası
Blok tabakası

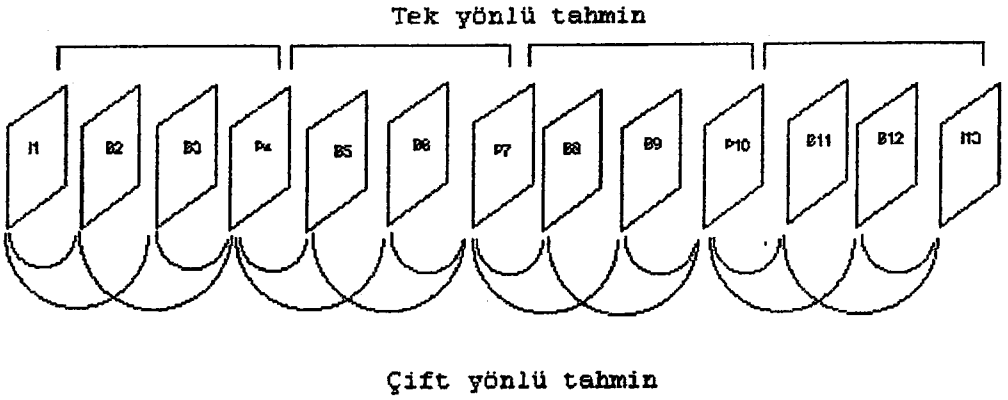
7.2.1.1. Sıra tabakası

En üst tabaka olan sıra tabakası(sequence layer), resim sırasına kontekst (bağlam) erişimin yapılmasını sağlar. Sıra tabakası, bir başlıkta resim formatıyla ilgili

bilgiler ve kod çözücü için gerekli çerçeve parametrelerini karşılayan bazı uygulamaya yönelik özel detayları bulundurur.

7.2.1.2. Resim grubu(gop) tabakası

Bir resim sırasını periyodik olarak tanımlar. Kod çözme işlemine, kodu çözülecek bir resim için gerekli olan bilginin öncesi yoksa, yani tekrar oluşturma için başka resimler gerekmiyorsa başlanabilir. Böyle bir resme intrakodlu resim denir. Şekil 7.2. 'de gösterildiği gibi düzenli aralıklarla bu tür "I" resimlerin iletilmesi sağlanmalıdır. Bunlar örneğin program değiştirildiğinde resim sırasına tekrar ulaşılmasını sağlar. "P" resimler ise resim sırasındaki dayanak noktaları olup sadece tahmine dayalı fark resimlerinin oluşturulmasında kullanılırlar. Burada sadece tahminden farklı görüntüler iletilir. Böylece kodlama için gereken bit sayısı "I" resmine göre 2/3 oranında azalır. Zaman açısından "I" ve "P" resimleri arasında kalan resimler çift yönlü olarak tahmini kodlanır. Burada iletimde "I" ve/veya "P" resminden olan fark iletilir ve gerekli bit sayısı gerekenden 2/3 oranında azalır. "I", "P" ve "B" resimlerinin sıkıştırma faktörleri birbirinden farklı olduğundan dolayı veri miktarı sürekli olarak değişir. Ama komple bir resim düzleminde ortalama değerlerle verilen bir ortalama veri miktarı verilir. Bu nedenle kodlayıcının çıkışındaki tampon, resim grupları arasındaki veri farklarını dengeleyecek şekilde düzenlenmiştir. Buna paralel olarak kod çözücünün girişinde de ortalama hızla gelen verilerin gerektiği kadar



Şekil 7.2. Resim akış yönü ve tahmin resimleri

hızlandırılması ve yavaşlatılması için tampon belleği bulundurulmalıdır

7.2.1.3. Resim tabakası

Resimlerin kodlanması için gerekli bilgiler resim tabakasına yerleştirilir. Burada 8x8 'lik bloklar AKD 'ye maruz kalır.

7.2.1.4. Zaman dilimi(slice) tabakası

Yeniden senkronizasyon sağlamak için kullanılır. Çünkü başlığında kesin bir resim konumu belirlenmemiştir.

7.2.1.5. Makroblok tabakası

Kod çözücüdeki tekrar oluşumun(reconstruction) temellerini oluşturur.

7.2.1.6. Blok tabakası

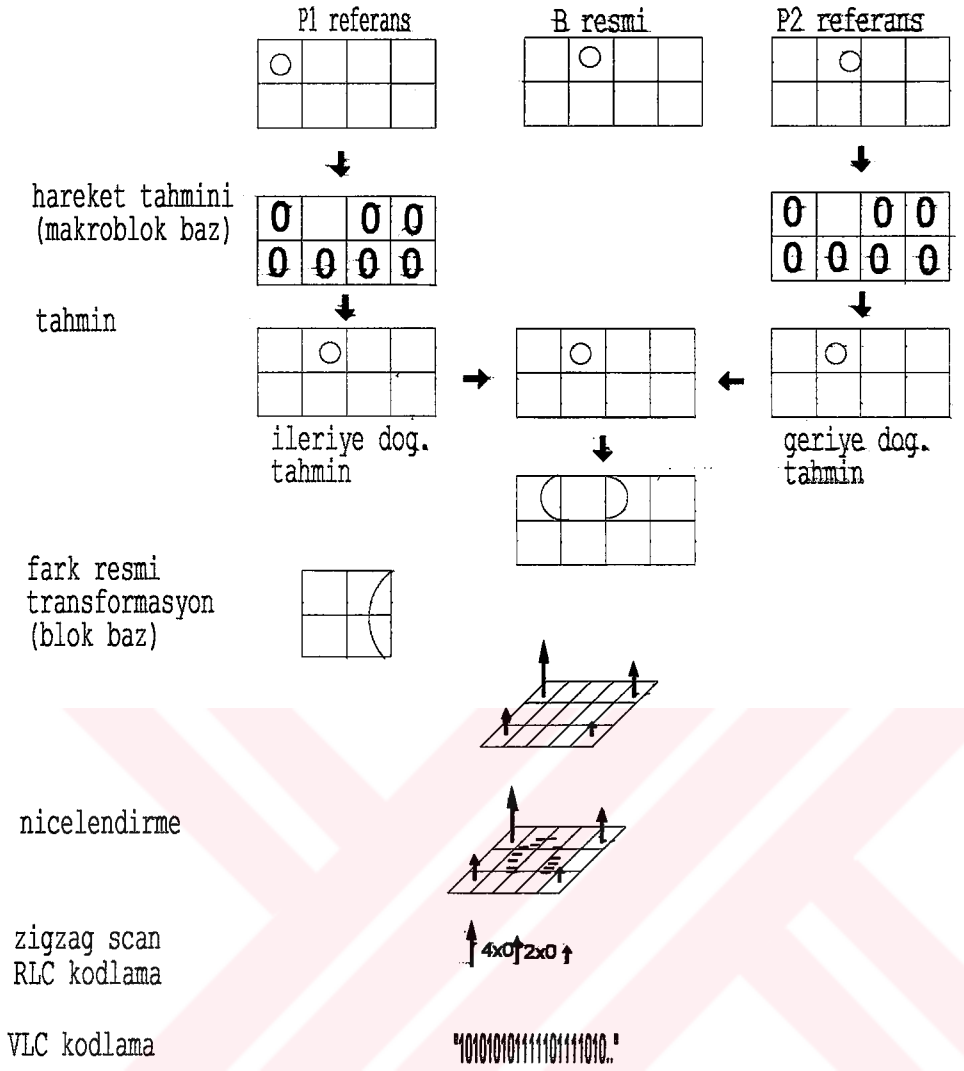
DCT 'ye maruz kalan bloklar iletilir. Intrakodlu bloklarda orijinal görüntülerden dönüştürülen bilgiler söz konusudur. P ve B bloklarında ise fark görüntüleri dönüştürülür ve kodlanır.

7.2.2. Çift yönlü hareket kompanzasyonlu veri sıkıştırması

"P" resimler tek yönde yani önceki resimlerden yola çıkılarak kodlanırken "B" resimlerinde iki yönde tahmin yapılarak kodlama yapılır. Yani zaman akışı içinde geçmişten geleceğe doğru yola çıkılarak tahmin için referans görüntüler olarak bir resim grubu içinde sadece "I" ve "P" resimleri kullanılır.

Örneğin P1 ve P2 resimleri arasında bulunan "B" resmi veri sıkıştırma amacıyla sadece fark resmi olarak iletilir. Burada görüntü akışı içindeki fazlalıklardan yararlanılır. Art arda gelen görüntüler birbirinden çıkartıldığında geriye sadece bir akışın görüntü içinde sebep olduğu değişiklikler kalır ve bunların yeri işaretlenir. Eğer görüntüde bir değişiklik olmazsa kodlayıcı bitlerden tasarruf edebilir çünkü bu resim daha önce iletilmiş idi.

