YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FOURIER DÖNÜŞÜMÜ ile DİJİTAL GÖRÜNTÜ İŞLEME

Matematik Müh. Ebru DİNÇSOY

F.B.E Matematik Mühendisliği Anabilim Dalında
Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. Coşkun Güler

Prof Dr Mustafa Sivri

Doç. Dr. Ömer Gök

İSTANBUL, 2003

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

DC	Discrete Components(Ayrık Bileşenler)
DFT	Discrete Fourier Transform(Ayrık Fourier Dönüşümü)
FFT	The Fast Fourier Transform(Hızlı Fourier Dönüşümü)

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Bu tezin gerçekleştirilmesinde desteğini esirgemeyen değerli danışmanım Yrd.Doç.Dr. Coşkun GÜLER'e teşekkür ederim.

Ayrıca yardımlarından dolayı Yrd.Doç.Dr. İbrahim EMİROĞLU'na teşekkürlerimi sunarım.

ÖZET

Siyah-beyaz görüntü ya da basit görüntü,iki boyutlu ışık yoğunluk fonksiyonu $f(x,y)$ ile ifade edilir.Burada x,y uzaysal koordinatlardır ve herhangi bir (x,y) noktasının f değeri görüntünün o noktadaki parlaklığı ile orantılıdır. $f(x,y)$'nin görüntüsü anlamına gelen dijital görüntü,görüntüdeki bir noktayı tanımlayan indekslerden ibaret satır ve sütunlardan oluşmuş bir matris olarak ifade edilebilir.Burada ki uygun matris elemanı değeri o noktanın parlaklık derecesini tanımlar.İşlemlerde esas alınacak olan parlaklık derecesidir ve aynı resmi frekans verilerinde tanımlamak ta mümkündür. Fourier dönüşümü görüntüyü sinüs ve kosinüs bileşenlerine ayırmakta kullanılan önemli bir görüntü işleme aracıdır.Girdi görüntü uzaysal verilerde olduğunda dönüşümün çıktısı,görüntünün Fourier ya da frekans verilerinde ki ifadesidir.Ayrık görüntü verilerinde süreklinin tam tersi olarak Fourier dönüşümünün ayrık formu DFT vardır.FFT ise DFT'nin hesaplamada daha başarılı versiyonudur.

Bu çalışmada,Bölüm2'de bir ve iki sürekli değişkenli durumda Fourier dönüşümleri gösterilmiştir. Bölüm3'te bu kavramlar ayrık form için tanımlanmıştır. Bölüm4'te 2-D Fourier dönüşümünün bazı önemli özellikleri gösterilmiştir.Bölüm5'te Ayrık Fourier Dönüşümü'nü tamamlamak için gereken hesaplamaların sayısını indirmekte kullanılabilir bir FFT algoritması geliştirilmiştir.

Anahtar kelimeler: Görüntü işleme,Fourier dönüşümü,Ayrık Fourier Dönüşümü,Hızlı Fourier Dönüşümü , örnekleme.

ABSTRACT

The term monochrome image or simply image,refers to a two-dimensional light intensity function $f(x,y)$,where x and y denote spatial coordinates and the value of f at any point *(x,y)* is proportional to the brightness(or gray level) of the image at that point. A digital image is an image $f(x,y)$ can be considered a matrix whose row and column indices identify a point in the image and the corresponding matrix element value identifies the gray level at that point.The elements of such a digital image array are called image elements,picture elements or pixels. Only the brightness levels have been considered, but it is possible to represent the same picture in the frequency domain. The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent.. For discrete image data, as opposed to continuous functions, there is a discrete version of the Fourier transform called the DFT. There is another more computationally efficient version of the DFT called the Fast Fourier Transform (FFT).

The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression.

In this study;section 2 introduces the Fourier transforms of one and two continuous variables,section 3 then expresses these concepts in discrete form,section 4 develops and illustrates several important properties of the 2-D Fourier transform,section 5 develops a fast Fourier transform algorithm,which can be used to reduce the number of calculations to a fraction of that required to implement the discrete Fourier transform by direct methods.

Keywords: Image processing,Fourier transforms,Discrete Fourier transforms,Fast Fourier transforms,sampling

1. GİRİŞ

Görüntü işleme yöntemlerine olan ilgi iki temel uygulama alanından kaynaklanmaktadır;insanların yorumlamak zorunda kaldığı resimsel bilgiler ve kendi kendine algılama yapan makineler için ekran bilgilerindeki gelişme.Görüntü işlemede ilk uygulama bunlardan ilkiyle ilgiliydi.1920 başlarında,Londra ile NewYork arasında su altı kablolarıyla gazete resimleri dijital hale getirilerek göderiliyordu.Görüntüyü 15 farklı bölümlere kodlamak 1929 yılında gerçekleşti.35 yıl sonra insansız uzay roketlerinden gelen görüntüler için bilgisayar tekniklerinin kullanılmasına başlandı.(Jet Propulsion labaratuarı-California)

İşlem ilk olarak görüntünün elde edilmesidir(Kamera).İkinci işlem,diğer işlemlerin başarı oranını yükselten ön işlemedir(istenen bölgenin diğer alandan izolasyonu).Üçüncü işlem ayırmadır(Girdi görüntünün bileşenlerine ayrılması).Görüntü işlemede anahtar rolü dönüşüm teorileri üstlenmiştir.Görüntüyü işlemede en geniş uygulama alanına sahip olan teknik,Fourier dönüşümüdür.

2. FOURIER DÖNÜŞÜMÜ

Fourier dönüşümü iki görüntü şeklinde ifade edilebilecek çıktı görüntüsünün kompleks sayı değerlerinde olmasını sağlar(reel ve imajinel ya da mutlak değeri ve faz şeklinde).Görüntü işlemede genelde yalnızca Fourier dönüşümünün büyüklük değeri(mutlak değeri) görüntülenir. Fourier görüntüsünü frekans verilerinden yeniden uzaysal verilere dönüştürmek istenirse,mutlak değerin ve fazın korunduğundan emin olunmalıdır.

Teorik olarak Fourier dönüşümünü incelemek istersek;

$f(x)$,reel x'lerden oluşan sürekli fonksiyon olsun. $f(x)$'in Fourier dönüşümü $\Im\{f(x)\}$ şeklinde gösterilir ve $j=\sqrt{-1}$ olmak üzere;

$$\Im\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)\exp[-j2\pi ux]dx \tag{2.1}$$

şeklindedir. $F(u)$'nun verilmesiyle $f(x)$ ters Fourier dönüşümünden çıkarılabilir.

$$\begin{aligned}\Im^{-1}\{F(u)\} &= f(x)\\ &= \int_{-\infty}^{\infty} F(u)\exp[j2\pi ux]du\end{aligned} \tag{2.2}$$

(2.1) ve (2.2) eşitlikleri fourier dönüşüm çiftidir. $f(x)$ sürekli ve integrallenebilir ve $F(u)$ nun integrallenebilir olduğu durum için vardır.

Konu boyunca $f(x)$ reel olarak kabul edilmiştir.Reel fonksiyonların fourier dönüşümü genelde komplekstir.

$$F(u)=R(u)+jI(u) \tag{2.3}$$

$R(u)$ ve $I(u)$, $F(u)$'nun sırasıyla reel ve imajinel bileşenleridir.(2.3)'ün exponansiyel formu;

$$F(u)=|F(u)|e^{i\phi(u)} \tag{2.4}$$

şeklindedir.Burada;

$$|F(u)|=\left[R^2(u)+I^2(u)\right]^{1/2} \tag{2.5}$$

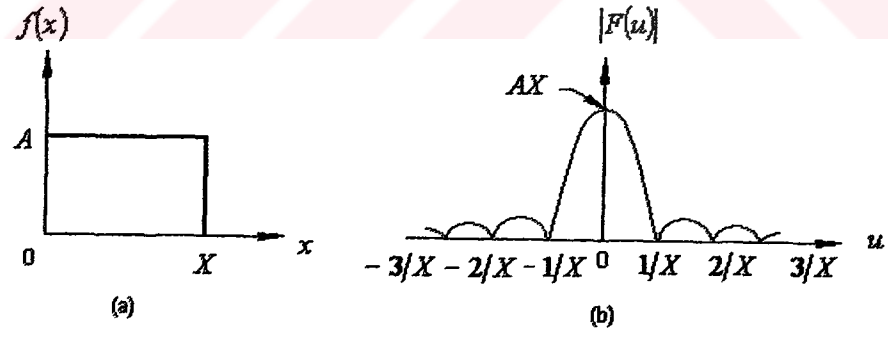
$$\phi(u)=\tan^{-1}\left[\frac{I(u)}{R(u)}\right] \tag{2.6}$$

$|F(u)|$; $f(x)$'in Fourier spektrumu, $\phi(u)$ ise faz açısı olarak adlandırılır.Spektrumun karesi $f(x)$'in güç spektrumu olarak isimlendirilir.Bu değer spektral yoğunluk olarakta geçer.

$$P(u)=|F(u)|^2=R^2(u)+I^2(u) \tag{2.7}$$

Fourier dönüşümünde geçen değişken u;frekans değişkenidir.Bu isim exponansiyelin tanımından gelir.

$$\exp[-j2\pi ux]=\cos 2\pi ux - j\sin 2\pi ux \tag{2.8}$$



Şekil 2.1 $f(x)$ fonksiyonu ve Fourier dönüşümü

ÖRNEK: Şekil 2.1 (a)'daki fonksiyonu ele alalım.Fourier dönüşümü (2.1)'den;

$$F\{f(x)\} = F(u) = \int_{-\infty}^{+\infty} f(x)\exp[-j2\pi ux]dx$$

$$= \int_{0}^{x} A\exp[-j2\pi ux]dx$$

$$= \frac{-A}{j2\pi u}\left[e^{-j2\pi ux}\right]_0^x = \frac{-A}{j2\pi u}\left[e^{-j2\pi ux} - 1\right]$$

$$= \frac{A}{j2\pi u}\left[e^{+j\pi ux} - e^{-j\pi ux}\right]e^{-j\pi ux}$$

$$= \frac{A}{\pi u}\sin(\pi ux)e^{-j\pi ux}$$

kompleks fonksiyondur.Fourier spektrumu;

$$|F(u)| = \left|\frac{A}{\pi u}\right|\left|\sin(\pi ux)\right|\left|e^{-j\pi ux}\right|$$

$$= Ax\left|\frac{\sin(\pi ux)}{(\pi ux)}\right|$$

Şekil 2.1(b), $|F(u)|$'yu göstermektedir.

Fourier dönüşümü iki değişkenli $f(x,y)$ fonksiyonuna kolayca genişletilebilir.Eğer $f(x,y)$ sürekli ve integrallenebilir, $F(u,v)$ integrallenebilir ise Fourier dönüşüm çifti;

$$\Im\{f(x,y)\} = F(u,v) = \int_{-\infty}^{+\infty}\int f(x,y)\exp[-j2\pi(ux+vy)]dxdy \qquad (2.9)$$

$$\Im^{-1}\{f(u,v)\} = f(x,y) = \int_{-\infty}^{+\infty}\int F(u,v)\exp[+j2\pi(ux+vy)]dudv \qquad (2.10)$$

olur.Burada u ve v frekans değişkenleridir.

Bir boyutta olduğu gibi,Fourier spektrumu ,fazı ve güç spektrumu sırasıyla;

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2} \qquad (2.11)$$

$$\phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right] \tag{2.12}$$

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v) \tag{2.13}$$

ÖRNEK: Şekil 2.2(a) da gösterilen fonksiyonun Fourier dönüşümü;

$$F(u,v) = \int_{-\infty}^{+\infty}\int f(x,y)\exp[-j2\pi(ux+vy)]dxdy$$

$$= A\int_0^X \exp[-j2\pi ux]dx \int_0^Y \exp[-j2\pi vy]dy$$

$$= A\left[\frac{e^{-j2\pi ux}}{-j2\pi u}\right]_0^X \left[\frac{e^{-j2\pi vy}}{-j2\pi v}\right]_0^Y$$

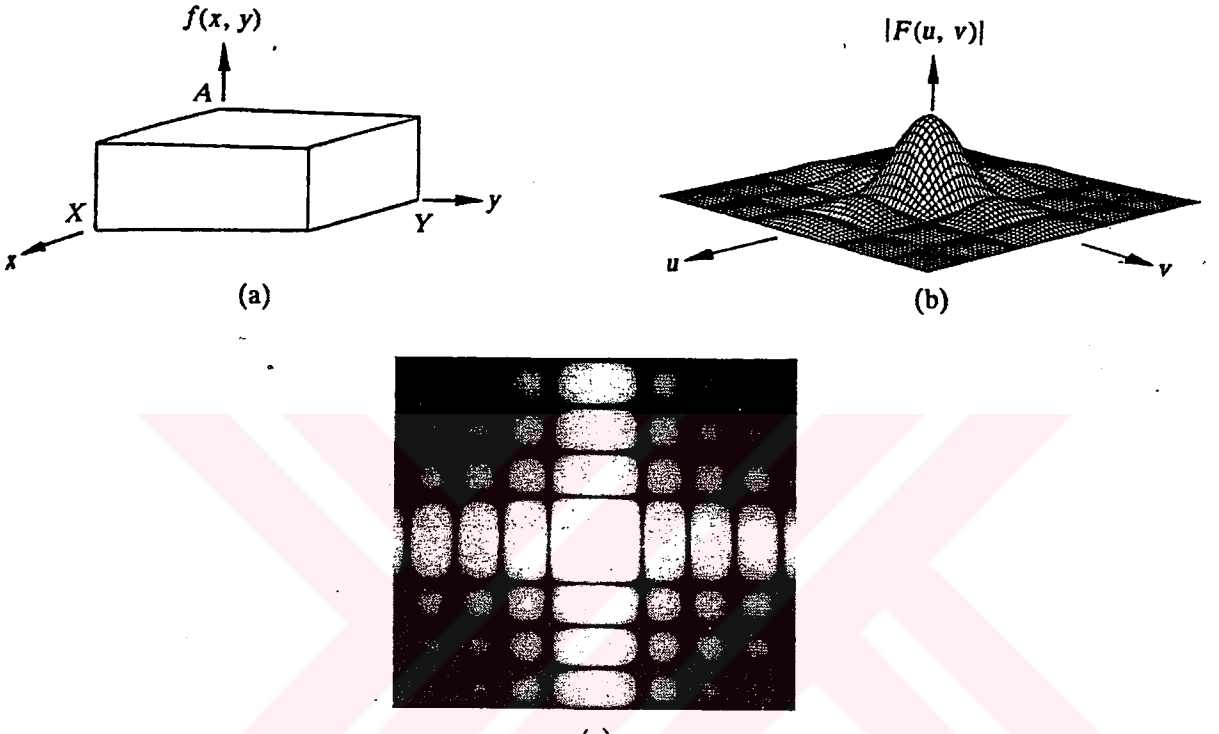
$$= \frac{A}{-j2\pi u}\left[e^{-j2\pi uX} - 1\right]\frac{1}{-j2\pi v}\left[e^{-j2\pi vY} - 1\right]$$

$$= AXY\left[\frac{\sin(\pi uX)e^{-j\pi uX}}{(\pi uX)}\right]\left[\frac{\sin(\pi vY)e^{-j\pi vY}}{(\pi vY)}\right]$$

Spektrum ise;

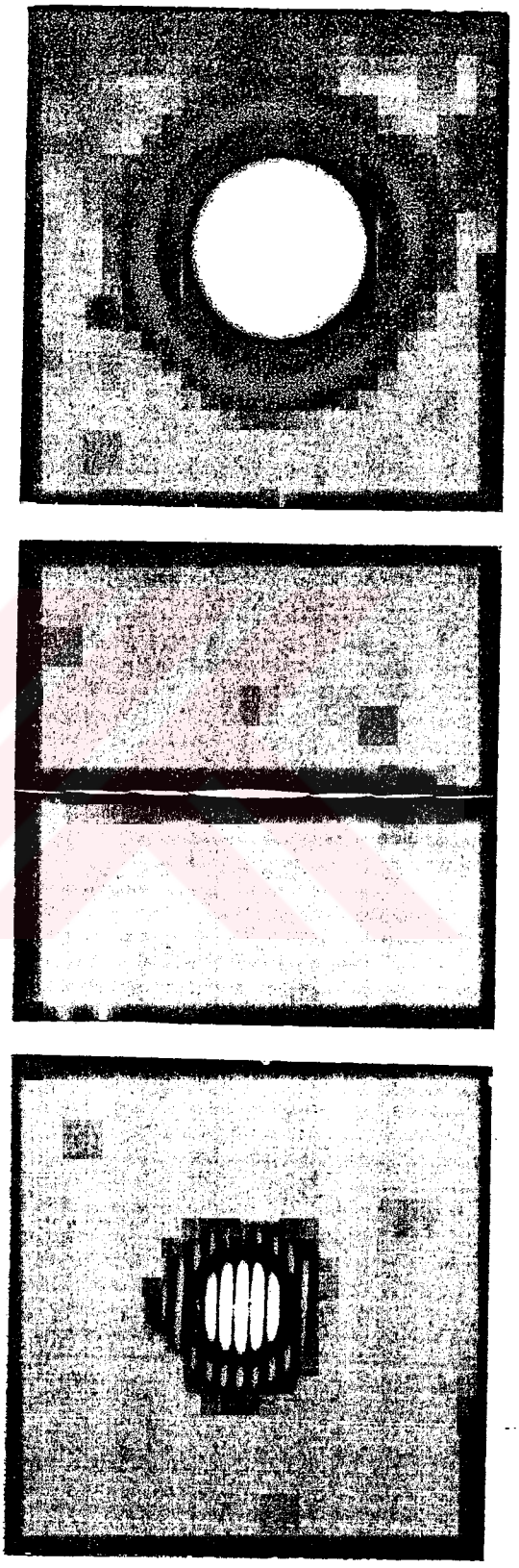
$$|F(u,v)| = AXY\left|\frac{\sin(\pi ux)}{(\pi ux)}\right|\left|\frac{\sin(\pi vy)}{(\pi vy)}\right|$$

şeklinde olacaktır.