Her makroblok içinde P1 ve P2 görüntülerinin "B" görüntüsüne göre ne kadar değiştiği saptanır. Bu işlem fark resminin ne kadar değiştiği ve fark resim bileşenlerinin azaltılması için yapılır. Yukarıdaki örnekte siyah bir küre bir ızgara şeklindeki fon üzerinde hareket etmektedir. Hareketsiz noktalar üzerinde kayma vektörü sıfırdır(pratikte yatay ve dikey olmak üzere iki boyutlu bir vektör kullanılabilir).



Şekil 7.3. MPEG veri sıkıştırma ilkesi

"B" görüntülerinin tahmini için MPEG2 farklı imkanlar sunar: Tek,çift, yarım ve tam yönden tahmin yapabilir. Kodlayıcı her makroblok için en düşük veri miktarını oluşturan yöntemi seçebilir. Gerçekten de "B" resminin sol üstteki bloğu geriye doğru tahmin ile daha iyi etüt edilebilir. Burada mantığın daha iyi anlaşılabilmesi için ileriye ve geriye doğru tahminler yapılmıştır. İleriye ve

geriye doğru yapılan tahminlerin ortalaması alınarak yaklaşık bir tahmin yapılarak hata yarıya indirilir.

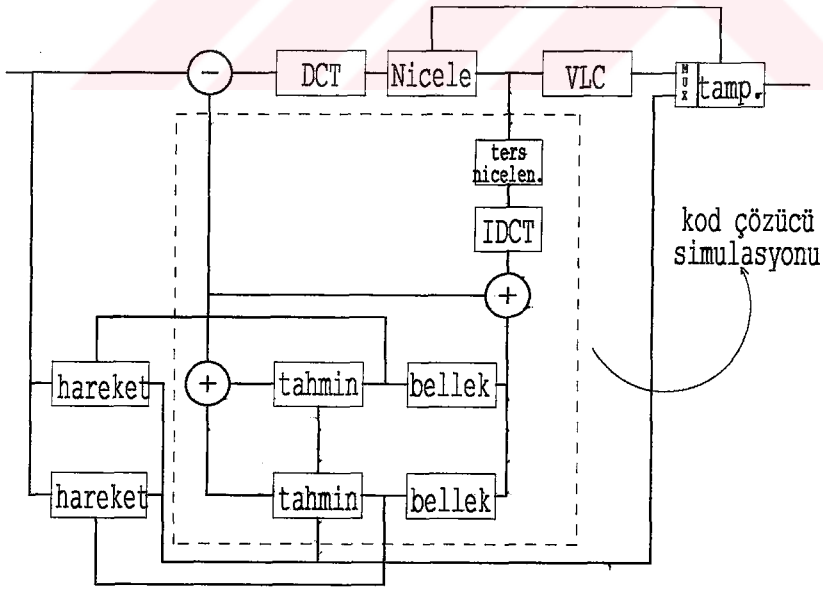
Şekil 7.3. 'den de görülebileceği gibi kodlayıcı P1 ve P2 görüntülerini değil de yeniden oluşturulan P1' ve P2' görüntülerini tanır. Bu durum itibariyle kodlayıcı P1 ve P2 bazında sıkıştırma P1' ve P2' bazında bir tahmin yapmak zorundadır. Oluşturulan fark resimleri 2 boyutlu AKD 'ye tabi tutulursa görüntü içeriği frekans domenine dönüştürülür. AKD 'nin avantajı enerjiyi birkaç katsayıda yoğunlaştırmasıdır. Örneğin bir bloktaki parlaklık sabit ise sadece DC bileşenin iletilmesi yeterlidir çünkü geriye kalan terimler ki bunlar AC terim olarak adlandırılır sıfır olacaklardır. Daha sonra bir nicelikleştirme matrisi ile çarpılan katsayılar DC katsayıdan başlamak üzere zig-zag sırada taranır. Elde edilen katsayılar Huffman veya VLC kodlamasından geçirilerek kodlanır. VLC 'de sıfırdan farklı bir veri kelimesinden önceki sıfırlar o veri kelimesinde toplanır. Sonuç olarak bu veri kodlama tekniği ile gerekli veri miktarı azaltılır.

7.2.2.1 Kodlayıcı ve kod çözücünün yapısı

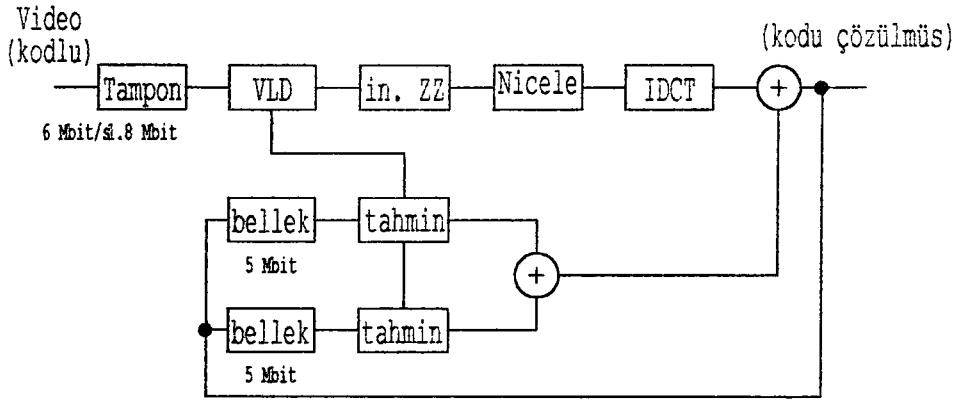
Kod çözücünün kendisi de kodlayıcının bir bölümünü oluşturur çünkü bu referans resimlerin kod çözücüde yeni üretilebilmesi için gereklidir. VLC ve RLC kodlamasının simülasyonuna gerek yoktur çünkü bunlar kayıpsızdır ve hiç bir değişikliğe uğramaz. İki referans resmi için yapılması gereken hareket tahmini ise çok büyük bir hesap gücü gerektirir. Buna örneğin bir de makroblok kodlamasını türünü belirleyen kumanda blokları -ki çizilmemiştir- eklenir. Üretilen veri hızındaki dalgalanmalar ise tampon

belleği ile kompanze edilir. 'Karmaşık' resimlerde veri miktarı fazla büyür ve kodlayıcının çıkışındaki tampon belleğine sığmaz hale gelirse bu durumda AKD katsayıları daha kabaca niceliklestirilir. Elbette bu durumda doğruluğun azalması ve kod çözücüdeki tekrar oluşumun olumsuz yönde etkilenmesi doğaldır. Ortaya çıkan olumsuzluklar özellikle hareketli resimlerin sınırlarında ortaya çıkar.

Kod çözücüde sıkıştırma işlemi ters sıraya göre çözülür ve fark resmi blok halinde hesaplanır. Referans resimlerin önceden iletilmesi ve bellekte saklanması gerekir. Bu demektir ki bir resim grubunda kodlu resimlerin sırası onların ekranda görüntülenme sırasından farklıdır. Birlikte iletilen hareket vektörleri aracılığı ile kod çözücü fark resimleri ile düzeltilen tahmini hesabını yapar. Şekil 7.4. ve 7.5. 'da bir MPEG kodlayıcısının ve kod çözcüsünü temel yapı elemanları görülmektedir.



Şekil 7.4. MPEG kodlayıcısı



Şekil 7.5. MPEG kod çözücüsü

7.3. H.261 Görüntü Sıkıştırma Standardı

H.261 görüntü sıkıştırma standardı 1990 'da ITU (International Telecom Union) tarafından gerçekleştirildi. 64Kbit/sn 'nin katlarındaki veri oranları için dizayn edildiği için bazen px64Kbit/sn olarak da adlandırılır. Burada p, 1-30 arasında değişen bir tamsayıdır. Bu veri oranları bu kodlama için dizayn edilmiş ISDN veri hatları için oldukça uygundur. Genel olarak ISDN video telefon uygulamaları için kullanılan bir standarttır.

7.3.1. Uygulamaları

Genel olarak uygulamaları videofon ve video konferanstır. Bu nedenle kodlama algoritması gerçek zamanda ve minimum gecikme ile olmalıdır. P=1 ve/veya p=2 olması

durumunda masa üstünde (desktop) gerçek zamanda haberleşme mümkün olabilir. Eğer p daha büyük veya altıya eşit ise daha kompleks resimlerin iletilmesi mümkün olabilir ki bu da video konferans uygulamalarında kullanılır.