Şekil 2.2 (a)2-D fonksiyonu;(b)Fourier spektrumu;(c)spektrumum yoğunluk fonksiyonu olarak görüntülenmesi

Şekil 2.2(b) ;fonksiyonun 3 –D de spektrumunu gösterir.Şekil 2.2(c) yoğunluk fonksiyonunun spektrumunu gösterir.Burada parlaklık $|F(u,v)|$'nin boyutlarıyla orantılıdır.Şekil 2.3;2-D de diğer bir örnektir.



Şekil 2.3 2-D fonksiyonlar ve Fourier spektrumları

3. AYRIK FOURIER DÖNÜŞÜMÜ

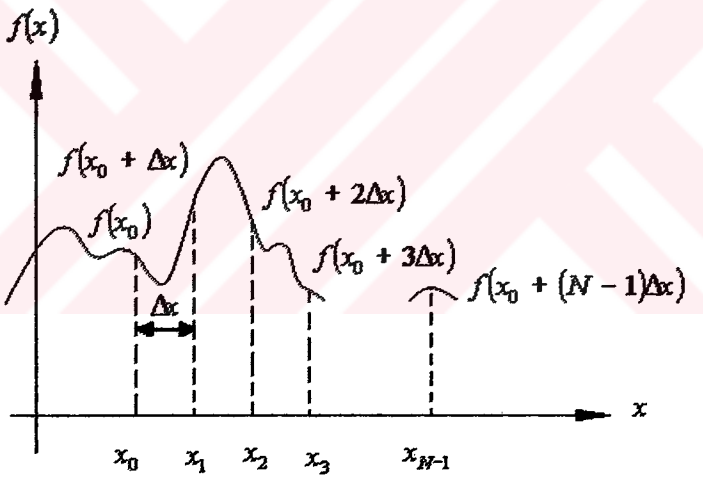
$f(x)$ sürekil fonksiyonunu

$$\{f(x_0), f(x_0 + \Delta x), f(x_0 + 2\Delta x), ..., f(x_0 + [N-1]\Delta x)\}$$

dizisine ayrıştırdığımızı farzedelim.Burada Şekil 3.1'te gösterildiği gibi ΔX birim aralıklı N örnek alınmıştır.Daha sonraki gelişmelerde X'in ayrık ya da sürekli değerde olması konunun bağlamına uygun olarak seçilecektir. X' i ayrık $0,1,2.....,N-1$ değerlerinde tanımladığımızda aşağıdaki bağıntı geçerli olacaktır.

$$f(x) = f(x_0 + x\Delta x) \tag{3.1}$$

Başka bir deyişle $\{f(0), f(1), f(2), ..., f(N-1)\}$ dizisi uygun sürekli fonksiyondan herhangi N boşluklu örneklerle yazılabilir.



Şekil 3.1 Sürekli fonksiyonun örneklenmesi

Yukarıda ki tanımlamayı hatırlayarak örnekli fonksiyonlarda uygulanan ayrık Fourier dönüşüm çifti, $u = 0,1,2.....,N-1$ ve $x = 0,1,2.....,N-1$ için

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\exp[-j2\pi ux/N] \tag{3.2}$$

$$f(x) = \sum_{u=0}^{N-1} F(u)\exp[j2\pi ux/N] \tag{3.3}$$

Ayrık Fourier dönüşümündeki (3.2 eşitliğindeki) $u = 0,1,2.....,N-1$ değerleri; $0, \Delta u, 2\Delta u, ...(N-1)\Delta u$ değerlerindeki sürekli dönüşümün örneklerine uygundur.Başka bir deyişle $F(u); F(u\Delta u)$ şeklinde gösterilebilir. $F(u)$'nun örneklerinin frekens ekseninin orjininden başladığı durum dışında,bu notasyon ayrık $f(x)$ için kullanılanla benzerdir. Δu ve Δx terimleri arasında aşağıda ifade edilen ilişki vardır.

$$\Delta u = \frac{1}{N\Delta x} \tag{3.4}$$

şeklinde tanımlanır.

İki değişkenli durum için ayrık fourier dönüşüm çifti, $u = 0,1,2.....,M-1$ ve $v = 0,1,2,....,N-1$ için aşağıdaki gibidir;

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp[-j2\pi(ux/M + vy/N)] \tag{3.5}$$

$x = 0,1,2....., M-1$ ve $y = 0,1,2,....., N-1$ için;

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp[j2\pi(ux/M + vy/N)] \tag{3.6}$$

Görüntü kare dizide örneklenirse, $M = N$ ve $u,v = 0,1,2,....,N-1$ için;

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\exp[-j2\pi(ux + vy/N)] \tag{3.7}$$

Burada $f(x,y)$,uzaysal verilerdeki görüntüdür,exp. terim ise fouirer uzayındaki herbir $F(u,v)$ noktasına uygun ana fonksiyondur.Yukarıdaki eşitlik;her $F(u,v)$ noktasının değerinin,uzaysal görüntünün,uygun temel fonksiyon ile çarpılıp,çıkan sonuçların toplanması şeklinde elde edileceğini ifade etmektedir.

Ana fonksiyonlar artan frekanslı sinus ve cosinus dalgalarıdır.Örneğin $F(0,0)$ ortalama parlaklığa uygun görüntünün ayrık bileşenini ifade ederken, $F(N-1, N-1)$;en yüksek frekansı gösterir.

Aynı şekilde,Fourier görüntüsü tekrar uzaysal alana dönüştürülebilir.

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)\exp[j2\pi(ux + vy/N)] \tag{3.8}$$

yukarıdaki işlemin sonucunu elde etmek için Fourier dönüşümünün bölünebilirlik özelliğinden(bknz. Bölüm 4);

$$F(u,v) = \sum_{y=0}^{N-1} P(u,y)\exp(-j2\pi vy/N) \tag{3.9}$$

$$P(u,y) = \frac{1}{N}\sum_{x=0}^{N-1} f(x,y)\exp(-j2\pi ux/N) \tag{3.10}$$

Bu formüllerin kullanımında,ilk olarak uzaysal verilerdeki görüntüden N bir-boyutlu Fourier dönüşümü ile ara görüntü elde edilir.Bu ara görüntüye,son görüntüyü elde etmek için,tekrar N bir- boyutlu Fourier dönüşümü uygulanır.Bu şekildeki iki-boyutlu Fourier dönüşümü,2N bir-boyutlu Fourier dönüşümüne dönüştürülerek işlemlerde azalma sağlanmıştır.

Uygulamaların çoğunda DC-değeri $F(0,0)$ görüntünün merkezi olacak şekilde Fourier görüntüsünün yeri değiştirilir.

4. İKİ BOYUTLU FOURIER DÖNÜŞÜMÜNÜN BAZI ÖZELLİKLERİ

Tanım 4.1 (pixel):Görüntü işlemede ilk olarak ele alınacak görüntünün bilgisayar programına adapte edilebilecek şekilde uygun bir formda saklanması gerekir.Bunu yapmanın en kolay yolu görüntüyü ayrık (genelde küçük) hücrelere ayırmaktır.Bu hücreler pixel olarak adlandırılır.Genelde görüntü pixellerden oluşmuş dikdörtgen bir ızgara şeklinde parçalanır.Herbir pixel de küçük bir dikdörtgendir.Bu işlem gerçekleştirildiğinde her pixel,kendi rengini temsil eden pixel değeri ile ifade edilir.Bütün bir pixelin tek bir renkte olduğu varsayılır,böylece görüntü parçalanmadan önceki herhangi bir renk varyasyonu kaybolur.Bu ayrılma elemanları gözle görülemez.

Tanım 4.2 (Gri-seviyeli Görüntü): Yalnızca gri ile renklendirilmiş görüntüdür.Genelde gri renkteki yoğunluk 8-bit tamsayı değerinde saklanır.(siyahtan beyaza mümkün olan 256 çeşit gölgeleme sayısı).

Tanım 4.3 (İkili Görüntü): Pixelleri yalnızca iki mümkün yoğunluk değerine sahip görüntülerdir.Siyah ve beyaz olarak görüntülenirler.Sayısal olarak 0 siyah için,1 yada 255 beyaz için kullanılır.İkili görüntü genelde görüntüdeki bir objeyi arka plandan ayırmak için kullanılır.Objenin rengi 'ön plan rengi' olarak adlandırılır ve genelde beyazdır.Kalan siyah kısım ise arka plandır.

Tanım 4.4 (Pixel Değerleri):Görüntüyü temsil eden pixellerin,parlaklık derecesini ve hangi renk olması gerektiğini gösteren değerler pixel değerleridir.İkili görüntünün avantajı pixel değeri ön ve arka plan için 1- bittir.Gri seviyeli görüntü için pixel değeri ise ,pixelin parlaklığını temsil eden tek sayıdır.En genel *pixel formatı* 0 dan 255'e mümkün olan değerler bölgesini veren 8-bit tamsayı olarak bu değerin saklandığı yer olan *byte görüntüdür.* 0 siyah,255 beyaz ve aradaki sayılar grinin farklı tonları olarak alınır.

Renkli görüntüler;kırmızı yeşil ve mavi bileşenlerine ayrılır ve her pixel değeri üç sayının vektörü olarak ifade edilir.

Bir görüntünün dinamik alanı,herbir pixel değeri kendisinin logaritmik değeri ile yer değiştirmesiyle sıkıştırılabilir.Bu düşük yoğunluk pixel değerlerinin yükseltilmesinde etkilidir.Dinamik kısmın görüntülenmek için fazla büyük olduğu uygulamalarda bu yöntem kullanışlı olabilir.Fonksiyonun görüntülenmesinde oluşan zorluğu gideren yöntem;

$$D(u,v) = c\log[1 + |F(u,v)|] \tag{4.1}$$

ifadesinin $F(u,v)$ yerine yazılmasıdır,*Logaritmik dönüşüm* olarak adlandırılır.Logaritma fonksiyonu istenen sıkıştırmayı sağlamaktadır.c sabiti,maksimum çıktı değeri 255 (8-bit formatı sağlayacak şekilde)olacak şekilde seçilir.Bunun anlamı R girdi görüntüdeki en büyük değer olduğunda,c değeri aşağıdaki gibidir.

$$c = \frac{255}{\log(1 + |R|)}$$

Dinamik alan sıkıştırmasının en genel uygulaması,Fourier dönüşümünün görüntülenmesidir.

Yüksek pixel değerlerinde bilgi kayıplarına neden olan düşük pixel değerlerini yükseltmek istediğimizde,logaritmik dönüşüm uygun olacaktır.Örneğin fotoğraftaki adam;

Şekil 4.1 Örnek görüntü

parlak bir arka planda görüntülenmiştir.Yüzündeki gri seviyeli görüntü küçük pixel değerlerinde bölgelerden oluşsun diye film materyalinin dinamik alanı çok küçüktür.Daha büyük değerler sıkıştırılırken,logaritmik dönüşüm bunları daha geniş alana yayar.Sonuç aşağıda görüldüğü gibidir.

Şekil 4.2 Örnek görüntünün Logaritmik dönüşümü

Diğer yandan aşağıdaki görüntüye dönüşümün uygulanması;

Şekil 4.3 Örnek görüntü

pek uygun değildir,çünkü detayların çoğu yüksek pixel değerleri içermektedir.Uygulama sonucu;

Şekil 4.4 Örnek görüntünün Logaritmik dönüşümü

birçok bilginin yok olduğu görülür.

4.1 Bölünebilme

(3.6) ve (3.7) eşitliklerinden oluşan ayrık Fourier dönüşümü bölünmüş formda ifade edilebilir. $u, v = 0,1,\ldots., N-1$ için;

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\exp[-j2\pi ux/N]\sum_{y=0}^{N-1}f(x,y)\exp[-j2\pi vy/N] \tag{4.2}$$

$x, y = 0,1,\ldots., N-1$ için;

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\exp[j2\pi ux/N]\sum_{v=0}^{N-1}F(u,v)\exp[j2\pi vy/N] \tag{4.3}$$

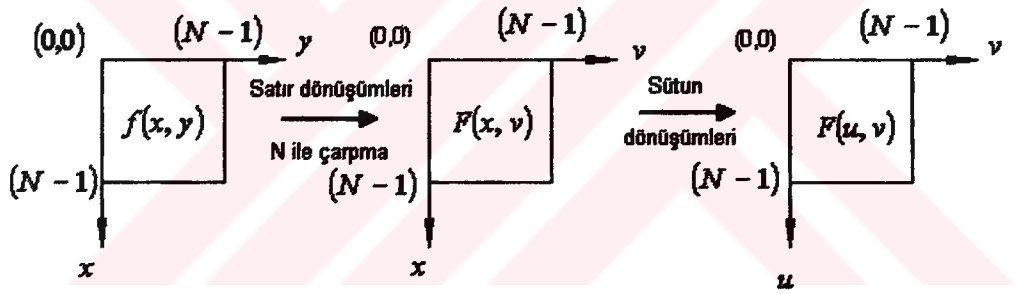
Ayrılabilirlik özelliğinin en temel avantajı $F(u,v)$ ya da $f(x,y)$'nin 1-D Fourier dönüşümü yada ters dönüşümünden iki adımda birbirini izleyen uygulamalarla elde edilebilmesidir.Bu avantaj (4.2) eşitliğinin aşağıdaki formda yazılmasıyla gösterilir.

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1} F(x,v)\exp[-j2\pi ux/N] \tag{4.4}$$

burada;

$$F(x,v) = N\left[\frac{1}{N}\sum_{y=0}^{N-1} f(x,y)\exp[-j2\pi vy/N]\right] \tag{4.5}$$

Herbir x değeri için,(4.5)'teki parantez içi ifade ; frekans değerleri $v = 0,1,....,N-1$ olan bir 1-D dönüşümüdür. Bu yüzden 2-D fonksiyonu olan $F(x,v)$; $f(x,y)$'nin her satırı boyunca dönüşüm alınıp,çıkan sonuç N ile çarpılarak elde edilebilir.İstenen sonuç $F(u,v)$,(4.4)'te gösterildiği gibi $F(x,v)$'nin her sütunu botunca dönüşüm alarak elde edilir.Bu işlemler Şekil 4.5 de özetlenmiştir.



Şekil 4.5 1-D dönüşümler serisi olarak 2-D Fourier dönüşümünün hesaplanması

Aynı sonuç $f(x,y)$'nin sütunları boyunca birinci dönüşüm daha sonra çıkan sonucun satırları boyunca dönüşüm uygulanarak elde edilebilir. (Ballard,1982)

4.2 Periyodik olma ve Conjugate simetri

Ayrık Fourier dönüşümü ve ters dönüşümü periyodiktir ve periyodu N dir

$$F(u,v) = F(u+N,v) = F(u,v+N) = F(u+N,v+N) \tag{4.6}$$

Bu özelliğin geçerliliği $(u+N)$ ve $(v+N)$'nin (3.6)'da yerleştirilmesiyle ispatlanabilir.(4.6) eşitliği u ve v'nin sonsuz birçok değeri için $F(u,v)$'nin kendini tekrarladığını gösterse de,yalnızca N değerleri için herhangi bir periyot $f(x,y)$'nin

$F(u, v)$'den elde edilmesini sağlar.Başka bir deyişle, $F(u, v)$'yi frekans verilerinde tanımlamak için dönüşümün yalnızca bir periyodu gereklidir.Benzer yorum uzaysal verilerde $f(x, y)$ için de geçerlidir.