7.3.2. Video kodlama algoritması

Video kodlamanın temel yaklaşımı ihtiyaç duyulmayan fazla bilgileri elimine etmektir. Kaynak(source) ve düzensiz(entropy) kodlama olarak iki ana kodlama planı vardır. Kaynak kodlamada kayıplar söz konusudur ve resim kalitesi azalır. Bu ayrıca ara çerçeve ve ara çerçeve kodlama olarak ikiye ayrılabilir. Ara çerçeve kodlama ilk resme ve değiştikten sonraki resme uygulanır. Ara çerçeve kodlama hareket eden nesnelerin bulunduğu benzer resimlerin bir dizisi içindir. Ara çerçeve kodlama çerçeveler arası geçişteki fazlalıklardan kurtarır. Düzensiz kodlama da fazlalıklardan kurtarır ve aynı zamanda kayıpsızdır. H.261 hem ara çerçeve hem de ara çerçeve kodlamayı kullanır. H.261, AKD ve DPCM(Differansial Pulse Code modulation) planları ile hareket tahmininin bir birleşimidir. Ara çerçeve modda DPCM bir işlev görmez. Resimdeki her 8x8 'lik blok AKD ile dönüşüme uğratılır. Doğrusal olarak quantize edilir. Video çoğullayıcıya gönderilir. Aynı resim çerçeveleri ters quantizer ve ters dönüşümden kurtarılır ve ara çerçeve kodlama için resim hafızasında kaydedilir. Ara çerçeve kodlamada DPCM aktiftir. Tahmin, bir önceki ve bir sonraki makroblokların karşılaştırılıp aralarından bir hareket farkı çıkartılması ile olur. Eğer kaynak ve tahmini makroblok arasındaki fark eşikten az ise o makroblok için veri dönüştürülmez, diğer durumlarda AKD uygulanır,

quantlanır ve hareket vektör bilgisi ile birlikte çoğullayıcı kodlayıcıya gönderilir.

7.3.3. Çoğullayıcının yapısı

Çoğullayıcı dört kısımdan oluşur. Resim tabakası; bir video resime tekabül eder, blok grupları, makroblok, bloklar.

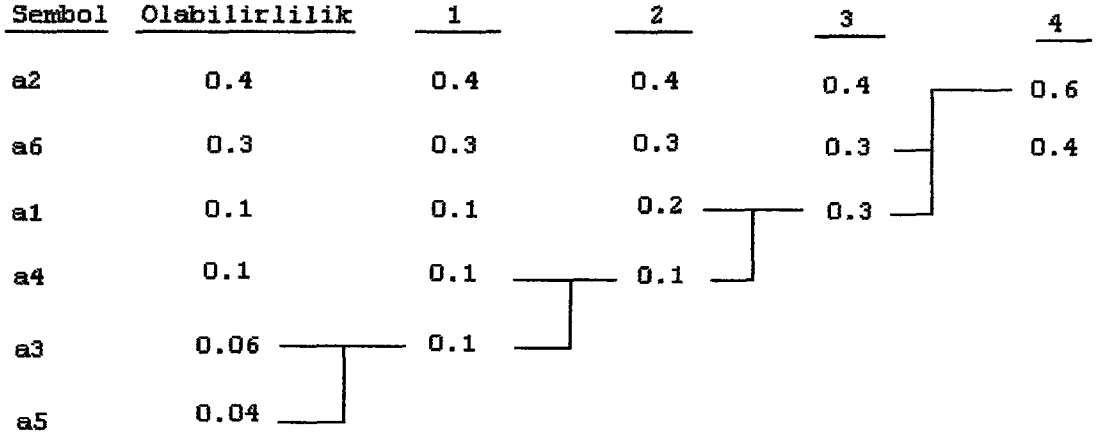
8. KODLAMA FAZLALIKLARINDAN ARINDIRMA

8.1. Değişebilir Uzunlukta Kodlama(DUK-VLC)

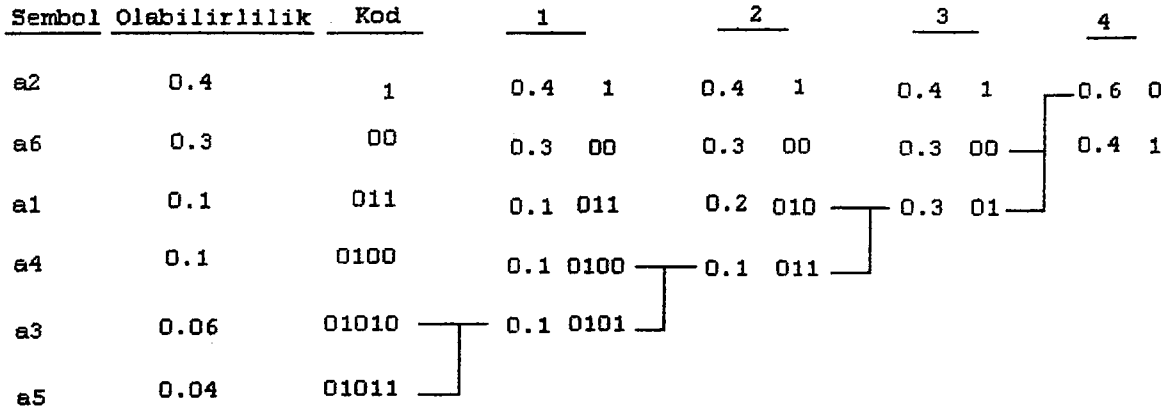
Kodlamadaki amaç kodlama fazlalıklarından kurtulmaktır. Burada incelenecek kodlama türleri hatasız kodlama kategorisine girmektedir. Hatasız kodlamadan kasıt kodlanmış sembolün çözüldüğünde tekrar orijinal haline dönebilmesidir. Kodlama kayıplı ve kayıpsız olarak ikiye ayrılır(Gonzalez, 1993).

8.1.1. Huffman kodlaması

Kodlamadaki fazlalıklardan kurtulmak için en popüler yöntem olup temel mantığı oluş tekrarlılığı fazla olana kısa, oluş tekrarlılığı az olana uzun kod uzunluğu vermektir. Ayrıca diğer kodlama türlerine göre en büyük avantajı en kısa kod uzunluğunu vermesidir. Elimizde şekil 8.1. 'deki gibi oluş tekrarlılıkları olan bir sembol grubu olsun.



Şekil 8.1. Huffman yaklaşımındaki ilk adım



Şekil 8.2. Huffman kodlamadaki ikinci adım

Önce olabilirliliklerine göre sıralanan değerler en alttan başlamak üzere en alttaki iki olabilirlilik toplanarak tekrar sıralanır. Bu sıralama işlemine sadece iki tane olabilirlilik kalıncaya kadar devam edilir.

Şekil 8.2. 'deki gibi iki tane durum kalınca olabilirliliği fazla olana keyfi olarak 0 veya 1 verilir. Daha sonra bir önceki adımda toplanan olabilirlilikler ters yönde işlenmeye başlanır. Örneğin 0.6 olabilirliliği iki tane 0.3 'lük olabilirliliğin toplamından oluşuyordu: Üstekinin olabilirliliği fazla olduğu için soluna 0 alttakinin olabilirliliği az olduğundan dolayı soluna 1 konur ve işlem en başa ulaşılıncaya kadar devam eder.

Dikkat edilmelidir ki bu kodlanmış değerler tek olarak ayrıştırılabilir(uniquely decedable)'dir. Örneğin şu şekilde bir kod dizisi gelsin:

010100111100

01010 011 1 1 00 = $a_3\ a_1\ a_2\ a_2\ a_6$ Şeklinde ve tek olarak ayrıştırılabilir.

a_3 a_1 a_2 a_2 a_6

Sonuç olarak kodlama için ihtiyaç duyulan toplam bit sayısı;

$L_{avg}=(0.4).1+ (0.3).2+(0.1).3+(0.1).4+(0.06).5+(0.04).5=2.2$ bit/sembol

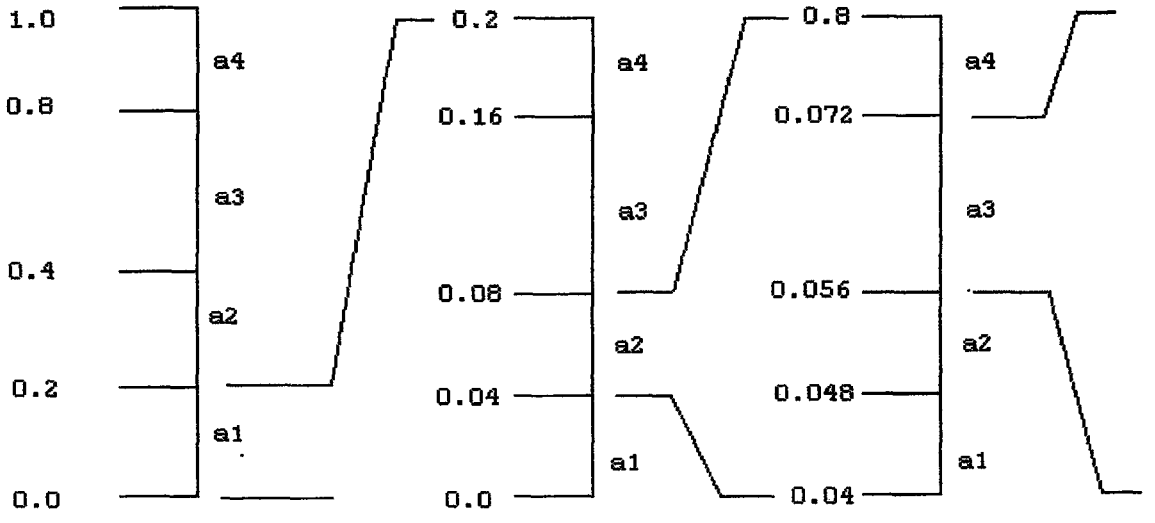
8.1.2. Aritmetik kodlama

Şekil 8.3. ve şekil 8.4. aritmetik kodlama işleminin temellerini gösterir. Burada $a_1a_2a_3a_3a_4$ şeklinde bir sembol dizisini geldiği varsayılarak bu dizi üzerinde kodlama mantığı verilmeye çalışılacaktır. Olabilirlikleri de tablo 8.1. 'deki gibi olsun.