$f(x, y)$ reel ise Fourier dönüşümü Conjugate simetri sergiler;

$$F(u, v) = F^*(-u,-v) \tag{4.7}$$

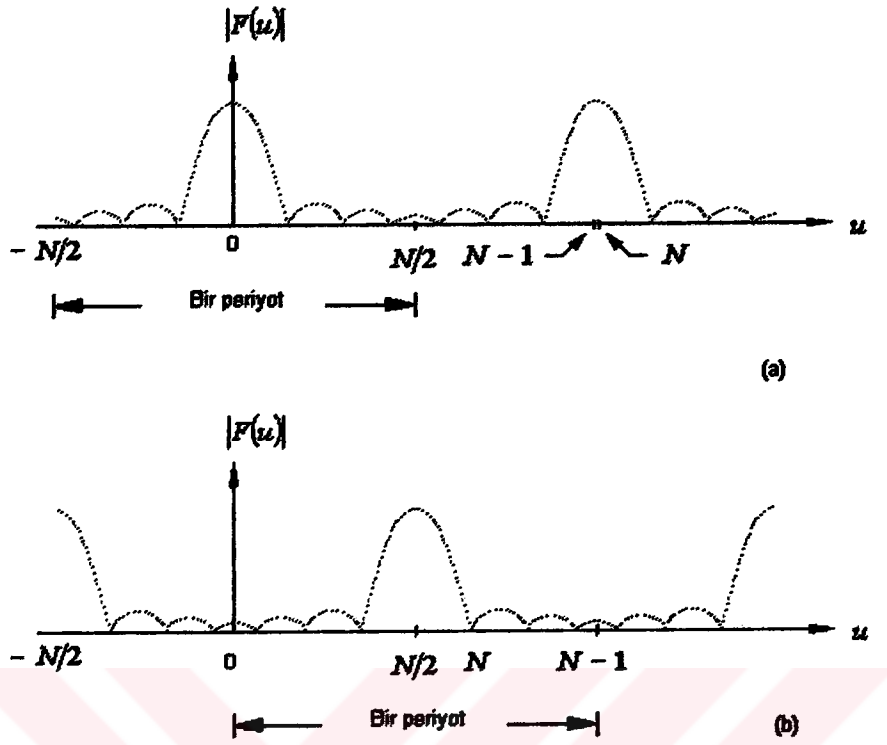
$$|F(u, v)| = |F(-u,-v)| \tag{4.8}$$

$F^*(u, v)$; $F(u, v)$'nin kompleks Conjugate'idir.Daha öncede bahsedildiği gibi bizim ilgileneceğimiz kısım Fourier dönüşümünün mutlak değerinin görüntülenmesidir. Dönüşüm mutlak değerinin görüntüsü üzerinde (4.6) ve (4.8) eşitliklerini test etmek için,önce bir değişkenli durumu ele alalım.

$F(u) = F(u + N)$ ve

$|F(u)| = |F(-u)|$

Periyodik olma özelliğinden, $F(u)$'nun N boyunda periyoda sahip olduğunu ve simetri özelliğinden de Şekil 4.6(a) 'da gösterildiği gibi dönüşümün mutlak değerinin merkezinin orjinde olduğunu biliyoruz. Şekil 4.6(a) ve daha önceki yorumlar ,$(N/2) + 1$ den $N - 1$'e kadar olan dönüşüm değerlerinin mutlak değerlerinin,orjinin sol tarafında yarı periyot içindeki değerlerin yansıması olduğunu göstermektedir.Çünkü ayrık Fourier dönüşümü u değeri için $[0, N - 1]$ aralığında formüle edilmişti, formülasyon sonucu bu aralıktaki iki arka arkaya yarı periot sonucunu verir.Bir tam periyodu görüntülemek için gereken tek şey Şekil 4.6(b)'de gösterildiği gibi dönüşümün orjinini $u = N/2$ noktasına taşımaktır.Daha öncede belirtildiği gibi dönüşümü almadan önce sadece $f(x)$'i $(-1)^x$ ile çarparız. (Chapman,1991)



Şekil 4.6 Fourier dönüşümünün periyodik olam özelliği (a)[0,N-1] aralığında arka arkaya yarı periyottaki Fourier spektrumu;(b) Aynı aralıkta kaydırılmış spektrumun tam periyodu

Dönüşümün orjini frekans noktası olan $(N/2, N/2)$'ye taşınmaması halinde sonuçların yorumlanmasının çok daha zor olması haricinde aynı yorum 2-D Fourier dönüşümünün mutlak değeri için de yapılabilir.

4.3 Rotasyon

Eğer kutupsal koordinatları

$$x = r\cos\theta \qquad y = r\sin\theta \qquad u = w\cos\phi \qquad v = w\sin\phi$$

olarak ifade edersek, $f(x, y)$ ve $F(u, v)$ sırasıyla $f(r, \theta)$ ve $F(w, \phi)$ olur.Sürekli ya da ayrık Fourier dönüşüm çiftindeki doğrudan yer değiştirme

$$f(r, \theta + \theta_0) \Leftrightarrow F(w, \phi + \theta_0) \tag{4.9}$$

şeklinde gerçekleşir.Başka bir deyişle $f(x, y)$, θ_0 açısıyla döndürüldüğünde $F(u, v)$'de aynı

açıyla döner.Benzer şekilde $F(u, v)$ döndürüldüğünde $f(x, y)$ de aynı açıyla döner.

4.4 Dağılım ve Scaling

Sürekli ve ayrık dönüşüm çiftinin tanimindan;

$$\Im\{f_1(x, y) + f_2(x, y)\} = \Im\{f_1(x, y)\} + \Im\{f_2(x, y)\} \tag{4.10}$$

ve genel olarak;

$$\Im\{f_1(x, y) \cdot f_2(x, y)\} \neq \Im\{f_1(x, y)\} \cdot \Im\{f_2(x, y)\} \tag{4.11}$$

Diğer bir deyişle Fourier dönüşümü ve ters dönüşümü toplama üzerine dağılma özelliğine sahiptir ama çarpmada dağılma özelliği yoktur.

a ve b skalerleri için ;

$$af(x, y) \Leftrightarrow aF(u, v) \tag{4.12}$$

$$f(ax, by) \Leftrightarrow \frac{1}{ab} F(u/a\ ,\ v/b) \tag{4.13}$$

4.5 Ortalama Değer

2-D ayrık fonksiyonların ortalama değerleriyle ilgili en geniş kullanıma sahip tanımlama aşağıdaki ifadedir;

$$\bar{f}(x, y) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x.y) \tag{4.14}$$

$u = v = 0$ (3.6) eşitliğinde yerine yazılırsa;

$$F(0,0) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x.y) \tag{4.15}$$

olur.Bu yüzden $\bar{f}(x, y)$ ile $f(x, y)$'nin Fourier dönüşümü arasında aşağıdaki ilişki vardır;

$$\bar{f}(x, y) = \frac{1}{N} F(0,0) \tag{4.16}$$

4.6 Konvolusyon

Konvolusyon,birçok genel görüntü işleme operatörleri içinde temel teşkil eden matematiksel bir işlemdir.Konvolusyon aynı boyutlarda üçüncü bir dizi oluşturmak için,genelde farklı ölçülerde ama aynı boyutlarda iki sayı dizisini 'birlikte çarpmayı' sağlayan bir yöntemdir.Görüntü işlemede,girdi pixel değerlerinin basit lineer kombinasyonları olan çıktı pixellerini elde etmeye yarayan işlemlerin gerçekleştirilmesi durumunda kullanılır.

Bir görüntü işlemede,girdi dizilerinden bir tanesi normal olarak yalnızca siyah-beyaz görüntü olacaktır.İkinci dizi kernel olarak bilinen genelde daha küçük ve iki boyutlu dizidir.Aşağıda ki şekil,örnek bir görüntüyü ve konvolusyonu görüntülemek için kullanacağımız kerneli gösterir.

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}

Şekil 4.7 Küçük örnek görüntü(sol) ve konvolusyonu görüntülemek için kullanılan kernel(sağ). Kareler içindeki ifadeler bulunduğu kareyi temsil etmektedir.

Konvolusyon kerneli görüntü boyunca kaydırmayı sağlar,genelde sol üst köşeden başlanır,kernel görüntünün sınırları içine tamamen uyduğu bütün pozisyonlar için hareket ettirilir.Herbir kernel pozisyonu bir çıktı pixelini karşılar.Bu değer,kerneldeki hücrelerin herbirindeki değerin,bu değere karşılık gelen görüntü değeriyle çarpılıp,sonuçta elde edilen sayıların toplanmasıyla hesaplanır.

Böylece bu örnekte alt sağdaki pixelin çıktı görüntüsünde ki değeri;

$$O_{57} = I_{57}K_{11} + I_{58}K_{12} + I_{59}K_{13} + I_{67}K_{21} + I_{68}K_{22} + I_{69}K_{23}$$

Eğer görüntü *M* satır ve *N* sütuna sahipse,kernel de *m* satır ve *n* sütundan oluşuyorsa,çıktı olan görüntü *M-m+1* satır ve *N-n+1* sütundan oluşur.*i*;1 den *M-m+1*'e , *j*;1 den *N-n+1*'e çalıştırıldığında konvolusyonu aş.şekilde yazılabilir.

$$O(i, j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i + k - 1, j + l - 1)K(k, l)$$

Bu yüzden konvolusyonun birçok uygulaması daha büyük görüntü çıktısını sağlar çünkü bu uygulama sadece görüntü sınırlarına tamamen uyduğu yerlerde hareket edebilen kernel kısıtlamasını ortadan kaldırmaktadır.Buna karşılık bu uygulamalar ,yalnızca kernelin sol üst köşesi görüntünün içinde olacak şekilde kerneli bütün pozisyonlar için kaydırır.Bu yaklaşımın bir avantajı çıktı olarak elde edilen görüntü,girdiyle aynı ölçüdedir.Ne yazık ki,görüntünün alt ve sağ kenarlarının çıktı pixel değerlerini hesaplamak için ,kernelin görüntünün sonuna dayandığı durumlarda girdi pixel değerleri buna uygun hale getirilmelidir.Genelde sıfır pixel değerleri gerçek görüntünün dışındaki bölgeler için seçilir,fakat bu işlem,bölgeye ait görüntünün çıktısında bozulmalara neden olabilir.Bu yüzden eğr böyle bir duruma yol açacak konvolusyon uygulaması kullanılıyorsa,bu bölgenin görüntüden çıkartılması iyi olacaktır.Sağ taraftan *n-1* pixelin ve alttan da *m-1* pixelin çıkartılması görüntüyü düzeltecektir. (Prentice, 1989)

Konvolusyon özellikle uzaysal filtreleme ve parça ayıklamada kullanılır.

Teorik olarak incelemek istersek,öncelikle bir boyutlu fonksiyonlar için;

$f(x)$ ve $g(x)$ fonksiyonlarının konvolusyonu $f(x) * g(x)$ şeklinde gösterilir;

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x - a)d\alpha \qquad (4.17)$$

burada α keyfi sabittir.Konvolusyon integralinin mekaniğini görsel hale getirmek kolay değildir bu yüzden (4.17) eşitliğinin kullanımını grafik olarak sunan bir örnekle tanımlamaya başlayacağız.

Örnek:İlk örnekte sırasıyla Şekil 4.4(a) ve (b)'de gösterilen $f(x)$ ve $g(x)$ fonksiyonlarının konvolusyonları ispat edilmektedir.İntegrasyondan önce $g(x - a)$ oluşturulmalıdır.Bu da Şekil 4.4 (c) ve (d)'de gösterilen iki adımı gerektirir.Bu işlem $g(\alpha)$'yı $g(-\alpha)$ olacak şekilde orjin atrafından katlamak(folding),sonra da fonksiyonun x ile yerini değiştirmekten ibarettir.Sonra her x değeri için $f(\alpha)$,uygun olan $g(x - a)$ ile çarpılır ve sonuç $-\infty$ dan

∞'a integre edilir. $f(\alpha)$ ve $g(x-a)$'nın çarpım sonuçları Şekil 4.4 (e)'deki taralı kısımdır.Bu şekil $0 \le x \le 1$ için geçerlidir. $[0,x]$ aralığı dışındaki α değerleri için çarpımın sonucu 0 dır. Şekil 4.4 (e)'deki karalanmış kısımda gösterilen $f(x) * g(x) = x/2$ dir. [1,2] aralığında ki x için, Şekil 4.4 (f) de gösterilen $f(x) * g(x) = (1 - x/2)$ dir.Böylece [0,2] dışındaki x değerleri için

$f(\alpha)\ g(x-a)$ değeri sıfır olur.Sonuç olarak ;

$$f(x) * g(x) = \begin{cases} x/2 & 0 \le x \le 1 \\ 1 - x/2 & 1 \le x \le 2 \\ 0 & diger \end{cases}$$

elde edilir. Şekil 4.4 (g) de sonuç gösterilmektedir.

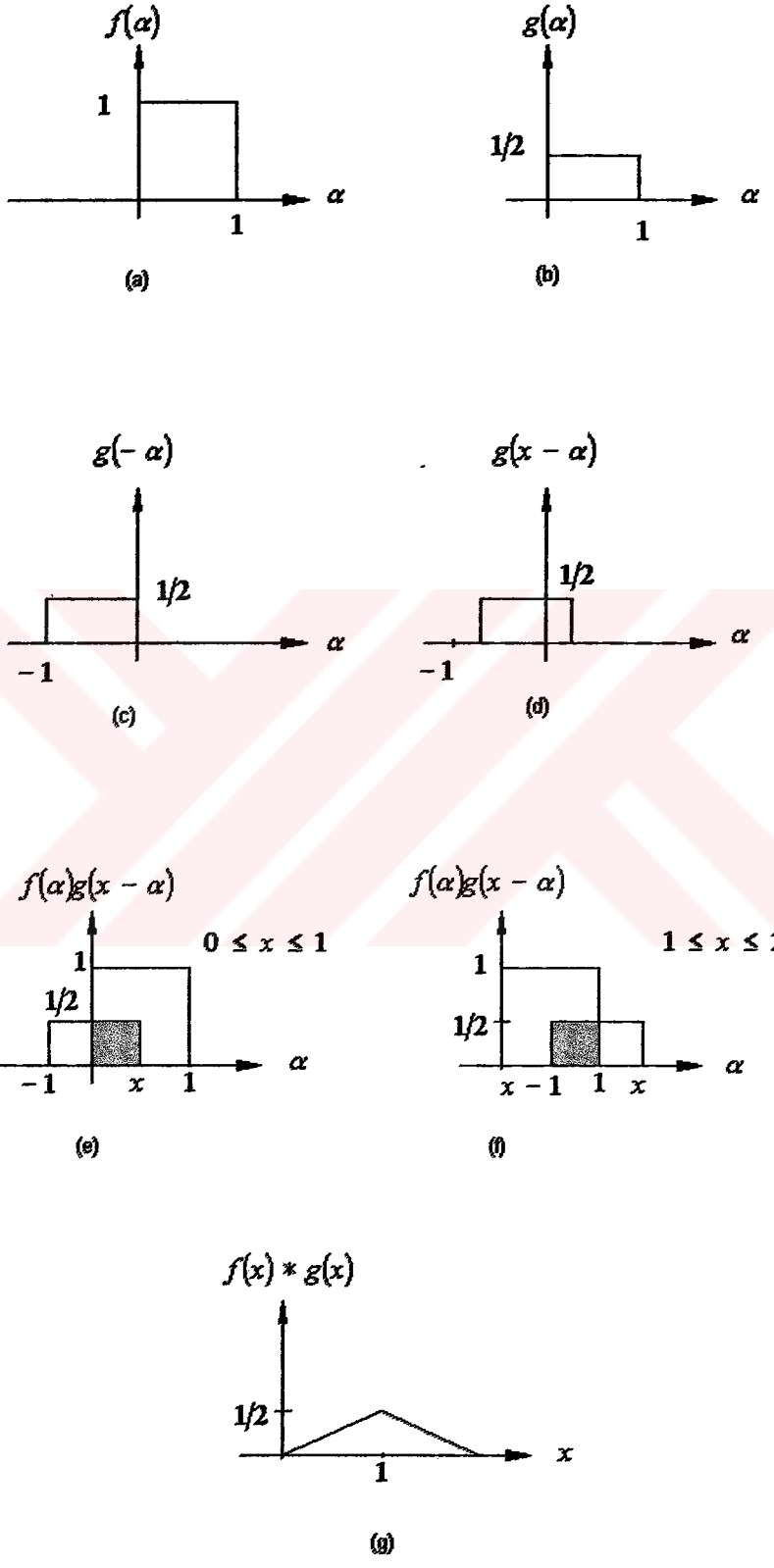
Bu bölümde daha sonra kullanılacak olan ve $f(x)$'in konvolusyonu ile impuls fonksiyonu olan $\delta(x - x_0)$'ı gerektiren ,(4.17) eşitliğinin farklı bir uygulaması aşağıdaki şekildedir;

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)dx = f(x_0) \qquad (4.18)$$

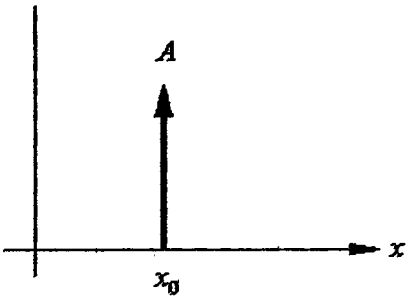
$\delta(x - x_0)$ fonksiyonu, x_0 civarında sonsuz komşuluğa sahip ve bu komşuluğun dışındaki her yerde sıfır olarak gösterilirse;

$$\int_{-\infty}^{\infty} \delta(x - x_0)dx = \int_{x_0^-}^{x_0^+} \delta(x - x_0)dx = 1 \qquad (4.19)$$

Bir çok uygulamada $\delta(x - x_0)$'in $x = x_0$'da sabitlendiğini ve impulsun gücünün $x = x_0$'da $f(x)$'in değerini belirlemekle elde edildiğini söyleyebiliriz. $f(x) = A$ olduğunda $A\delta(x - x_0); x = x_0$'da bulunan A'nın güç impulsudur.Pratikte impuls x_0'da a uzunluğunda (impulsun gücüne eşit yükseklikte) bir okla gösterilir. Şekil 12 $A\delta(x - x_0)$'nıngösterimidir.



Şekil 4.8 Konvolusyonun grafiksel gösterimi.Taralı bölgeler sonucun sıfır olmadığı yerleri gösterir.



Şekil 4.9 $A\delta(x - x_0)$'ın grafiksel gösterimi

Örnek:(4.17)'nin kullanımına ikinci örnek, Şekil 4.6(a) da gösterilen $f(x)$'in Şekil 4.10(b) de gösterilen $g(x) = \delta(x + T) + \delta(x) + \delta(x - T)$ ile konvolusyonunu içerir. $g(x)$'i , $f(x)$ boyunca kaydırmak,daha sonrada (4.17) ve (4.18) denklemlerini uygulamak Şekil 4.10(c)deki sonucu verecektir. Bu durumda konvolusyon yalnızca $f(x)$'in her bir impuls noktasında kopyalanmasıdır.

Frekans alanında konvolusyonun önemi, $f(x) * g(x)$ ve $F(u)G(u)$'nun Fourier dönüşüm çifti oluşturması gerçeğine uzanmaktadır.Başka bir deyişle, $f(x)$, $F(u)$ Fourier dönüşümüne sahip ve $g(x)$ te $G(u)$ Fourier dönüşümüne sahip işe, $f(x) * g(x)$ te Fourier dönüşümüne sahiptir ve $F(u)G(u)$ dur. Bu sonuç;

$$f(x) * g(x) \Leftrightarrow F(u)G(u) \tag{4.20}$$

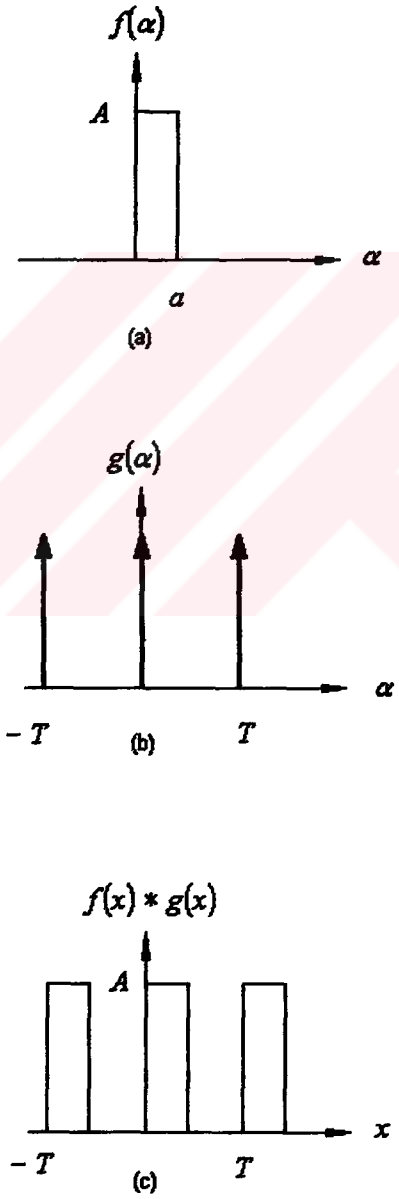
$F(u)G(u)$'nun ters Fourier dönüşümünü alarak x alanında konvolusyonun elde edilebileceğini göstermektedir.Buradan çıkan temel sonuç,frekans alanındaki konvolusyon x alanındaki çarpmayı azaltmaktadır.

$$f(x)g(x) \Leftrightarrow F(u) * G(u) \tag{4.21}$$

Bu iki sonuç Konvolusyon Teoremi olarak adlandırılır.

$f(x)$ ve $g(x)$'in sürekli olması yerine, sırasıyla $\{f(0), f(1), f(2), \ldots\ldots, f(A-1)\}$ ve

$\{g(0), g(1), g(2), \ldots\ldots, g(B-1)\}$ olmak üzere A ve B boyutlarında örneklenmiş dizilere ayrıldığını varsayalım. Bölüm(3.3)'te belirtildiği gibi ayrık Fourier dönüşümü ve ters Fourier dönüşümü periyodik fonksiyonlardır. Konvolusyon teoreminin formülasyonu ayrık fonksiyon olan $f(x)$ ve $g(x)$ fonksiyonlarını herhangi M periotlu periyodik fonksiyon olarak kabul etmeyi gerektiren periyodik olma özelliğini taşır. Böylece konvolusyonun sonucu da aynı periotta periyodik olur.Problem M değerinin seçimidir. (Anuta , 1969)



Şekil 4.10 İmpuls fonksiyonu içeren konvolusyon

$M \geq A + B - 1$

Şartını sağlayacak şekilde seçilmediğinde gösterilebilir ki, Konvolusyon'un periyodu çakışacaktır(üst üste gelecektir),bu WRAPAROUND hatası olarak adlandırılır.Eğer $M = A + B - 1$ ise periyotlar komşudur. $M > A + B - 1$ ise periotlar ayrılacaktır, ayrılmanın derecessi M ile $A + B - 1$ arasındaki fark kadar olacaktır.Çünkü kabul edilen periot değeri A ya da B'den daha büyük olmalı ,örneklenmiş dizinin uzunluğu da arttırılmış olmalıdır,böylece her iki uzunluk ta M dir.Genişletilmiş dizilerin örnek formlarına sıfırları ekleyerek;

$$f_e(x) = \begin{cases} f(x) & 0 \leq x \leq A - 1 \\ 0 & A \leq x \leq M - 1 \end{cases}$$

$$g_e(x) = \begin{cases} g(x) & 0 \leq x \leq B - 1 \\ 0 & B \leq x \leq M - 1 \end{cases}$$

Bu genişletmeye bağlı olarak , $f_e(x)$ ve $g_e(x)$'in ayrık konvolusyonu $x = 0,1,2,\ldots., M - 1$ için aşağıdaki şekilde ifade edilir;

$$f_e(x) * g_e(x) = \frac{1}{M} \sum_{m=0}^{M-1} f_e(m) g_e(x - m) \qquad (4.22)$$

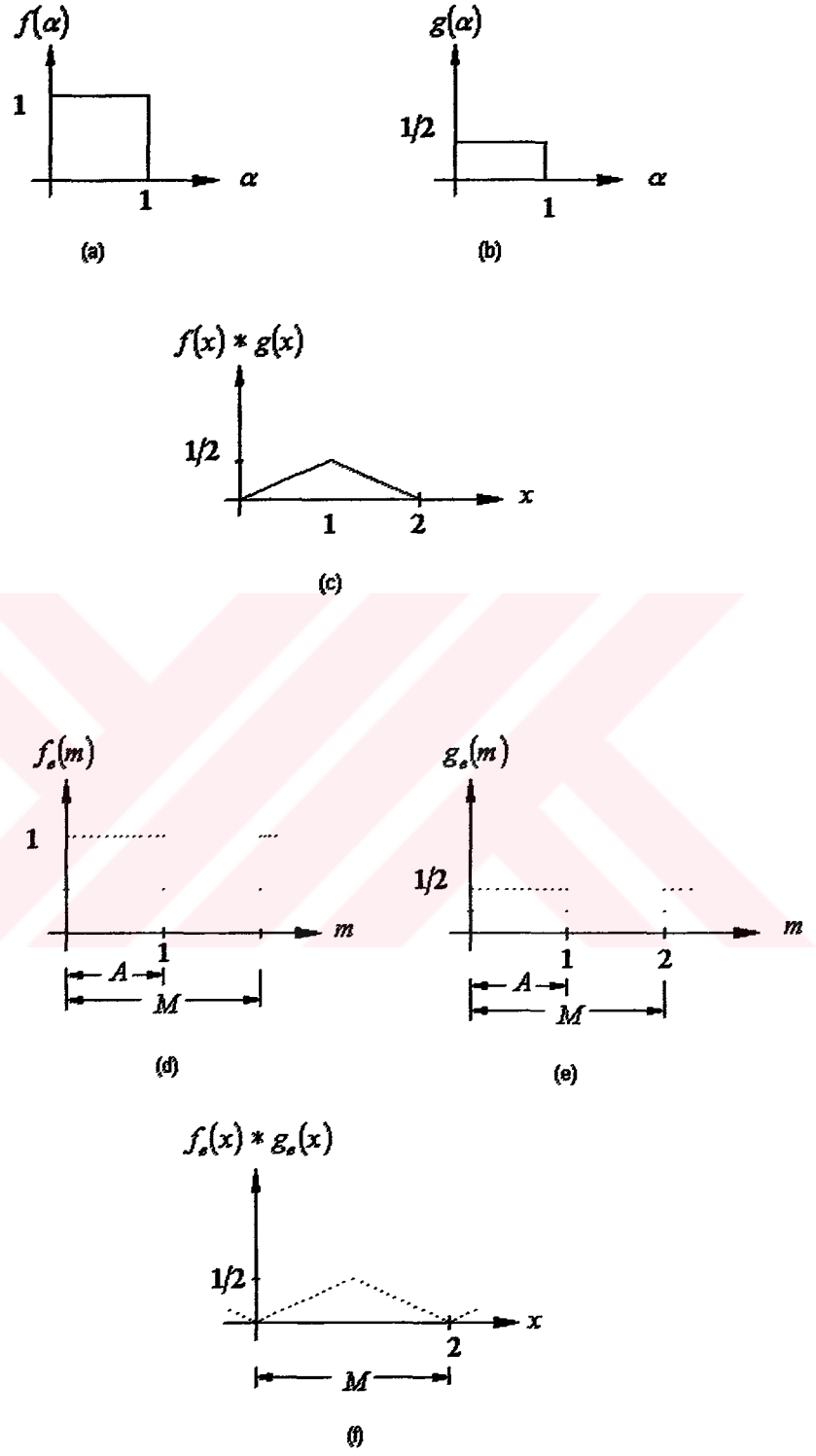
Konvolusyon fonksiyonu ayrıktır, $x = 0,1,2,\ldots., M - 1$ değerleri ile M uzunluklu periyodik dizi $f_e(x) * g_e(x)$'e için bir tam periot tanımlar.

Ayrık konvolusyonun mekaniği temelde sürekli konvolusyonla aynıdır.Aralarındaki fark örnekler arasındaki ayrılmaya uygun olarak ayrık bileşenlerin olması ve integral yerine toplamanın gelmesidir.Wraparound hatasından kaçınmak için $f_e(x)$ ve $g_e(x)$'i kullandığımızda (4.20) ve (4.21) bağıntıları da ayrık form için de geçerlidir.Ayrık değişken x ve u ;0,1,2,...., $M - 1$ değerleri içinden seçilir.

Örnek:Şekil 4.11 daha önce ele alınan sürekli ve ayrık konvolusyonu grafiksel olarak göstermektedir.Diagramlarda $f(x)$ ve $g(x)$ fonksiyonları ayrık durum için $[0,1]$ aralığında A örnekli,ve $M = A + B - 1 = 2A - 1$ periotlu olarak alınmıştır.

Burada konvolusyon fonksiyonunun periyotik olduğuna ve $M = 2A - 1$ olduğundan periotların komşu olduğuna dikkat ediniz. $M > 2A - 1$ almak bu periotlar arasında daha büyük aralıklar sağlardı. Bu yüzden M örneklerinin tamamen bir periot tanımladığına dikkat

ediniz.



Şekil 4.11 Sürekli ve ayrık konvolusyon arasında karşılaştırma

İki boyutlu konvolusyon (4.17) eşitliğine benzer formdadır. $f(x, y)$ ve $g(x.y)$ için;

$$f(x.y) * g(x.y) = \int_{-\infty}^{\infty} \int f(\alpha, \beta) g(x - \alpha, y - \beta) d\alpha d\beta \tag{4.23}$$

iki boyutlu konvolusyon denklemidir,aşağıdaki ilişki ile ifade edilir,

$$f(x, y) * g(x, y) \Leftrightarrow F(u, v) G(u, v) \tag{4.24}$$

$$f(x, y) g(x, y) \Leftrightarrow F(u, v) * G(u, v) \tag{4.25}$$

(4.23)'un grafiksel ifadesi (4.17)'ten daha zordır.Şekil 4.12 ,2-D konvolusyon için gerekli olan katlama,yer değiştirme ve çarpmayı göstermiştir.

2-D ayrık konvolusyon $f(x, y)$ ve $g(x, y)$'nin sırasıyla $A \times B$ VE $C \times D$ boyutlarında ayrık diziler olarak alınmasıyla form üle edilir.1-D de olduğu gibi diziler x ve y eksenlerinde sırasıyla periyotları M ve N olmak üzere periyodik kabul edilmelidirler. Individual periyotta Wraparound hatasından kaçınmak için,periyot aşağıdaki şekilde seçilir.

$$M \geq A + C - 1 \tag{4.26}$$

$$N \geq B + D - 1 \tag{4.27}$$

Periyodik diziler $f(x.y)$ ve $g(x.y)$ fonksiyonlarının genişletilmesiyle oluşturulur.

$$f_e(x, y) = \begin{cases} f(x, y) & 0 \leq x \leq A - 1 \quad ve \quad 0 \leq y \leq B - 1 \\ 0 & A \leq x \leq M - 1 \quad yada \quad B \leq y \leq N - 1 \end{cases}$$

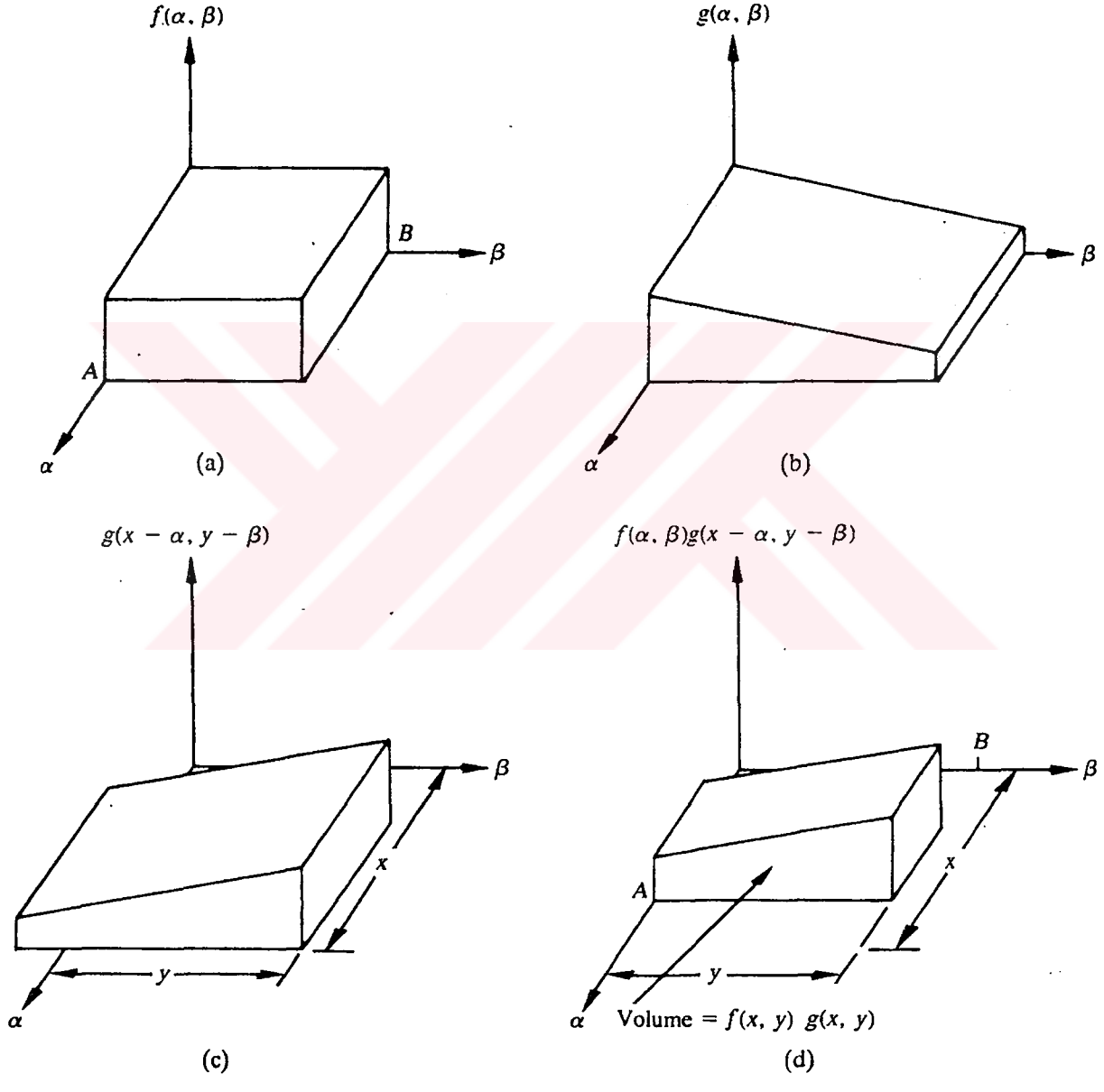
$$g_e(x, y) = \begin{cases} g(x, y) & 0 \leq x \leq C - 1 \quad ve \quad 0 \leq y \leq D - 1 \\ 0 & C \leq x \leq M - 1 \quad yada \quad D \leq y \leq N - 1 \end{cases}$$

$f_e(x, y)$ ve $g_e(x, y)$'nin konvolusyonu $x = 0,1,2,\ldots., M - 1$ ve $y = 0,1,2,\ldots., N - 1$ için;

$$f_e(x, y) * g_e(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_e(m, n) g_e(x - m, y - n) \tag{4.28}$$

ile tanımlanır.Bu eşitlikteki $M \times N$ dizisi,2-D ayrık konvolusyonun bir periyodudur.Eğer M ve N ,(4.26) ve (4.27) eşitliklerine uygun olarak seçilirse bu dizi diğer komşu periyotlar arasında özgür olmayı garantiler.1-D de sürekli konvolusyon teoreminde olduğu gibi (4.24) ve (4.25) bağıntıları ayrık durum için $u = 0,1,2,\ldots., M - 1$ ve $v = 0,1,2,\ldots., N - 1$

alınarak uygulanabilir.Bütün hesaplamalar genişletilmiş $f_e(x, y)$ ve $g_e(x, y)$ fonksiyonlarını gerektirir.