Tablo 8.1. Olabilirliliklerin sıralanması ve aralıkların tayini

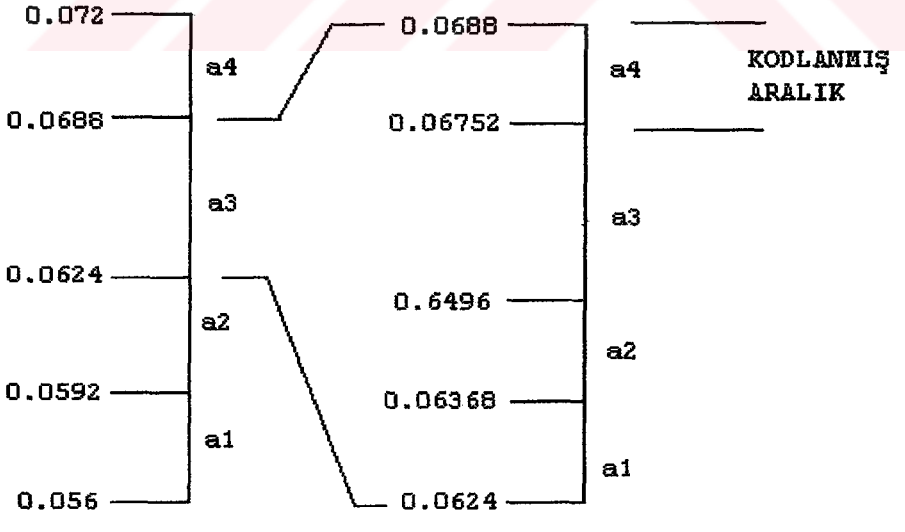
SEMBOL	OLABİLİRLİLİK	ARALIK
a_1	0.2	[0.0 0.2)
a_2	0.2	[0.2 0.4)
a_3	0.4	[0.4 0.8)
a_4	0.2	[0.8 1.0)

Aritmetik kodlamanın temeli gelen sembol dizisine göre [0.0 1.0) yarı açık aralığında bu aralığı daraltarak daha küçük bir aralıkta bu sembol dizisini ifade etmektir. Bunu yaparken şu yollar takip edilecektir.



Şekil 8.3. Aritmetik kodlamada birinci adım

Şeklinde devam edilerek en son şifre kelimesindeki sembole (a_4) ulaşılıncaya kadar bu daraltma işlemine devam edilir.



Şekil 8.4. Aritmetik kodlamada son adım

Sembolde önce a1 geldiğinden dolayı a_1 'e göre bir daraltma yapılır. Burada a_1 [0.0 0.2) yarı açık aralığında olduğundan dolayı [0.0 1.0) yarı açık aralığı a_1 aralığına göre daraltılacaktır yani bu aralık 5 'e bölünerek indirgenecektir.

Diğer sembol elamanları içinde bu işlemler tekrarlanırsa sonuç olarak [0.06752 0.0688) aralığı elde edilmiş olunur. Artık bu aralıkta gelen tüm sembol dizileri $a_1a_2a_3a_3a_4$ olarak algılanacaktır.

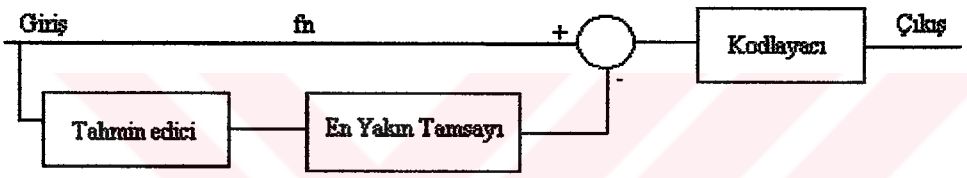
Bu teknik Huffman 'ın yaptığı gibi her bir kod sembolünü başka bir kod sembolüne dönüştürmeye ihtiyaç duymaz.

Aritmetik kodlamayı sınırlayan iki etken vardır.
1-Mesaj sonu belirteci (End Of Message Indicator-EOB) diğer mesajlardan bir yolla ayrılmalıdır. Bunun için ek bilgi iletimine ihtiyaç duyulacaktır.
2-Diğer bir önemli sınırlayıcı unsurda sonlu aritmetik elaman kullanımıdır(Gonzales, 1993).

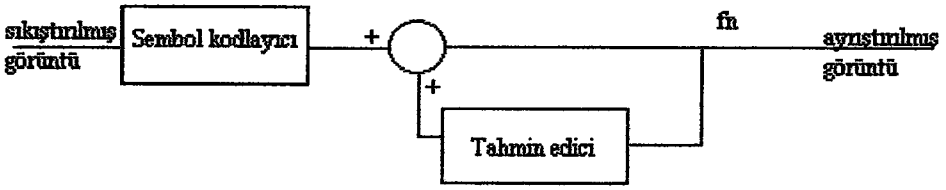
8.1.3. Kayıpsız tahmini kodlama

Tahmine dayalı kodlama tekniğinin temel yaklaşımı orijinal görüntü ve tahmin edilmiş görüntü arasındaki farkların işlenmesidir. Buradaki tahmin, resim akışında bazı temel resim noktaları seçilmesi ve temel resimlerin birbirlerine göre olan değişimlerinin incelenmesidir.

Sistem temel olarak şekil 8.5 'deki gibi bir kodlayıcı ve şekil 8.6 'daki bir ayrıştırıcıdan oluşur. Her ikisinde de bulunan tahmin edici aynıdır. Burada giriş görüntüsündeki piksel değeri fn ile gösterilir. Tahmin edicinin çıkışı en yakın tamsayıya yuvarlanır ve $\hat{f}_n$ ile gösterilir. Tahmin hatası ya da fark en=fn- $\hat{f}_n$ ile verilir ki bu fark değeri VLC kullanılarak sıkıştırılmış veri akışının bir sonraki elamanını üretmek için kullanılır.



Şekil 8.5. Kayıpsız tahmini kodlama safhası



Şekil 8.6. Kayıpsız tahmini ayrıştırma safhası

8.1.4. Kayıplı tahmini kodlama

Kayıpsız tahmini kodlamadan tek farkı nicelikleştiricinin bulunmasıdır. Bilindiği üzere nicelikleştirme işlemi sıkıştırma işlemini kayıplı yapar.

8.1.5. Tek ve iki boyutlu run length kodlama

Run length kodlamanın iki türü vardır: Tek boyutlu run length kodlama ve iki boyutlu run length kodlama. Tek boyutlu run length kodlama bir görüntü satırındaki siyah bit kümeleri arasındaki ilişkiyi kullanır. İki boyutlu run length kodlama ise bunun basit bir uzantısı olmayıp bir üst satırı referans alarak bu iki satır arasındaki ilişkiyi kullanır.

8.1.5.1. Tek boyutlu run length kodlama algoritması

Elimizdeki sayısal görüntü satırının değerleri aşağıdaki gibi olsun:
000011111111001111111111100000001111000
Bu satırı kodlarken şu şekildeki sayı çiftlerinden faydalanılarak kodlama yapılır:<siyah bit kümesinin yeri, siyah bit kümesinin uzunluğu>. Kodlama sonunda satırın bittiği satır sonu belirteci (EOL) ile belirlenir. O zaman bu satır şu şekilde kodlanır.
<5,8>, <15,11>, <33,4> EOL

8.1.5.2. İki boyutlu run length kodlama algoritması

8.1.5.2. İki boyutlu run length kodlama algoritması

Burada bir alt satır bir üst satırı referans alarak işlem yürütülür ve (δ,δ') gibi iki sayı çifti ile ifade edilen rakamlar kodlanarak iletilir. Burada δ referans satırın siyah bit kümesinin başlangıç yeri ile kodlanacak olan satırın siyah bit kümesinin başlangıç yerleri arasındaki farkı, δ' ise referans satırın siyah bit kümesinin uzunluğu ile kodlanacak olan satırın siyah bit kümesinin uzunlukları arasındaki farktır.