Şekil 4.12 İki boyutlu konvolusyon için gerekli olan folding(katlama),yer değiştirme ve çarpma işlemleri

Örnekleme teoremi açıklanırken Konvolusyon teoreminin teorik gücü ispatlanacaktır.Pratikte frekans alanında ayrık konvolusyonun hesaplanması,(4.28)eşitliğinin doğrudan kullanılmasından daha etkilidir. $f_e(x, y)$ ve $g_e(x, y)$'nin Fourier dönüşümünün hesaplanması Bölüm 5'te gösterilecek hızlı Fourier dönüşümü (FFT) ile olacaktır. (Anuta , 1969)

4.7 Örnekleme(sampling)

Bilgisayar işlemlerine uygun olarak görüntü fonksiyonu $f(x, y)$ uzaysal ve genişlik olarak dijitize edilmelidir.Uzaysal koordinatlardaki *(x,y)* dijitizasyon görüntü örnekleme ve genişliklerde yapılan dijitizasyon gri-seviyeli hale getirme olarak adlandırılır.

Sürekli $f(x, y)$ görüntüsünün,aşağıda gösterilen eşit aralıklı örneklerin N x M formunda oluşturduğu dizisiye yaklaştığını düşünelim;

$$f(x, y) \approx \begin{bmatrix} f(0,0) & f(x, y) & \ldots\ldots & f(0, M-1) \\ f(1,0) & f(x, y) & \ldots\ldots & f(1, M-1) \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ f(N-1,0) & f(x, y) & \ldots\ldots & f(N-1, M-1) \end{bmatrix} \tag{4.29}$$

yukarıda ki (4.29) ifadesinde sağ taraf dijital görüntüyü gösterir.Dizinin herbir elemanı;görüntü elemanı,resim elmanı,pixel ya da pel olarak adlandırılır.Görüntü ve pixel terimleri bundan sonra dijital görüntü ve elemanları olarak anılacaktır.

Burada problem,seçilen örnekleme değerlerinden sürekli görüntü tekrar elde edilebilecek şekilde örnekleme şartlarının oluşturulmasıdır.İncelemeye 1-D durumu için başlanacaktır.

Bir boyutlu fonksiyonlar

Şekil 4.13 de gösterilen, $-\infty$ dan ∞'a genişletilmiş fonksiyonu ele alalım. $f(x)$'in fourier dönüşümünün $[-W, W]$ aralığı dışındaki u değerlerinde varolmadığını kabul edelim.Dönüşüm Şekil 4.13(b) deki gibi olacaktır. W'nin herhangi sonlu değeri için bu özelliği sağlayan fonksiyonlara *bant limitli fonksiyon* denir.

$f(x)$'in örneklenmiş versiyonunu elde etmek için,fonksiyonu örneklenmiş fonksiyon $s(x)$ ile çarpmak gerekmektedir. $s(x)$;Şekil 4.13(c) de görüldüğü gibi Δx aralıklı impulslardan

oluşmaktadır. Konvolusyon teoreminden x verilerindeki çarpım,frekens alanındaki konvolusyona eşittir. Şekil 4.13(f) teki fourier dönüşümü $s(x)f(x)$'in sonucundan elde edilmiştir.Dönüşüm periyodu $1/\Delta x$ olmak üzere periyodiktir ve $F(u)$'nun kendini tekrarı üst üste gelebilir.Örneğin ilk periyotta üst üste gelen bölgenin merkezi $u = 1/2\Delta x$ olur. ($u = 1/2\Delta x$ değeri W den az olduğunda)Bu problemden kaçınmak için Δx örnekleme aralığını $1/2\Delta x \geq W$ olacak şekilde ya da;

$$\Delta x \leq \frac{1}{2W} \tag{4.30}$$

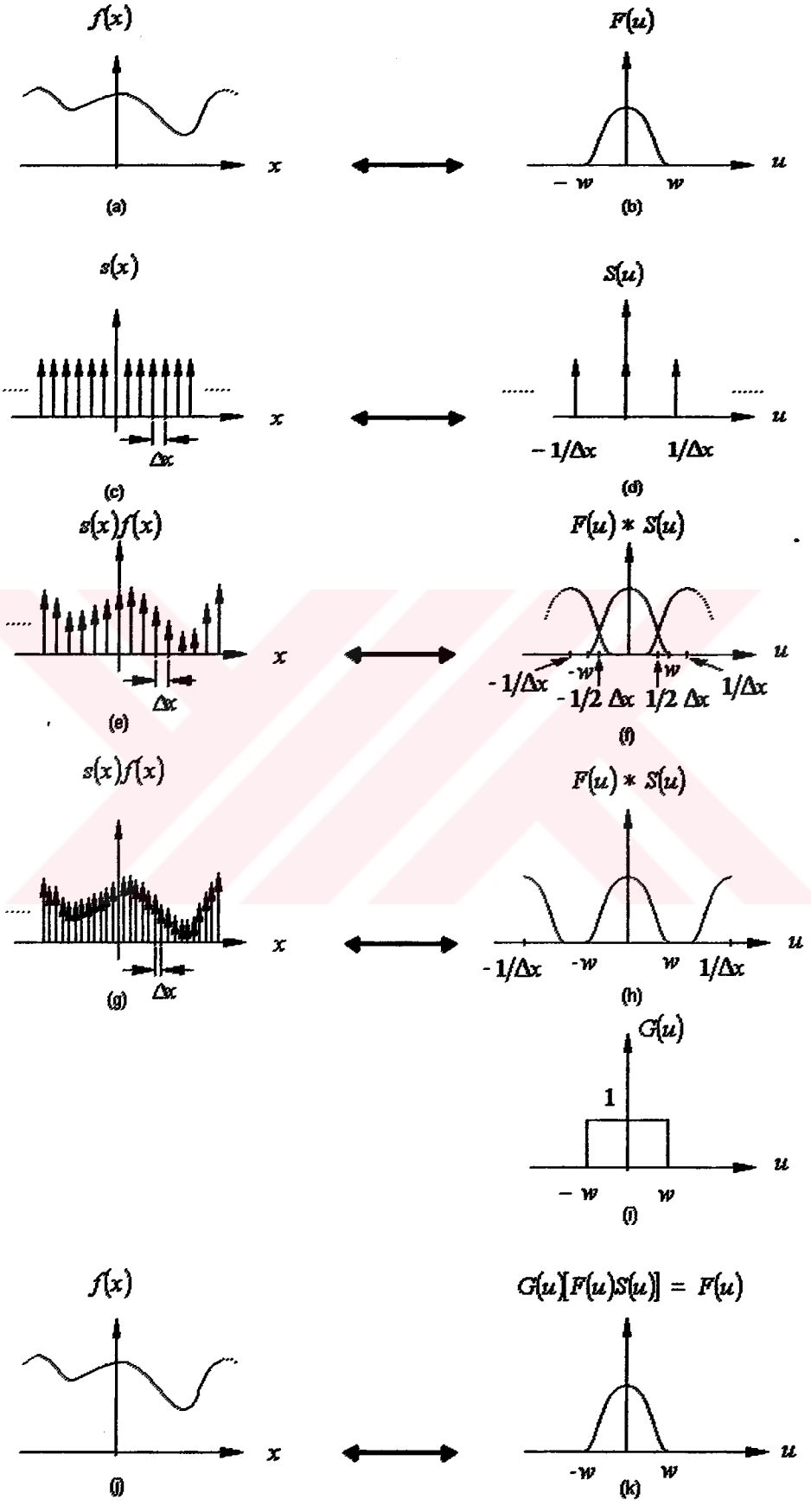
olacak şekilde seçeriz

Δx teki azalmanın sonucu Şekil 4.13(g) ve (h) de görülmektedir.Kesin etki ise üst üste gelmenin olmaması için periyotların bölünmesidir.Bu işlemin önemi Şekil 4.13(k) da görüldüğü gibi $F(u)$ nun tam izolasyonunu mümkün kılmak için ,Şekil 4.13(h) nin dönüşümünün aşağıdaki fonksiyonla çarpımıdır.

$$G(u) = \begin{cases} 1 & -W \leq u \leq W \\ 0 & diger \end{cases} \tag{4.31}$$

Ters dönüşüm sonrasında orijinal sürekli fonksiyon $f(x)$ elde edilir. Bant limitli fonksiyonun aralıkları (4.30) şartını sağlayan örneklerinden tamamen geri elde edilmesi *Whittaker-Shannon örnekleme teoremi* olarak bilinir.

Bant limitli fonksiyonun bütün frekans verilerindeki bilgileri $[-W, W]$ aralığındadır.Eğer (4.30) sağlanmıyorsa,bu aralıktaki dönüşüm komşu periyotların katılımıyla bozulur.Bu durum *aliasing* olarak adlandırılır.Bu örneklenmiş fonksiyonun yeniden elde edilmesini engeller.



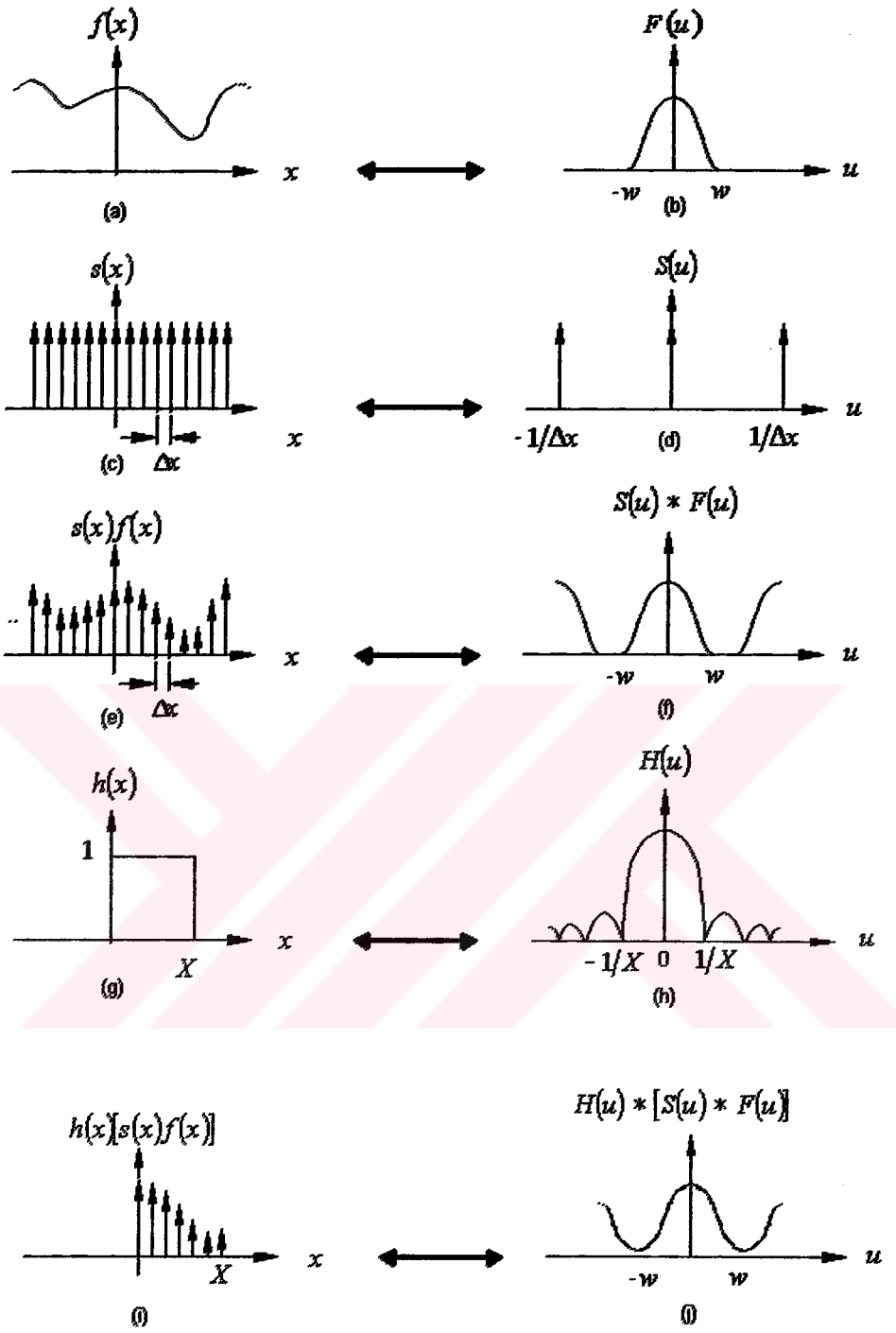
Şekil 4.13 Sampling(örnekleme)'nin grafiksel gösterimi

Önceki sonuçlar , x verilerinde limitsiz zamanlı fonksiyonlara uygulanabilir.Bunun anlamı sonsuz örneklenmiş aralıktır,bu yüzden uygulama durumunda fonksiyon bir sonlu bölge üzerinde örneklenir. Şekil 4.14(a)-(f) te grafiksel olarak gösterilen bu durum, aliasing den kaçınmak için örnekler arasındaki aralıkların örnekleme teoremini sağlıyor kabul edilmesi haricinde Şekil 4.13 ile aynıdır.Sonlu örnekleme aralığı $[0, X]$; Şekil 4.14(e)'de gösterilen örnekleme sonucu ile,aşağıda verilen fonksiyonun çarpımı ile matematiksel olarak gösterilebilir.

$$h(x) = \begin{cases} 1 & 0 \le x \le X \\ 0 & diger \end{cases} \tag{4.32}$$

Bu fonksiyon genelde *pencere* olarak adlandırılır ve Fourier dönüşümüyle sırasıyla Şekil 4.14(g) ve(h) de gösterilmektedir.Çarpmanın sonucu Şekil 4.14(i) ve(j) de görülmektedir.En son frekans alanındaki sonuç $S(u) * F(u)$'nun $H(u)$ ile konvolusyonundan elde edilir.($H(u)$ burada $h(x)$'in Fourier dönüşümüdür.)Çünkü $H(u)$ sonsuza genişletilmiş frekans bileşenleri içerir,bu fonksiyonların konvolusyonu örneklenmiş ve $h(x)$ile sonlu bölgeye limitlenmiş fonksiyonun frekens alanındaki ifadesinde Şekil 4.14(j) de görüldüğü gibi hasara yol açar.Böylece örnekler arasındaki aralık örnekleme teoremini sağlıyor bile olsa,yalnızca x verilerinin sonlu bölgesinde örneklenmiş fonksiyonu yeniden tamamen elde etmek genelde imkansızdır.Bu şartlar altında orijinal Fourier dönüşümü yalnızca $f(x)$ bant limitli ve periyodu X omak üzere periyodik olduğunda korunabilir.Bu durumda $H(u)$'nun sebep olduğu bozulma ortadan kalkar,örnekleme teoremi sağlanırsa $f(x)$ yeniden elde edilebilir. Yeniden elde edilen fonksiyon hala $-\infty$ dan ∞'a genişletilmiş olup $h(x)$'in sıfır olduğu bölge dışında da sıfır değildir.Bu tanımlamalar bizi sonlu zamanlı $f(x)$ fonksiyonunun hiçbir zaman bant limitli olamayacağı sonucuna götürür.Ya da tersi söylenirse,bant limitli bir fonksiyon x verilerinde $-\infty$ dan ∞'a genişletilmelidir.Bu önemli bir uygulama sonucudur çünkü dijital fonksiyonların uygulamasında ana limitasyon olarak kabul edilir.