Şimdi bu mantıkla hareket ederek aşağıdaki gibi bir sayısal görüntü bölümünü şu şekilde kodlayabiliriz:

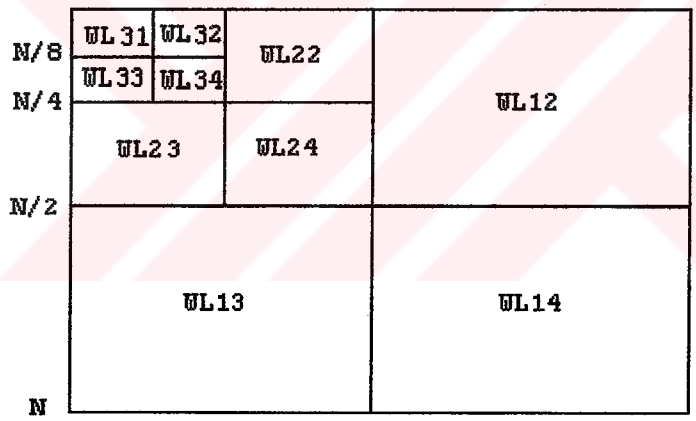
```
01110000111111000000000000011111111100000
00011100000001111111000000111100000000000
01111111111111111111110000011111111111100
11111000000111111110000111111000001111111
```

```
<2,3>, <10,5>, <28,9>, <EOL>
<2,0>, <4,1>, <-2,-5>, <EOL>
<-2,18>, <14,6>, <EOL>
<-1,-16>, <16, 5>, <-5, -7>, <6,-4>, <EOL>
```

Eğer referans satırdaki siyah bit kümesi sayısı kodlanacak olan satırla aynı ise problem yoktur. Fakat referans satırdaki siyah bit kümelerinin sayısı altındakinden fazla ise arta kalan kümeler göz ardı edilir. Referans satırdaki siyah bit kümelerinin sayısı kodlanacak satırınkinden az ise en son kalan siyah bit kümesi bundan sonraki işlemlere referans olarak girer.

9. SİMULASYON SONUÇLARI

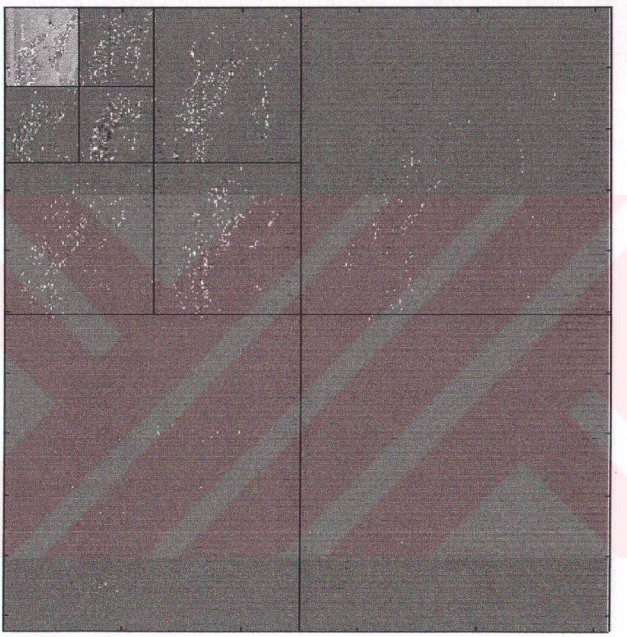
Şekil 9.1. 'de üçüncü seviyeden çok çözünürlülük analizi sonucu görüntünün ayrıştırma seviyeleri görülmektedir. Görüntü ilk seviyede dört değişik bölgeye ayrılır(WL11, WL12, WL13, WL14). WL12, WL13 ve WL14 ayrıntı görüntüleri olup WL11 esas görüntü gibi düşünülüp tekrar alt görüntülere ulaşılır bu işlem üç defa tekrarlanır. Bu işlemelerde Daubechies filtreleri kullanılmıştır. Yapılan uygulama ile yüksek sıkıştırma oranlarına ulaşılır fakat bu durumda yüksek kayıplar ve düşük görüntü kalitesi söz konusu olacaktır.



Şekil 9.1. Çok çözünürlülük analizinde herhangi bir görüntünün üçüncü seviyeden çözünürlülük değişimi

Şekil 9.2. 'de ise Lena görüntüsünün çok çözünürlülük analizi sonucundaki görüntü değişimi görülmektedir. Şekil 9.2. 'den de görüldüğü üzere üçüncü seviyedeki görüntü şekil 9.1. 'deki WL31 sıkışmış görüntüdür ve orijinalinin

1/8 'i kadardır. Tekrar oluşturma safhasında sadece bu görüntü kullanılarak orijinal görüntüye ulaşılabilinir.



Şekil 9.2. Çok çözünürlülük analizinde Lena görüntüsünün çözünürlülük değişimi

Tablo 9.1. Çok çözünürlülük analizi simülasyon sonuçları

GÖRÜNTÜ	QUANTLAMA SEVİYELERİ			SIKIŞTIRMA ORANI	HATA ORANI (%)
	Q1	Q2	Q3		
LENA (512x512)	256	256	256	51.06	40.3
	256	128	64	44.46	22.4
	128	64	32	34.80	15.3
	64	32	16	23.17	9.6
	32	16	8	13.40	6.0
	32	8	8	8.90	4.9
	16	8	8	7.80	4.8
	8	8	8	5.23	4.6
AĞAÇLAR (256x256)	256	256	256	54.79	100
	256	128	64	46.32	87.0
	128	64	32	34.22	54.9
	64	32	16	21.17	36.6
	32	16	8	11.62	25.5
	32	8	8	8.14	19.5
	16	8	8	6.14	17.4
	8	8	8	3.79	15.6
ORMAN (256x256)	256	256	256	46.27	64.0
	256	128	64	27.81	37.7
	128	64	32	16.55	20.9
	64	32	16	9.57	11.8
	32	16	8	5.54	7.7
	32	8	8	4.79	7.5
	16	8	8	3.49	4.6
	8	8	8	2.59	3.4

Tablo 9.2. Altband kodlama simülasyon sonuçları

GÖRÜNTÜ	QUANTLAMA SEVİYESİ		DAUBECHİES FİLTRE KATSAYISI	SIKIŞTIRMA ORANI	HATA ORANI (%)
	Q1	Q2			
LENA (512x512)	256	256	10	4.764	12.21
	256	256	8	4.764	12.26
	128	64	8	3.978	5.7
	128	64	6	3.969	5.75
	32	16	6	3.310	0.63
	32	16	4	3.285	0.62
	8	8	10	2.408	0.15
	8	8	6	2.416	0.15
AĞAÇLAR (256x256)	256	256	10	12.21	21.85
	256	256	8	12.21	21.77
	128	64	8	8.45	9.33
	128	64	6	8.45	9.48
	32	16	6	3.26	2.06
	32	16	4	3.29	2.06
	8	8	10	1.88	0.52
	8	8	6	1.92	0.52
ORMAN (256x256)	256	256	10	4.14	4.86
	256	256	8	4.15	5.17
	128	64	8	3.18	2.29
	128	64	6	3.20	2.35
	32	16	6	1.89	1.09
	32	16	4	1.92	1.09
	8	8	10	1.54	0.15
	8	8	6	1.59	0.16

(a)

(b)

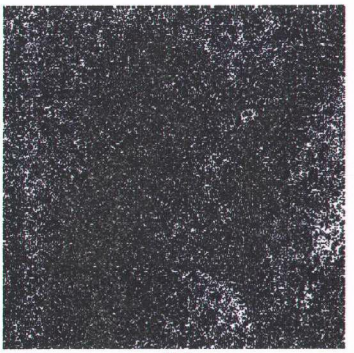
Şekil 9.3. MRA analizi kullanılarak **(a)** 34.8 kat sıkıştırılmış Lena görüntüsü **(b)** Oluşan hata görüntüsü

(a)

(b)

Şekil 9.4. MRA analizi kullanılarak **(a)** 51.05 kat sıkıştırılmış Lena görüntüsü **(b)** Oluşan hata görüntüsü

(a)



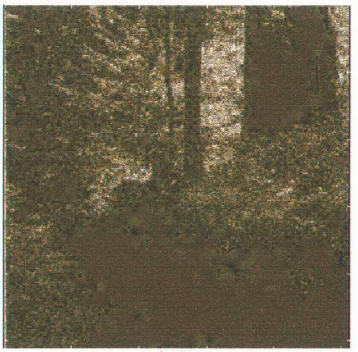
(b)

Şekil 9.5. MRA analizi kullanılarak **(a)** 23.16 kat sıkıştırılmış Lena görüntüsü **(b)** Oluşan hata görüntüsü

(a)

(b)

Şekil 9.6. MRA analizi kullanılarak **(a)** 11.62 kat sıkıştırılmış ağaç görüntüsü **(b)** Oluşan hata görüntüsü