1-D fonksiyonlarını geçmeden önce ,ayrık Fourier dönüşümünün periyodik olması için,önceki sonuçlardan bulunacak alternatif bir sebep daha vardır.Şu ana kadar frekans alanındaki bütün sonuçlar sürekli yapıdaydı.Ayrık Fourier dönüşümünü elde etmek için sadece sürekli dönüşümü Δu aralıklı impulslar ile örneklemek gerekir. Şekil 4.15; Şekil 4.14(i) ve(j) ye bağlı olarak bu durumu göstermektedir. Şekil 4.15 (a) ve (b) , Şekil 4.14 deki işlemler dizisinin sonucu olarak kabul edilir. (Anuta , 1969)

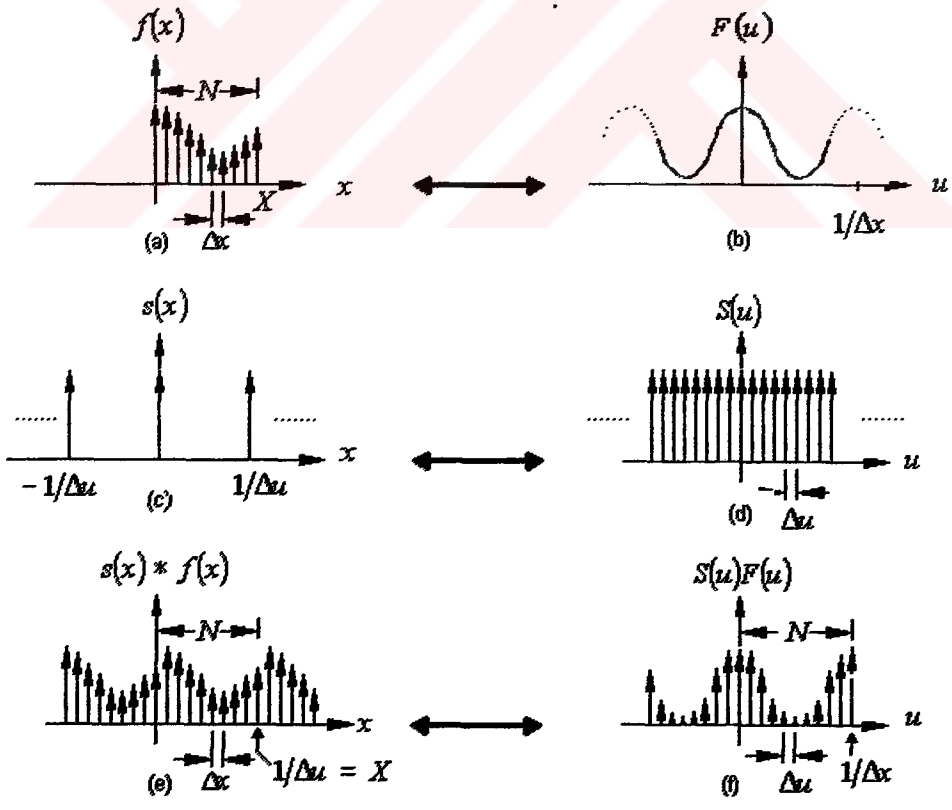


Şekil 4.14 Sonlu örneklemenin grafiksel gösterimi

Öncelikle dikkat edilmesi gereken,örnekleme;ilgili fonksiyonun impuls kümesi ile çarpılmasıdır.Bu durumda $F(u)$ nun $S(u)$ ile çarpımı Şekil 4.15 (f) deki sonucu verir. x verilerindeki eşdeğer işlem konvolusyondur ve Şekil 4.15 (e)de gösterilmiştir.Bu fonksiyon periyodu $1/\Delta u$ olmak üzere periyodiktir.Eğer $f(x)$ ve $F(u)$ N örnekli alınırsa ve örnekler arasındaki boşluklar her alanda(x verilerinde,frekans verilerinde...) bir periyot N eşit boşluklu örneklerden oluşacak şekilde seçilirse, x verilerinde $N\Delta x = X$ ve frekans verilerinde $N\Delta u = 1/\Delta x$ olacaktır.Sonraki eşitlik örneklenmiş fonksiyonun Fourier dönüşümünün periyodiklik özeeliğine bağlıdır,periyot $1/\Delta x$ tir.Buradan

$$\Delta u = \frac{1}{N\Delta x} \tag{4.33}$$

(3.4) eşitliğiyle aynı sonuç çıkmaktadır. Bu koşulu sağlayan boşluğun seçimi $1/\Delta u$ periyotlu Şekil 4.15 (e) deki fonksiyonu verecektir.(4.33)'den çıkan $1/\Delta u = N\Delta x = X$;Şekil 4.15 (a)'daki örnekleme aralığının tümünü ifade etmektedir.



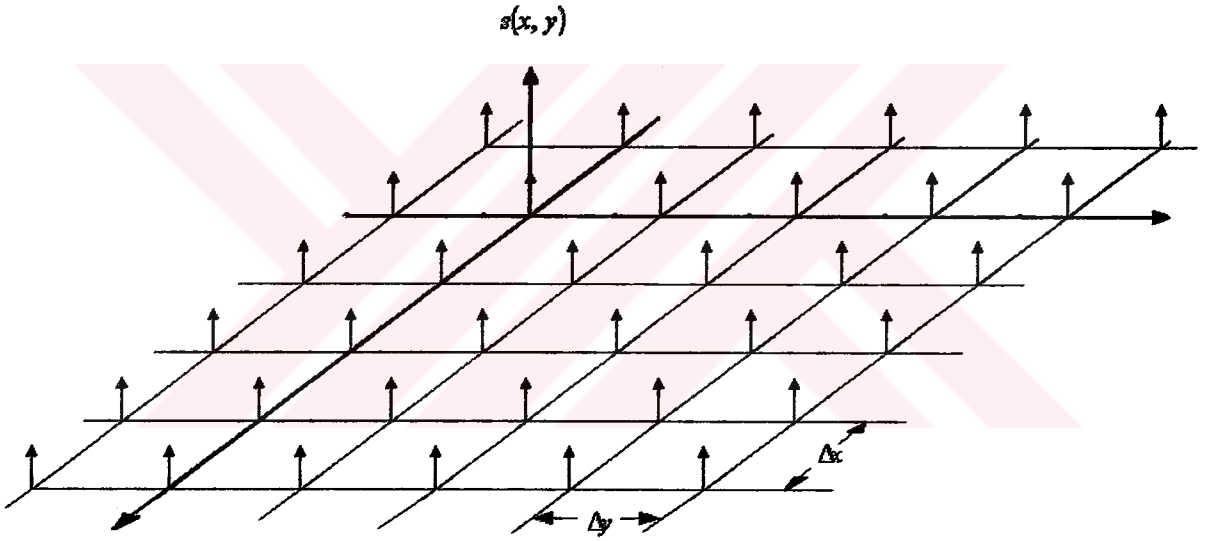
Şekil 4.15 Ayrık Fourier dönüşümünün grafiksel gösterimi

İki boyutlu fonksiyonlar

Önceki örnekleme kavramları,notasyonda yapılacak birkaç değişiklikten sonra 2-D fonksiyonlara doğrudan uygulanabilir. yapılacak birkaç değişiklikten sonra 2-D fonksiyonlara doğrudan uygulanabilir.Bu fonksiyonlar için örnekleme süreci 2-D impuls fonksiyonu $\delta(x, y)$ ile matematikel olarak ifade edilebilir.

$$\int_{-\infty}^{\infty}\int f(x, y)\delta(x - x_0, y - y_0)dxdy = f(x_0, y_0) \tag{4.34}$$

Bir 2-D örneklenmiş fonksiyonu, x doğrultusunda Δx, y doğrultusunda ise Δy aralıklarına bölünmüş impulslardan oluşur.



Şekil 4.16 2-D sampling fonksiyonu

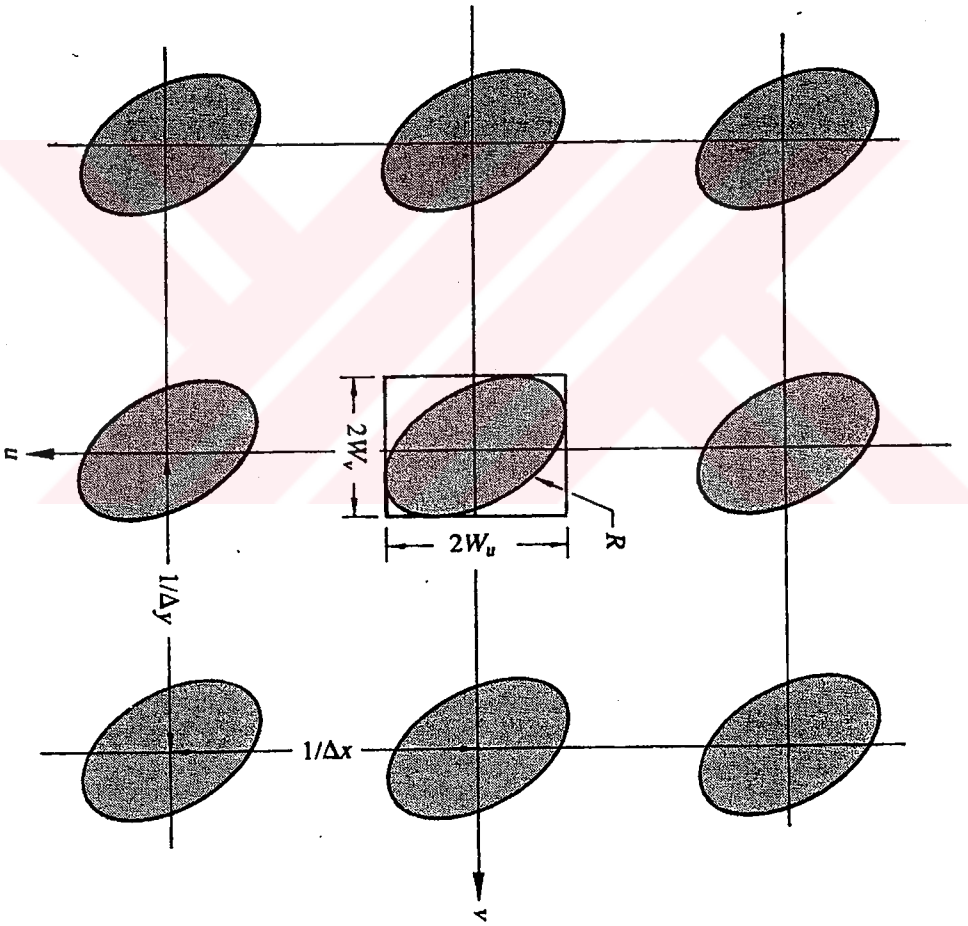
Bir $f(x, y)$ fonksiyonu için, x ve y sürekli,örneklenmiş fonksiyon $s(x, y)f(x, y)$ nin düzenlenmesiyle elde edilir.Aynı işlem frekans alanında $S(u, v)$ ve $F(u, v)$'nin konvolusyonudur. $S(u, v)$; u ve v doğrultularında sırasıyla $1/\Delta x$ ve $1/\Delta y$ aralıklarıyla ayrılmış impulslardan oluşur. $f(x,y)$ bant limitli fonksiyon ise(yani sonlu bir R bölgesi dışında Fourier dönüşümü olmuyorsa) $S(u, v)$ ve $F(u, v)$'nin konvolusyonu Şekil 4.17 deki gibi olacaktır.Fonksiyon iki boyutta periyodik olarak görülüyor.

$2W_u$ ve $2W_v$ sırasıyla u ve v doğrultularında R bölgesini içine alan en küçük dikdörtgenin

genişlikleri olsun.Böylece Şekil 4.17 den eğer ve $1/\Delta y > 2W_v$ (aliasing olmadan) olursa, $S(u, v) * F(u, v)$ ile aşağıdaki fonksiyonun çarpılmasıyla periyotlardan biri yeniden elde edilebilir;

$$G(u, v) = \begin{cases} 1 & (u, v), \ dörtgenin \\ & içinde \\ 0 & baska \quad yerde \end{cases} \tag{4.35}$$

$G(u, v)[S(u, v) * F(u, v)]$'nin ters fourier dönüşümü, $f(x, y)$'yi verir.



Şekil 4.17 2-D Örneklemede frekans verilerinin gösterimi,bant limitli fonksiyonlar

Yukarıda ki tanımlamalar 2-D örnekleme teoremi için yol gösterecektir.Bant limitli fonksiyon $f(x.y)$,aşağıdaki şartları sağlayacak aralıklarla örneklendiğinde tamamen yeniden elde edilebilir;

$$\Delta x \leq \frac{1}{2W_u} \tag{4.36a}$$

$$\Delta y \leq \frac{1}{2W_v} \tag{4.36b}$$

$f(x.y)$;2-D pencere fonksiyonu $h(x, y)$ kullanılarak uzaysal olarak limitlendiğinde örneklenmiş fonksiyonun dönüşümü, $H(u, v)$ ile $S(u, v) * F(u, v)$'nin konvolusyonu tarafından bozulmaya uğrar.Dijital görüntünün uzaysal limitlenmesinden dolayı olan bu bozulma,örneklerinden $f(x.y)$'nin tamamen elde edilmesini engeller. 1-D de olduğu gibi (periyodik fonksiyonların dışında) bu şartları sağlayan görüntüler pratikte çok az gerçekleşir.

$N \times N$ lik bir görüntü için örneklerin bölünmesiyle ilgili aşağıdaki ilişkiler,uzaysal ve frekans verilerinde, bir tam 2-D periyodunu $N \times N$ eşit aralıklı değerlerle tamamen örtülmesini garantiler.

$$\Delta u = \frac{1}{N\Delta x} \tag{4.37a}$$

$$\Delta v = \frac{1}{N\Delta y} \tag{4.37b}$$

Bahsedilen Fourier özellikleri ve dönüşümle ilgili aşağıda örnekler verilmiştir;

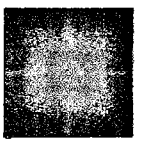
Aşağıdaki görüntüye Fourier dönüşümünü uygulayalım;

Şekil 4.18 Fourier dönüşümü uygulanacak örnek görüntü

Şekil 4.18'deki kompleks sonuçtan hesaplanan mutlak değer(büyüklük) ;

Şekil 4.19 Örnek görüntünün Fourier dönüşümü

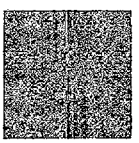
şeklinde görülmektedir.DC-değerinin,görüntünün en geniş bileşeni olduğunu görebiliriz. Fourier katsayılarının(Fourier görüntüsündeki yoğunluk değerleri) dinamik bölgesi ekranda gösterilebilmek için fazla geniştir.Bu yüzden diğer bütün değerler siyah olarak görünür.Eğer *Logaritmik dönüşüm* uygulanırsa aşağıdaki görüntü elde edilir;



Şekil 4.20 Fourier dönüşümü alınan görüntünün Logaritmik dönüşümü

Bu sonuç bize,görüntünün,frekensların bütün bileşenlerini içerdiğini gösterir.Buradan düşük frekansların yükseklere oranla daha fazla görüntü bilgisi içerdiği sonucunu çıkarabiliriz.Dönüşüm görüntüsü bize Fourier görüntüsünde iki baskın yön olduğunu göstermektedir.Bunlar merkezden yatay ve dikey şekilde geçer.Bunlar orijinal görüntünün arka planındaki düzenli desenlerden kaynaklanmaktadır.

Aynı görüntünün Fourier dönüşümünün fazı aşağıdaki şekilde gösterilmiştir;



Şekil 4.21 Örnek görüntünün Fourier dönüşümünün faz görüntüsü

Herbir noktanın değeri,uygun frekansın fazını belirler.Mutlak değer görüntüsünde olduğu gibi,orijinal görüntüdeki desenlere uygun yatay ve dikey çizgiler tanımlayabiliriz.Faz görüntüsü,uzaysal verilerdeki görüntünün yapısıyla ilgili yeni bilgiler vermeyecektir.Bu yüzden bundan sonraki örnekte yalnızca Fourier dönüşümünün mutlak değeri görüntülenecektir.