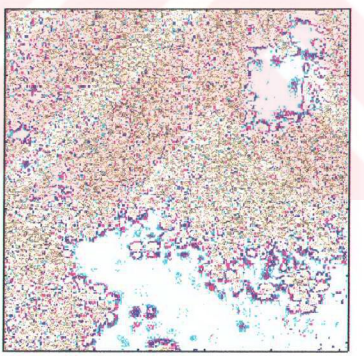


(a)

(b)

Şekil 9.7. MRA analizi kullanılarak **(a)** 9.57 kat sıkıştırılmış orman görüntüsü **(b)** Oluşan hata görüntüsü

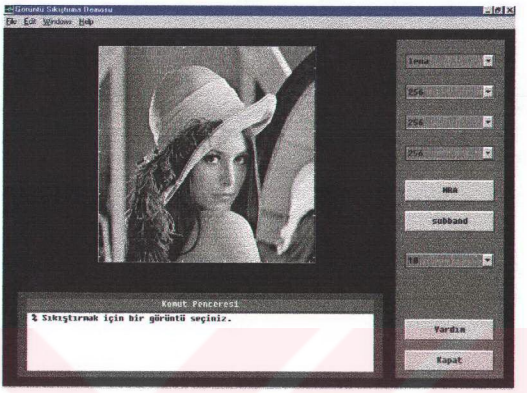
(a)



(b)

Şekil 9.8. MRA analizi kullanılarak **(a)** 16.55 kat sıkıştırılmış orman görüntüsü **(b)** Oluşan hata görüntüsü

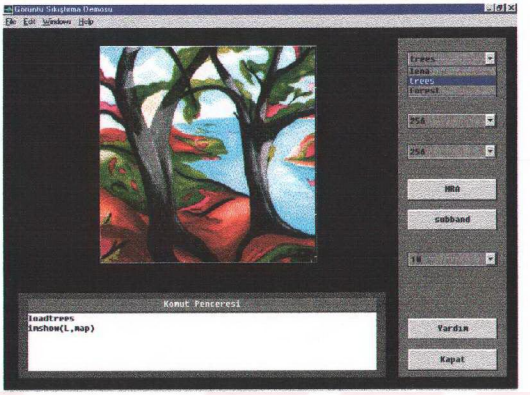
9.2.Sıkıştırma Programının Tanıtılması



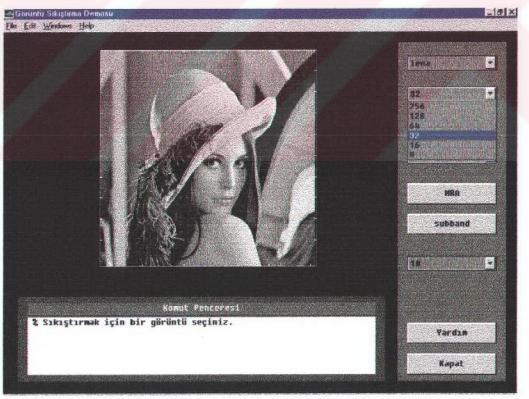
Şekil 9.9. Sıkıştırma programının genel görünümü

Şekil 9.9 'de Matlab 4.0 programlama dili kullanılarak yapılmış olan görüntü sıkıştırma programı görülmektedir.

Pencere içinde açılan, görüntü penceresi içindeki sıkıştırılması istenen görüntü, Lena yazılı menüdeki oka fare ile basılarak seçilir. Burada üç adet görüntü bulunmaktadır: Lena(512x512), trees(256x256) ve forest(256x256). Seçim işlemi şekil 9.10 'daki gibi yapılır. Burada trees görüntüsünün nasıl yüklendiği örnek olarak verilmiştir. Eğer herhangi bir seçim yapılmaz ise program başlangıçta direkt olarak yüklediği Lena görüntüsünü yüklenmiş gibi kabul eder.



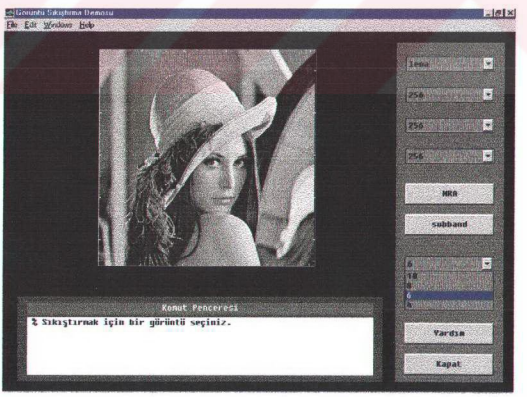
Şekil 9.10. Sıkıştırılacak görüntünün seçimi



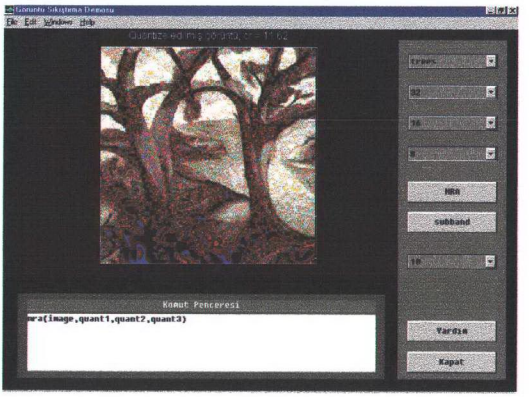
Şekil 9.11. Quantlama seviyelerinin seçimi

Daha sonra nicelikleştirme(quantlama) seviyelerinin seçimi yapılır. Bu seçimi yapmak için menüden 256, 128, 64, 32, 16 ve 8 seviyelerinden biri seçilir. Asıl sıkıştırmayı sağlayan bu işlemdir. Üç menü MRA işlemi için ilk iki menü ise altbant kodlama içindir. Ama unutulmaması gereken şey işlemin kayıplı ve kayıpsız olmasına sebep olan bu işlemdir. Altbant kodlama işlemi için ayrıca filtre derecesi(10, 8, 6 ve 4) seçimi şekil 9.12 'deki gibi yapılmalıdır.

Tüm bu işlemler yapıldıktan sonra MRA(çok çözünürlülük analizi) ve altbant kodlama analizi seçimi için MRA veya Subband yazılı butonlardan birisine basılarak sıkıştırma işlemi yapılır. Şekil 9.13. 'deki gibi tekrar oluşturulmuş görüntü ve sıkıştırma oranı ekranda görülür. Aynı zamanda bu işlemlerin yapılması için Matlab komut satırında yürütülmesi gereken komut görüntü penceresinin altındaki pencerede belirir.

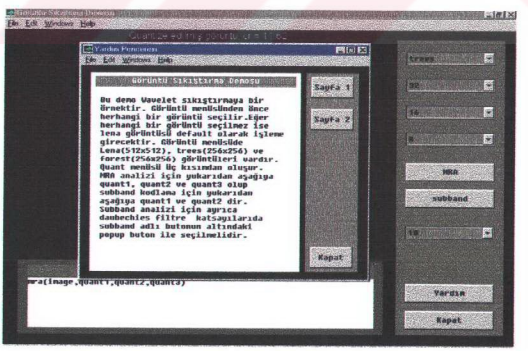


Şekil 9.12. Filtre derecesinin seçimi



Şekil 9.13. Sıkıştırma sonrası yeniden oluşturulmuş görüntünün pencerede görüntülenmesi

Eğer kullanıcının bazı problemleri olur ise yardım butonuna basarak şekil 9.14. 'de görülen gerekli yardım sayfalarına ulaşabilir.



Şekil 9.14. Yardım penceresinin açılması

10. SONUÇ

Günümüzde herhangi bir verinin iletiminde, işlenmesinde ve saklanmasında hız ve zaman oldukça önemlidir. Veri değişik yollarla saklanabilir. Herhangi bir işleme tabi tutulmadan verinin saklanması zaman alacak hem de gereğinden fazla yer işgal edecektir. Bu gibi durumlardan kaçınmak için bir veriye sıkıştırma işlemi uygulanmalıdır. Sıkıştırmadan sonra verinin tuttuğu yer miktarı azalacak, iletilmesi ve kaydedilmesi daha kısa bir süre içinde gerçekleştirilecektir.

Görüntü sıkıştırılması özellikle iletimlerinde oldukça önemlidir. (Videofon, video konferans v.b.) Bu uygulamalarda özel standartlar vardır. Örneğin videofon uygulamalarında kullanılacak sıkıştırma aracının gerçek zamanda işlev görmesi yani minimum gecikme ile veriyi sıkıştırıp açması gerekmektedir. Bunun için H.261 görüntü sıkıştırma standardı geliştirilmiştir.

Görüntü sıkıştırılmaya başlanırken, dönüşüm tabanlı metotlarda önce görüntü piksellerdeki veri fazlalıklarından arındırılır. Bunu sağlamak için görüntü bir matris gibi düşünülüp iki boyutlu bir dönüşüm uygulanır. Bu işlem aynı zamanda bir nevi filtrelemedir. Daha sonra görüntü Huffmann, aritmetik kodlama ve vektör nicelikleştirme işlemlerinden birinden geçirilerek bir sıkıştırma sağlanır. Bu sıkıştırma, kayıpsız bir sıkıştırmadır. Kayıpsız sıkıştırmadan kasıt, kodlamadan önceki görüntü ile kodlamadan sonraki görüntü bir biri ile aynıdır. Kayıplı sıkıştırma da ise işin içine nicelikleştirme işlemi girer ki bu piksellerin değerini bir miktar değiştirir ve bu nedenle kayıplar söz konusu olur.

Wavelet dönüşümü, veriyi değişik frekans bölgelerine ayırmasından dolayı literatürde önemli bir yer tutar. Bu çalışmada Wavelet dönüşümü kullanılarak bir sıkıştırma işlemi sağlanmaya çalışıldı. Wavelet dönüşümünün kendisi kayıpsızdır. Ama yüksek oranlarda sıkıştırma sağlanması istendiğinde yüksek değerli nicelikleştirme seviyeleri kullanıldığı için kayıplar söz konusudur.

Aslında wavelet dönüşümü vektör nicelikleştirme(Vector Quantization) veya huffman kodlaması ile beraber kullanılırsa mükemmel sonuçlar verir. Bu nedenle Wavelet görüntü sıkıştırma herhangi bir veri sıkıştırma algoritması ile desteklenmelidir.

Yapılan uygulamada yüksek sıkıştırma oranlarına ulaşılmıştır. Ama sıkıştırma oranı artıkça görüntü kalitesinde ve doğruluğunda düşüşler gözlenmiştir. Bu durum oldukça doğaldır. Çünkü verinin sıkıştırılması için herhangi bir veri kodlama tekniği kullanılmamaktadır. Bu nedenle yüksek sıkıştırma oranlarında yüksek kayıplar söz konusu olmuştur.

Ayrıca değişik sıkıştırma oranları için hata görüntüleri çıkartılarak görsel olarak sıkıştırma oranı ile doğruluğun değişimi gözlenmeye çalışılmıştır.