Faz görüntüsünü geçerken,yukarıdaki mutlak değer görüntüsüne fazı ihmal ederek ters Fourier dönüşümünü uygularsak;

Şekil 4.22 Faz ihmal edilerek Örnek görüntünün Fourier dönüşümüne ters dönüşümün uygulanması

elde ederiz.Bu görüntü orijinal görüntü ile aynı frekansları içermesine rağmen,seçilemeyecek kadar bozulmuştur.Bu bize uzaysal verilerde doğru görüntünün yeniden elde edilmesinde faz bilgisinin çok önemli olduğunu gösterir.

Bazı basit görüntülerle dönüşümün yapısını daha iyi anlamak için örnekler verilecektir.Uzaysal verilerdeki periyodik desenlerin Fourier dönüşümü etkilerini incelemek için aşağıdaki yapay görüntüyü ele alalım.

Şekil 4.23 Örnek yapay görüntü

görüntü 2 pixel genişliğinde genişliğinde çizgilerden oluşmaktadır. Fourier dönüşümü;

Şekil 4.24 Örnek görüntünün Fourier dönüşümü

şeklindedir.Dikkatlice bakılırsa 3 ana değer içerdiğini görebiliriz;DC değeri ve, Fourier görüntüsünün merkezinden simetrik olmasından dolayı,orijinal görüntüdeki çizgilerin frekensına karşılık gelen iki nokta.İki nokta merkezde yatay çizgi üzerindedir,bunun sebebi uzaysal verilerde yoğunluk en çok yatay olarak ilerlediğimizde değişir. (Andrews , 1977)

Noktaları merkeze olan uzaklığı şu şekilde açıklanabilir; maksimum frekans uzayda bir pixel genişliğinde çizgiler olarak ifade edilebilir;

$$f_{\max} = \frac{1}{1 pixel} \tag{4.38}$$

buradan iki pixel genişliğindeki çizgiler;

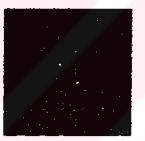
$$f = \frac{1}{2pixel} = \frac{f_{max}}{2} \tag{4.39}$$

olarak ifade edilir.Böylece noktaların merkez ile kenar ortasında olduğu sonucu çıkar.

Aşağıdaki görüntüye Fourier uygulandığında;

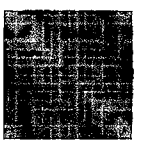
Şekil 4.25 Örnek görüntü

Aşağıdaki sonuç elde edilir;



Şekil 4.26 Örnek görüntünün Fourier dönüşümü

Fourier dönüşümünün mutlak değerini gösteren görüntüde,DC değeri ve çizgilerin frekansına uygun iki noktayı tekrar görmekteyiz.Logaritmik dönüşümü alındığında;



Şekil 4.27Fourier dönüşümünün Logaritmik dönüşümü

görüntü birçok küçük frekans içermektedir.Bunun sebebi yalnızca köşegen kare pixellerle görüntüye yaklaşır,buradan görüntüyü oluşturmak için ilave frekanslara ihtiyaç duyulduğu sonucu çıkar.Logaritmik scaling orijinal görüntüdeki tekil frekansların etkisini söylemeyi zorlaştırır.En önemli frekansları bulmak için resmi threshold(siyah-beyaz hale getirme) ederiz.

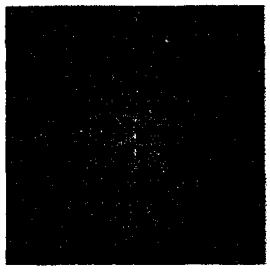
Şekil 4.28 Treshold edilmiş görüntü

Mutlak değeri,maksimumun en azından %5'i olan bütün frekansları gösterir.DC-değerinin %5'inden küçük olan bütün frekanslar yok olmuştur. (Ballard,1982)

Uzaysal verilerdeki görüntünün geometrik yapısı ile ilgili bilgi elde etmek için aşağıdaki görüntüye Fourier dönüşümünü uygulayalım.Yazının satırlarının doğrultusu önceden ifade edildiğinde ,yazı çözümleme görüntü işleme tekniklerinin kullanılmasıyla kolaylaşılır.Burada yazının ilgili düzenlemelerini bulmak için nasıl Fourier dönüşümünün kullanıldığı gösterilmiştir ve hataların düzeltilmesi için *döndürme özelliği* uygulanabilir.

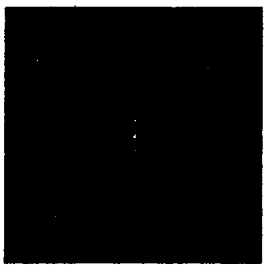
Şekil 4.29 Örnek görüntü

Bu teknik ingilizce bir yazının ikili görüntüsü üzerinde gösterilmiştir. Fourier dönüşümünün mutlak değerinin logaritması ;



Şekil 4.30 Örnek görüntünün Fourier dönüşümünün mutlak değerinin Logaritması

olur.Ve

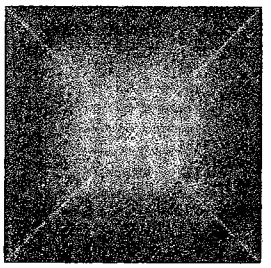


Şekil 4.31 Treshold edilmiş görüntü

Fourier görüntüsünün siyah-beyaza indirgenmiş mutlak değeridir.Dikey çizgide uzanan ana değerleri görebiliriz.Bu bize girdi görüntüdeki yazının satırlarının yatay olduğunu gösterir.Aynı şekide ilerlediğimizde , 45° 'lik döndürmeyle;

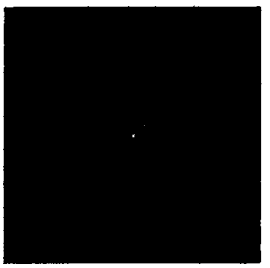
Şekil 4.32 Örnek görüntünün 45° 'lik döndürülmüş hali

olur.Buradan;



Şekil 4.33 45° 'lik döndürülmüş örnek görüntünün Fourier dönüşümünün mutlak değerinin Logaritması

elde edilir. Fourier uzayında;



Şekil 4.34 45° 'lik döndürülmüş örnek görüntünün Treshold edilmiş hali

şeklindedir. Fourier verilerindeki en yüksek değerdeki noktaların çizgisinin,girdi görüntüsünün döndürülmesine uygun olarak döndüğünü görebiliriz.Logaritmik görüntüdeki ikinci çizgi(ana doğrultuya dik olan) dönmüş resimdeki siyah köşelerden başlamaktadır.

Maksimum noktaları arka plandan ayırmış olmamıza rağmen,yazının düzensiz görüntüsünden elde edilen Fourier görüntüsünde birçok pürüz vardır.Eğer yazıdan içi dolu bloklar çıkarabilseydik,arka plan değerlerini azaltıp,maksimum noktaların görüntüsünü daha belirgin yapabilirdik. [Chang, 1989]

5. HIZLI FOURIER DÖNÜŞÜMÜ (The Fast Fourier Transform)

(3.2) eşitliğinin gerektirdiği kompleks çarpma ve toplama sayısı N^2 ile orantılıdır. u nun N tane değerinin herbiri için toplamanın büyümesi $f(x)$ ile $\exp[-j2\pi ux/N]$ in N adet kompleks çarpımını ve sonuca $N-1$ adet eklemeyi gerektirir. $\exp[-j2\pi ux/N]$'in terimleri daha sonraki uygulamalar için hesaplanarak saklanabilir.Bu sebepten dolayı terimlerin içindeki u nun x ile çarpımı uygulamanın doğrudan bir parçası olarak kabul edilmez.

(3.2) eşitliğinin uygun ayrıştırması toplama ve çarpmaların sayısı $N \log_2^N$ ile orantılı olarak yapılabilir.Ayrıştırma prosedürü *hızlı Fourier dönüşümü* (FFT) olarak adlandırılır.Bu orandaki azalma Tablo 5.1 de görüldüğü gibi hesaplamada harcanan çabada dikkate değer bir azaltma sağlayacaktır.FFT özellikle N in daha büyük olduğu durumlarda Fourier dönüşümünün doğrudan uygulaması üzerinde önemli şekilde hesaplamada avantaj sağlar.Örneğin 8192 noktalı bir dizi için FFT 5 saniyede hesaplama yaparken,aynı teknik şartlar altında (3,2) eşitliği ile Fourier dönüşümü aynı dizi için 600 kez daha fazla (50 dakika)zaman alır.

Bir sonraki konuda FFT algoritması 1-D de geliştirilecektir. (Andrews , 1977)

5.1 FFT Algoritması

Bu bölümde FFT Algoritması ardıl ikiye katlama yöntemi kullanılarak yapılacaktır.Uygun olarak (3.2) eşitliğini aş. şekilde yazacağız.

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) W_N^{ux} \tag{5.1}$$

$$W_N = \exp[-j2\pi/N] \tag{5.2}$$

Çizelge 5.1 N değişkenleri için $N \log_2^N$'e karşı N^2 nin karşılaştırılması

N	N^2 (doğrudan FFT)	$N \log_2^N$ (FFT)	$(N/\log_2^N)$ Hesaplama avantajı
2	4	2	2.00
4	16	8	2.00
8	64	24	2.67
16	256	64	4.00
32	1,024	160	6.40
64	4,096	384	10.67
128	16,384	896	18.29
256	65,536	2,048	32.00
512	262,144	4,608	56.89
1024	1,048,576	10,24	102.40
2048	4,194,304	22,528	186.18
4096	16,777,216	49,152	341.33
8192	67,108,864	106,496	630.15

n pozitif tamsayı olmak üzere N aşağıdaki formda kabul edilir.

$$N = 2^n \tag{5.3}$$

Buradan N ifadesi M de pozitif tamsayı olmak üzere aş.şekli alır.

$$N = 2M \tag{5.4}$$

(5.4)'ü (5.1) de yerine yazarsak aş. sonucu elde ederiz.

$$F(u) = \frac{1}{2M} \sum_{x=0}^{2M-1} f(x) W_{2M}^{ux} \tag{5.5}$$

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

(5.2) den , $W_{2M}^{2ux} = W_M^{ux}$,bu yüzden (5.5) aş.formda ifade edilebilir.

$$F(u) = \frac{1}{2}\left[\frac{1}{M} \sum_{x=0}^{M-1} f(2x) W_M^{ux} + \frac{1}{M} \sum_{x=0}^{M-1} f(2x+1) W_M^{ux} W_{2M}^{u}\right] \tag{5.6}$$

$u = 0,1,2,..., M-1$ için aş.tanımlama ile;

$$F_{çift}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x)W_M^{ux} \tag{5.7}$$

ve tekrar $u = 0,1,2,..., M-1$ için ;

$$F_{tek}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(2x+1)W_M^{ux} \tag{5.8}$$

(5.6) yı indirgersek;

$$F(u) = \frac{1}{2}\left[F_{çift}(u) + F_{tek}(u)W_M^{ux}\right] \tag{5.9}$$

Aynı zamanda $W_M^{u+M} = W_M^u$ ve $W_{2M}^{u+M} = -W_{2M}^u$ için,(5.7) ve (5.9) bize aş. eşitliği verir.

$$F(u+M) = \frac{1}{2}\left[F_{çift}(u) - F_{tek}(u)W_{2M}^u\right] \tag{5.10}$$

(5.7)-(5.10) eşitlikleri üzerinde dikkatli bir inceleme,bu ifadelerden bazı ilginç özellikler çıkrmamızı sağlayacaktır.(5.9) ve (5.10) da belirtildiği gibi N-nokta dönüşümü orjinal ifadeyi iki kısma ayırarak ta hesaplanabilir. $F(u)$'nun ilk yarısının hesabı (5.7) ve (5.8)'de verilen iki $(N/2)$-nokta değerlendirmesini gerektirir. $F_{çift}(u)$ ve $F_{tek}(u)$ sonucu çıkan değerler $u = 0,1,2,..., (N/2-1)$ için $F(u)$'yu elde etmek üzere (5.9) da yerleştirilir.Diğer yarı yalnızca (5.10) dan,ek dönüşüm değerlendirmeleri olmadan hesaplanır.

Bu prosedürün işlemsel anlamını incelemek için,sırasıyla kompleks çarpma ve toplama sayılarını temsil edecek $m(n)$ ve $a(n)$ ifadelerini alalım.Bundan önce n pozitif tamsayı olmak üzere,örneklerin sayısı 2^n dir. Öncelikle $n = 1$ alalım.İki nokta dönüşümü $F(0)$'ı daha sonrada (5.10)'u takiben $F(1)$'i gerektirir. $F(0)$ 'ı elde etmek için $F_{tek}(0)$ ve $F_{çift}(0)$'in hesaplanması gerekir.Bu durumda $M = 1$ ve (5.7),(5.8) eşitlikleri bir-nokta dönüşümleridir.Çünkü bir tek noktanın Fourier dönüşümü kendisinin örneğidir.Böylece $F_{çift}(0)$ ve $F_{tek}(0)$'ı elde etmek için çarpma yada toplama gerekli değildir. $F_{tek}(0)$'ın W_2^0 ile bir çarpımı ve bir toplama (5.9)'dan $F(0)$'ı verir.Böylece $F(1)$,fazladan bir toplama (çıkarma toplama ile aynı kabul edilir)ile (5.10)'un sonucu olarak elde edilir. $F_{tek}(0)W_2^0$'ın önceden hesaplanmış olması durumunda, iki-nokta dönüşümünün toplam işlem sayısı

$m(1) = 1$ adet çarpma, $a(1) = 2$ adet toplamadır.

n için bir sonraki değer 2 olsun.Yukarıdaki gelişmelere uygun olarak dört-nokta dönüşümü iki parçaya bölünebilir. $F(u)$'nun ilk yarısı iki için hesaplama (iki-nokta dönüşümü),gerektirir($M = 2$ için (5.7)ve (5.8) de olduğu gibi).İki nokta dönüşümü $m(1)$ adet çarpma ve $a(1)$ adet toplama gerektirir,bu yüzden bu iki denklemin hesaplanması toplamda $2m(1)$ çarpma ve $2a(1)$ toplama gerektirir. $F(0)$ ve $F(1)$'i (5.9) dan elde etmek için iki fazla toplama ve çarpma gerekmektedir.Çünkü $u = \{0,1\}$ için $F_{tek}(u)W_{2M}^{u}$ değeri önceden hesaplandığı için,fazladan iki toplama $F(2)$ ve $F(3)$'ü verir.Toplam alınırsa; $m(2) = 2m(1) + 2$ ve $a(2) = 2a(1) + 4$ olur.

n;3 e eşit olduğunda ,iki dört-nokta dönüşümü $F_{tek}(u)$ ve $F_{çift}(u)$'nun hesaplanmasında işleme katılır.Bu işlemler $2m(2)$ çarpma ve $2a(2)$ toplama gerektirir.Dört fazla çarpma ve sekiz fazla toplama ile dönüşüm tamamlanır.Toplam alınırsa; $m(3) = 2m(2) + 4$ ve $a(3) = 2a(2) + 8$ olur.

n'nin herhangi pozitif tamsayı değerleri için hesaplamalara devam edilirse,FFT'yi tamamlamak için gereken toplama ve çarpmaların sayısı için tekrarlanan ifadelerle karşılaşılır:

$$m(n) = 2m(n-1) + 2^{n-1} \qquad n \geq 1 \qquad (5.11)$$

$$a(n) = 2a(n-1) + 2^{n} \qquad n \geq 1 \qquad (5.12)$$

burada $m(0) = 0$ ve $a(0) = 0$ dır,çünkü tek noktanın dönüşümü çarpma ya da toplama gerektirmez.

(5.7)-(5.10) denklemlerinin uygulaması bir ardıl ikiye katlama FFT algoritması oluşturur.Bu isim ,iki adet bir-nokta dönüşümünden bir adet iki-nokta dönüşümü,iki adet iki-nokta dönüşümünden bir adet dört-nokta dönüşümü hesaplanmasından gelmektedir.Böylece devam edilirse,herhangi N için 2 nin kuvvetine eşit olacak sayıda dönüşüm gerekecektir.