EK 1: Görüntü Sıkıştırma Programı

```
function compdemo(action,s)

if nargin<1,
    action='initialize';
end;

if strcmp(action,'initialize'),
    figNumber=figure( ...
        'Name','Görüntü Sıkıştırma Demosu', ...
        'NumberTitle','off');

%==================================
% Görüntü axis ayarları
axes( ...
        'Units','normalized', ...
        'Position',[.10 0.35 0.60 0.6], ...
        'XTick',[],'YTick',[], ...
        'Box','on');
    set(figNumber,'defaultaxesposition',[.1 .35 0.6 0.6])

%==================================
% Komut Penceresi Ayarları
    top=.25;
    left=0.05;
    right=0.725;
    bottom=0.05;
    labelHt=0.035;
    spacing=0.005;

%==================================
% Birinci Mini Komut Penceresi
    frmBorder=0.02;
    frmPos=[left-frmBorder bottom-frmBorder ...
     (right-left)+2*frmBorder (top-bottom)+2*frmBorder];
    uicontrol( ...
        'Style','frame', ...
        'Units','normalized', ...
        'Position',frmPos, ...
        'BackgroundColor',[0.3 0.3 0.3]);

% Text Etiketi
   labelPos=[left top-labelHt (right-left) labelHt];
   uicontrol( ...
     'Style','text', ...
        'Units','normalized', ...
        'Position',labelPos, ...
        'BackgroundColor',[0.4 0.4 0.4], ...
```

```
      'ForegroundColor',[1 1 1], ...
    'String','Komut Penceresi');

%  Yazılabilir Text Alanı
   txtPos=[left bottom (right-left) top-bottom-labelHt-
spacing];
   txtHndl=uicontrol( ...
     'Style','edit', ...
        'Units','normalized', ...
        'Max',5, ...
        'BackgroundColor',[1  1  1], ...
        'Position',txtPos, ...
        'String',str2mat(' % Sıkıştırmak için bir görüntü
seçiniz.'));

set(gcf,'UserData',txtHndl);

%=====================================
% Bütün Pencere ve Menü Ayarları
   labelColor=[.5 .5 .5];
   yInitPos=0.90;
   menutop=0.95;
   top=0.75;
   left=0.785;
   btnWid=0.175;
   btnHt=0.06;

spacing=0.025;

%=====================================
   frmBorder=0.02;
   yPos=0.05-frmBorder;
   frmPos=[left-frmBorder yPos btnWid+2*frmBorder
0.9+2*frmBorder];
   h=uicontrol( ...
       'Style','frame', ...
       'Units','normalized', ...
       'Position',frmPos, ...
       'BackgroundColor',[0.4 0.4 0.4]);

%=====================================
%  Görüntü Menüsü
    menuNumber=1;
    yPos=menutop-(menuNumber-1)*(btnHt+spacing);
    labelStr='lena|trees|forest';
    callbackStr='compdemo(''image'');';

% Temel Buton Bilgileri
    btnPos=[left yPos-btnHt btnWid btnHt];
```

```
    imageHndl=uicontrol( ...
        'Style','popupmenu', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
        'Interruptible','yes', ...
        'Callback',callbackStr);

%====================================
% The Quant1 Menu
    menuNumber=2;
    yPos=menutop-(menuNumber-1)*(btnHt+spacing);
    labelStr='256|128|64|32|16|8';
    callbackStr='compdemo(''Quantization'')';

% Temel Buton Bilgileri
    btnPos=[left yPos-btnHt btnWid btnHt];
    quantHndl1=uicontrol( ...
        'Style','popupmenu', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
        'Interruptible','yes', ...
        'Callback',callbackStr);

% The Quant2 Menu
    menuNumber=3;
    yPos=menutop-(menuNumber-1)*(btnHt+spacing);
    labelStr='256|128|64|32|16|8';
    callbackStr='compdemo(''Quantization'')';

% Temel Buton Bilgileri
    btnPos=[left yPos-btnHt btnWid btnHt];
    quantHndl2=uicontrol( ...
        'Style','popupmenu', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
        'Interruptible','yes', ...
        'Callback',callbackStr);

%============================================
% The Quant3 Menu
    menuNumber=4;
    yPos=menutop-(menuNumber-1)*(btnHt+spacing);
    labelStr='256|128|64|32|16|8';
    callbackStr='compdemo(''Quantization'')';

% Temel Buton Bilgileri
```

```
    btnPos=[left yPos-btnHt btnWid btnHt];
    quantHndl3=uicontrol( ...
        'Style','popupmenu', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
        'Interruptible','yes', ...
        'Callback',callbackStr);


%=====================================
% Yardım Butonu
    labelStr='Yardım';
    callbackStr='compdemo(''yardım'')';
    helpHndl=uicontrol( ...
        'Style','pushbutton', ...
        'Units','normalized', ...
        'Position',[left bottom+btnHt+spacing btnWid
btnHt], ...
        'String',labelStr, ...
        'Callback',callbackStr);

 %=====================================
% button 1
    btnNumber=3;
    yPos=top-(btnNumber-1)*(btnHt+spacing);
    labelStr='MRA';
    % Setting userdata to -1 (=stop) will stop the demo.
    callbackStr='compdemo(''MRA'');';

% Generic button information
    btnPos=[left yPos-btnHt btnWid btnHt];
    btn3Hndl=uicontrol( ...
        'Style','pushbutton', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
        'Callback',callbackStr);


 %=====================================
 % button 2
    btnNumber=4;
    yPos=top-(btnNumber-1)*(btnHt+spacing);
    labelStr='subband';
    % Setting userdata to -1 (=stop) will stop the demo.
    callbackStr='compdemo(''subband'');';

% Temel Buton Bilgileri
```

```
    btnPos=[left yPos-btnHt btnWid btnHt];
    btn4Hndl=uicontrol( ...
        'Style','pushbutton', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
        'Callback',callbackStr);


%=====================================
% The daub  number Menu
    menuNumber=7.5;
    yPos=menutop-(menuNumber-1)*(btnHt+spacing);
    labelStr='10|8|6|4';
    callbackStr='compdemo(''katsayılar'');';

% Temel Buton Bilgileri
    btnPos=[left yPos-btnHt btnWid btnHt];
    daubHndl=uicontrol( ...
        'Style','popupmenu', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
        'Interruptible','yes', ...
        'Callback',callbackStr);
%=====================================
% Kapat butonu
    labelStr='Kapat';
    callbackStr='close(gcf)';
    closeHndl=uicontrol( ...
        'Style','pushbutton', ...
        'Units','normalized', ...
        'Position',[left bottom btnWid btnHt], ...
        'String',labelStr, ...
        'Callback',callbackStr);

hndlList=[txtHndl imageHndl quantHndl1 quantHndl2
quantHndl3 btn3Hndl btn4Hndl...
               daubHndl  helpHndl closeHndl];

    set(figNumber, ...
     'Visible','on', ...
     'UserData',hndlList);

    set(gcf,'Pointer','watch');
    drawnow
    load lena
    map1=gray(256);
```

```
    imshow(L,map1);
    set(gcf,'Pointer','arrow');
    return
end

danger = strcmp(get(gcf,'NextPlot'),'replace');

if strcmp(action,'image'),
   if danger,
      set(gcf,'nextplot','add');
      h = get(gcf,'children');
      for i=1:length(h),
        if strcmp(get(h(i),'type'),'axes'), delete(h(i)),
end
      end
    end

axHndl=gca;
hndlList=get(gcf,'Userdata');
txtHndl=hndlList(1);
imageHndl=hndlList(2);
quantHndl1=hndlList(3);
quantHndl2=hndlList(4);
quantHndl3=hndlList(5);
btn3Hndl=hndlList(6);
btn4Hndl=hndlList(7);
btn5Hndl=hndlList(8);
helpHndl=hndlList(9);
closeHndl=hndlList(10);
h=get(axHndl,'children');
set(gcf,'pointer','watch');
if strcmp(get(h(1), 'type'),'image');
X=get(h(1),'cdata');
map=colormap;
v=get(imageHndl,'value');
name=get(imageHndl,'String');
name=deblank(name(v,:));
commentStr=str2mat(['load',name],'imshow(L,map)');
set(txtHndl,'String',commentStr);
set(gcf,'Pointer','watch');
load (name)
if ~exist('map') map=gray(256);end
imshow(L,map)
 set(gcf,'Pointer','arrow');
%    return
end

end
```

```
%Butonların Çalışması
%====================
   if strcmp(action, 'MRA'),
     axHndl=gca;
     hndlList=get(gcf,'Userdata');
     h=get(axHndl,'children');
     set(gcf,'pointer','watch');
     txtHndl=hndlList(1);
     imageHndl=hndlList(2);
     quantHndl1=hndlList(3);
     quantHndl2=hndlList(4);
     quantHndl3=hndlList(5);
     btn3Hndl=hndlList(6);
     btn4Hndl=hndlList(7);
     btn5Hndl=hndlList(8);
     helpHndl=hndlList(9);
     closeHndl=hndlList(10);
     h=get(axHndl,'children');
     commentStr=str2mat('mra(image,quant1,quant2,quant3)');
     set(txtHndl,'String',commentStr),drawnow
     if danger, delete(haxes), end

v1=get(imageHndl,'value');
name1=get(imageHndl,'String');
image=deblank(name1(v1,:));

v2=get(quantHndl1,'value');
name2=get(quantHndl1,'String');
quant1=deblank(name2(v2,:));

v3=get(quantHndl2,'value');
name3=get(quantHndl2,'String');
quant2=deblank(name3(v3,:));

v4=get(quantHndl3,'value');
name4=get(quantHndl3,'String');
quant3=deblank(name4(v4,:));

load(image)

[cL,cr]=demomra(L,quant1, quant2, quant3);
if ~exist('map') map=gray(256);end
imshow(cL,map);
TITLE = [ 'Quantize edilmiş görüntü, cr = ' num2str(cr) ];
title(TITLE);


elseif strcmp(action,'subband'),
axHndl=gca;
```

```
   hndlList=get(gcf,'Userdata');
     h=get(axHndl,'children');
     set(gcf,'pointer','watch');
     txtHndl=hndlList(1);
     imageHndl=hndlList(2);
     quantHndl1=hndlList(3);
     quantHndl2=hndlList(4);
     quantHndl3=hndlList(5);
     btn3Hndl=hndlList(6);
     btn4Hndl=hndlList(7);
     daubHndl=hndlList(8);
     helpHndl=hndlList(9);
     closeHndl=hndlList(10);
     h=get(axHndl,'children');
     commentStr=str2mat('[deg1, deg2, deg3,
           deg4,n]=wvlt2da(x)',...
                'y=wvlt2ds(deg1,deg2,deg3,deg4,n);');
     set(txtHndl,'String',commentStr),drawnow
     if danger, delete(haxes), end

     v2=get(imageHndl,'value');
     name2=get(imageHndl,'String');
     image=deblank(name2(v2,:));

     v1=get(daubHndl,'value');
     name1=get(daubHndl,'String');
     ktsydb=deblank(name1(v1,:));

     v3=get(quantHndl1,'value');
     name3=get(quantHndl1,'String');
     quant1=deblank(name3(v3,:));

     v4=get(quantHndl2,'value');
     name4=get(quantHndl2,'String');
     quant2=deblank(name4(v4,:));

     load(image)
     [deg1, deg2, deg3, deg4,n]=wvlt2da2(L,ktsydb);

     [Lcomp,cr,deg11,deg22,deg33,deg44]=deneme2(deg1,deg2,
     deg3,deg4,L,quant1,quant2);
     y=wvlt2ds2(deg11,deg22,deg33,deg44,n);
     if ~exist('map') map=gray(256);end
     imshow(y,map);
     TITLE = [ 'Quantize edilmiş görüntü, cr = '
     num2str(cr) ];
     title(TITLE);

elseif strcmp(action,'yardım'),
```

```
set(gcf,'pointer','arrow')
ttlStr =get(gcf,'Name');

hlpStr1= ...
   ['                                          '
    ' Bu demo Wavelet sıkıştırmaya bir         '
    ' örnektir. Görüntü menüsünden önce        '
    ' herhangi bir görüntü seçilir.Eğer        '
    ' herhangi bir görüntü seçilmez ise        '
    ' lena görüntüsü default olarak işleme'
    ' girecektir. Görüntü menüsüde             '
    ' Lena(512x512), trees(256x256) ve         '
    ' forest(256x256) görüntüleri vardır.      '
    ' Quant menüsü üç kısımdan oluşur.         '
    ' MRA analizi için yukarıdan aşağıya       '
    ' quant1, quant2 ve quant3 olup            '
    ' subband kodlama için yukarıdan           '
    ' aşağıya quant1 ve quant2 dir.            '
    ' Subband analizi için ayrıca              '
    ' daubechies filtre  katsayılarıda         '
    ' subband adlı butonun altındaki           '
    ' popup buton ile seçilmelidir.           '];

hlpStr2= ...
   [' Daha ayrıntılı bilgi alabilmek için     '
    ' aşağıdaki   .m dosyalarını incelemek'
    ' faydalı olacaktır.                       '
    ' wmai2.m ikwt2.m imcomp2.m  qcp.m         '
    ' daubech.m ktsypn.m ktsygn.m              '
    ' wvlt2da2.m   compratio.m    dwnsmp.m'
    ' upsmp.m     dwnsmp2.m     satflt.m       '
    ' sutflt.m    upsmp2.m      quant.m        '
    ' wvlt2ds2.m                               '
    ' Sıkıştırma değişik seviyelerdeki         '
    ' quantlamaile sağlanır.                   '
    '                                          '
    '                                          '
    '                                          '
    '                                          '
    '                                          '
    '                                          '
    '                                          '
    '                                         '];
myFig = gcf;
helpfun1(ttlStr,hlpStr1,hlpStr2);

   % Protect against gcf changing -- Change close button
behind
   % helpfun's back
```

```
    ch = get(gcf,'ch');
    for i=1:length(ch),
      if strcmp(get(ch(i),'type'),'uicontrol'),
        if strcmp(lower(get(ch(i),'String')),'close'),
          callbackStr = [get(ch(i),'callback') ...
            '; compdemo(''closehelp'',' num2str(myFig)
')'];
          set(ch(i),'callback',callbackStr)
          return
        end
      end
    end
    return

end    % if strcmp(action, ...
set(gcf,'pointer','arrow')
```

KAYNAKLAR:

Bhaskaran V., Konstantinides K., 1996, **Image and video compression standards, Algorithms and architecture,** Kluwer Academic Press, Boston.

Mandal M. K., Panchanathan S., Aboulnasr T., 1993, "Choice of Wavelets for Image Compression", **lectures Notes in Computer Science,** Vol.1133, 239-249.

Graps A., 1995, "An Introduction to Wavelets", **IEEE Computational Science and Engineering,** Vol.2, no:2. 220-228.

Strang G., 1992, "Wavelets", **American Scientist,** Vol.82, 250-255.

Meyer Y., 1992, **Wavelets and Operators,** Cambridge University Press, England.

Ruskai M. B., 1992, **Wavelets and Their Application,** Jones and Burlett Publishers, England.

Gonzalez R. C., Woods R. E., 1993, **Digital Image Processing,** Adison-Wesley Publishing Company, USA.

Daubechies I., 1992, **Ten Lectures on Wavelets,** Society for Industrial and Applied Mathematics, USA.

Kayran A., 1989, **Çok Boyutlu İşaret ve Görüntü İşleme**, İTÜ, İstanbul.