5.2 İşlemlerin sayısı

Kompleks çarpma ve toplama işlem sayısı tümevarım ile FFT algoritmasının sırasıyla aşağıda ki uygulamalarını gerektirir.

$$\begin{aligned} m(n) &= \frac{1}{2} 2^n \log_2 2^n \\ &= \frac{1}{2} N \log_2 N \\ &= \frac{1}{2} Nn \quad n \geq 1 \end{aligned} \tag{5.13}$$

$$\begin{aligned} a(n) &= 2^n \log_2 2^n \\ &= N \log_2 N \\ &= Nn \quad n \geq 1 \end{aligned} \tag{5.14}$$

Tümevarım ile ispat etmek için (5.13) ve (5.14) eşitliklerinin $n = 1$ için doğru olduğu gösterilir;

$$m(1) = \frac{1}{2}(2)(1) = 1 \text{ ve } a(1) = (2)(1) = 2$$

Sonra ifadenin n için doğru olduğu kabul edilerek ,$n + 1$ için doğruluğu ispatlanır. (5.11)den;

$$m(n + 1) = 2m(n) + 2^n$$

n için geçerli olduğu kabul edilerek $m(n)$, (5.13) te yerine konulursa;

$$\begin{aligned} m(n + 1) &= 2\left(\frac{1}{2} Nn\right) + 2^n \\ &= 2\left(\frac{1}{2} 2^n n\right) + 2^n \\ &= 2^n(n + 1) \\ &= \frac{1}{2}(2^{n+1})(n + 1) \end{aligned}$$

böylece (5.13) eşitliği n 'nin bütün pozitif tamsayı değerleri için geçerli olur. (5.12)'den;

$$a(n + 1) = 2a(n) + 2^{n+1}$$

$a(n)$'i (5.14) te yerine yazılırsa ispat tamamlanmış olur.

$$\begin{aligned} a(n+1) &= 2Nn + 2^{n+1} \\ &= 2(2^n n) + 2^{n+1} \\ &= 2^{n+1}(n+1) \end{aligned}$$

5.3 Ters FFT

Ayrık ileri dönüşümü sağlayan herhangi bir algoritma aynı zamanda girdilerde çok küçük değişikliklerle tersini hesaplamak için uygulanabilir.bunu göstermek için (2.2) ve (2.3) eşitliklerine dönelim.Tekrar edersek;

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\exp[-j2\pi ux/N] \tag{5.15}$$

$$f(x) = \sum_{u=0}^{N-1} F(u)\exp[j2\pi ux/N] \tag{5.16}$$

(5.16)'nın kompleks konjugate ni alarak her iki tarafı N ile bölersek;

$$\frac{1}{N} f^*(x) = \frac{1}{N}\sum_{u=0}^{N-1} F^*(u)\exp[-j2\pi ux/N] \tag{5.17}$$

Bu sonucun (5.15) ile karşılaştırılması (5.17) de eşitliğin sağ tarafının ileri Fourier dönüşümü formunda olduğunu gösterir.İleri Fourier dönüşümünü hesaplamak üzere tasarlanan algoritmada $F^*(u)$ yu yerleştirmek , $f^*(x)/N$ miktarını verir.Kompleks konjugate in alınması ve N ile bölünmesi $f(x)$in tersini verir.

2-D kare dizi için (2.10)'un kompleks konjugate i alınırsa,

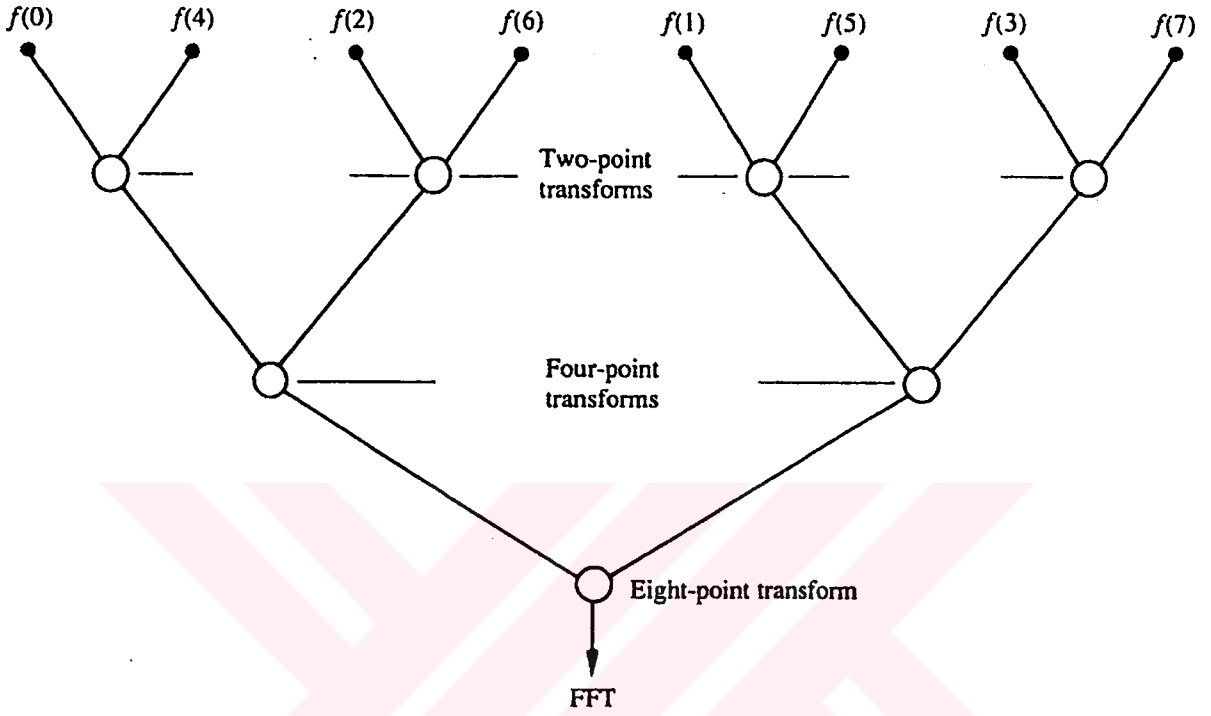
$$f^*(x, y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F^*(u, v)\exp[-j2\pi(ux + vy/N)] \tag{5.18}$$

elde edilir.Bu ifade (2.9) un 2-D ileri dönüşümünün formundadır.Bu yüzden ileri dönüşümü hesaplamak için hazırlanan bir algoritmada $F^*(u, v)$ 'nin yerleştirilmesi, $f^*(x, y)$'yi verir.Bu ifadenin Kompleks konjugate inin alınması $f(x.y)$'yi verecektir. $f(x)$ ya da $f(x.y)$ reel olduğunda Kompleks konjugate alınması gereksizdir,çünkü reel fonksiyonlarda $f(x) = f^*(x)$ ve $f(x, y) = f^*(x, y)$'dir.

1-D dönüşümden ardıl geçişle 2-D dönüşümün hesaplanması ,ters alınırken önceki tekniğin kullanılması durumunda sık görülen hataların kaynağıdır. Başka bir deyişle 2-D ters için, 1-D algoritması kullanıldığında ,herbir satır ve sütunda ilerledikten sonra Kompleks konjugate hesaplanmaz.

5.4 Uygulama

FFT algoritmasının bilgisayar uygulaması Bölüm 4.1 de açık bir şekilde geliştirilmiştir.Burada en önemli nokta (5.7) ve (5.8) in ardıl uygulaması için gereken şekilde girdi bilgilerinin düzenlenmesidir.Düzenleme prosedürü basit bir örnekle açıklanabilir. $\{f(0), f(1),....., f(7)\}$ sekiz-nokta fonksiyonunun FFT sini hesaplamak için ardıl ikiye katlama algoritması kullanıldığı düşünülürse,(5.7) eşitliği ;çift elemanlar için kullanılırken $\{f(0), f(2), f(4), f(6)\}$,(5.8) eşitliği; tek elemanlar için kullanılır $\{f(1), f(3), f(5), f(7)\}$ .Her dört-nokta dönüşümü,(5.7) ve (5.8)'in kullanılmasını gerektiren iki adet iki-nokta dönüşümü olarak hesaplanır.İlk kısmın FFT sinin hesaplanması için ,tek kısım $\{f(0), f(4)\}$ ve çift kısım $\{f(2), f(6)\}$ olmak üzere iki kısma ayrılması gerekir.Benzer şekilde ikinci kısımda (5.7) için $\{f(1), f(5)\}$ ve (5.8) için $\{f(3), f(7)\}$ olmak üzere ikiye ayrılır.Daha fazla düzenlemeye gerek yoktur çünkü herbir parça bir çift bir de tek olmak üzere iki elemandan oluşmaktadır.Bu sonuçların birleştirilmesi girdi dizisinin $\{f(0), f(4), f(2), f(6), f(1), f(5), f(3), f(7)\}$ formunda ifade edilmesini gerektirir. Ardıl ikiye katlama algoritması bu dizi üzerinde Şekil 22 de gösterildiği gibi çalışır.Hesaplamanın ilk aşaması $\{f(0), f(4)\}$, $\{f(2), f(6)\}$, $\{f(1), f(5)\}$ ve $\{f(3), f(7)\}$ yi kapsayan dört iki-nokta dönüşümleridir.Bir sonraki aşamada bu sonuçlar iki dört-nokta dönüşümleri için kullanılır,son aşamada ise bu iki sonuç istenen dönüşüm için kullanılır.



Şekil 5.1 Girdi dizisinin düzenlenmesi ve ardıl ikiye katlama metodunda kullanılışı

Ne yazık ki,girdi dizisinin yeniden düzenlenmesi için genel prosedür bit-ters dönme yöntemini izler. $x, f(x)$te herhengi bir geçerli değer olarak tanımlandığında ,yeniden düzenlenecek dizideki uygun eleman x'in ikili(binary) tanımlanması ve bitlerin ters çevrilmesi ile elde edilir.Örneğin $N = 2^3$ olduğunda ,ilk dizideki yedinci eleman $f(6)$,yeniden düzenlenen dizide dördüncü eleman olur,çünkü $6 = 110_2$;bitler ters çevrildiğinde $011_2 = 3$ olur.İkili numarada bu soldan sağa bir ters çevirmedir . Tablo 5.2; $N = 8$için olan prosedürü göstermektedir.Eğer yeniden düzenlenen dizi FFT'nin hesaplanmasında kullanılırsa ,Fourier dönüşümünün elemanları doğru sırada olacaktır.Tersini söylemek gerekirse eğer dizi ilk haliyle kullanılırsa ,sonuç ters dönmüş bitte olur.Aynı yorumlar ters dönüşümün hesaplanması için de geçerlidir.

Çizelge 5.2 Bit-ters çevirme örneği ve FFT algoritmasındaki girdiler için dizinin yeniden düzenlenmesi

Orijinal Diziliş			Orijinal Dizi	Ters Dönmüş Bitteki Diziliş			Yeniden Düzenlenmiş Dizi
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	$f(7)$

6. SONUÇLAR

Fourier dönüşümü yardımı ilı resimlerin geliştirilmesi,kullanılan resim işleme sistemlerinde başarı ile kullanılan bir sistemdir.Resim verilerindeki yüzde 10 un altında olan kayıplarda çok iyi sonuçlar vermektedir.Görüntünün elde edilmesinde kullanılan tekniklerden kaynaklanan bozulmaları kolayca yok etmesinin yanında,görüntünün karakteristik özelliklerini tamamen korumaktadır.

Sistemin zayıf noktası yarı yarıya bozulmaya uğramış resimlerde FFT nin yeterince başarılı olamamasıdır,bunun sebebi de sistemin elde ettiği girdiler üzerinde yaptığı iyileştirme ,giriş verisinin kalitesiyle doğru orantılıdır.

KAYNAKLAR

Andrews, H.C.,ve Hunt, BR. [1977].Digital Image Restoration,Prentice-Hall,Englewood Cliffs, N.J.

Anuta, P.F . [1969]. "Digital Registration of Multispectral Video Imagery. " Soc. Photo-Optical Instrum. Engs. ,vol. 7 , pp. 168-175.

A. Jain Fundamentals of Digital Image Processing, Prentice-Hall, 1989, pp 15 - 20.

A. Marion An Introduction to Image Processing, Chapman and Hall, 1991, Chap. 9.

B. Horn Robot Vision, MIT Press, 1986, Chaps 6, 7.

Chang, S.K.[1989].Principles of Pictorial ınformation Systems Design, Prentice-Hall, Englewood Cliffs, N.J.

D. Ballard and C. Brown Computer Vision, Prentice-Hall, 1982, pp 24 - 30.

R. Gonzales, R. Woods Digital Image Processing, Addison-Wesley Publishing Company, 1992, pp 81 - 125.

EKLER

FFT Hesaplamada Uygulama

```
#include
#include
#include
#include

#define FFTSIZE 1024
#define PPP 4
#define LGPPP 2
#define LGFFTSIZE 10
#define PPPSQ 512

main()
/* 2d fft cannot exceed 1kx1k */
{
   plural float rin[PPPSQ], iin[PPPSQ], rwork[PPPSQ], iwork[PPPSQ],
rout[PPPSQ],
                iout[PPPSQ], rtwiddles[LGFFTSIZE], itwiddles[LGFFTSIZE];
   register plural float zero, one;
   register int i;
   struct timeval beg, end;
   double stime, etime, eltime;

   gettimeofday(&beg,0);

   zero    = 0.0;
   one     = 1.0;

   for(i = 0; i < PPPSQ; i++)
   {
      rin[i] = zero;
      iin[i] = zero;
   }

   if(iproc == 0)  rin[3] = one;

   mpipl_cfft2df (rin, iin, rout, iout, rwork, iwork, 10);

   gettimeofday(&end,0);
   stime  = beg.tv_sec + 0.000001*beg.tv_usec;
   etime  = end.tv_sec + 0.000001*end.tv_usec;
   eltime = (etime - stime);
   printf("Elapsed time = %f\n",eltime);

}

mpipl_cfft2df(rinput, iinput, routput, ioutput, rwork, iwork, lgfftsize)
plural float *rinput, *iinput, *ioutput, *routput, *rwork, *iwork;
int lgfftsize;

{
   register int lgxppp,xppp, lgyppp, yppp,i, step;
   plural float rtwiddles[10];
   plural float itwiddles[10];
   register int temp;

   initialize_twiddles(rtwiddles, itwiddles, lgfftsize);
```

```
   step   = 1;
   xppp   = (1 << lgfftsize) / nxproc;
   yppp   = (1 << lgfftsize) / nyproc;
   lgxppp = 0;
   for (i = 1; i < xppp; i <<= 1) lgxppp++;
   lgyppp = 0;
   for (i = 1; i < yppp; i <<= 1) lgyppp++;

   temp   = 0;
   for (i = 0; i < yppp; i += 1)
   {
      mpipl_cfft1drowf(rinput + temp, iinput + temp, routput + temp,
                       ioutput + temp, rwork, iwork, lgxppp, lgfftsize,
                       step, rtwiddles, itwiddles);
      temp += xppp;
   }

   for (i = 0; i < xppp; i += 1)
   {
      mpipl_cfft1dcolf(routput + i, ioutput + i, routput + i, ioutput + i,

rwork,iwork,lgyppp,lgfftsize,xppp,rtwiddles,itwiddles);
   }

}

mpipl_cfft1dcolf(rinput, iinput, routput, ioutput, rwork, iwork, lgppp,
                 lgfftsize, step, rtwid, itwid)
plural float *rinput, *iinput, *routput, *ioutput, *rwork, *iwork,
             rtwid[10], itwid[10];
int lgppp, step, lgfftsize;

{
   register int stage, i;

   stage   = 0;

   compute_fftcol_stage(rinput, iinput, routput, ioutput, lgppp,
                        lgfftsize, step, stage, rtwid, itwid);

   for(stage = 1; stage < lgfftsize; stage++)
   {
      compute_fftcol_stage(routput, ioutput, routput, ioutput, lgppp,
                           lgfftsize, step, stage, rtwid, itwid);
   }


col_bit_reverse_permute(routput,ioutput,rwork,iwork,lgppp,lgfftsize,step);

}

col_bit_reverse_permute(routput, ioutput, rwork, iwork, lgppp, lgfftsize,
step)
plural float *routput, *ioutput, *rwork, *iwork;
int lgppp, lgfftsize, step;

{
   register plural int jxproc, jyproc, group, groupadd, groupdif;
   register plural float rswap, iswap;
   register plural int groupind, groupnum, index, rindex, destpe, destmem;
   register int bit, pppm1, zerooff, ppp;
```

```
    ppp      = 1 << lgppp;
    pppm1    = ppp - 1;
    group    = nyproc >> lgppp;
    jxproc   = ixproc;
    jyproc   = iyproc;
    groupind = jyproc&(group-1);
    groupnum = jyproc/group;
    groupadd = jyproc * ppp;

    for(zerooff = 0; zerooff < ppp; zerooff++)
    {
       groupdif = (groupnum + zerooff) & pppm1;
       rswap    = *(routput + step*groupdif);
       iswap    = *(ioutput + step*groupdif);
       index    = groupadd + groupdif;
       rindex   = 0;

       for (bit = 0; bit < lgfftsize; bit++)
       {
          rindex <<= 1;
          rindex  += index & 1;
          index  >>= 1;
       }

       destpe  = (rindex >> lgppp)*nxproc + jxproc;
       destmem =  rindex & pppm1;
       router[destpe].rswap    = rswap;
       router[destpe].iswap    = iswap;
       router[destpe].destmem  = destmem;
       *(rwork + step*destmem) = rswap;
       *(iwork + step*destmem) = iswap;
    }

    bit   = ppp*step;

    for(zerooff = 0; zerooff < bit; zerooff+= step)
    {
       *(routput + zerooff) = *(rwork + zerooff);
       *(ioutput + zerooff) = *(iwork + zerooff);
    }

}

mpipl_cfft1drowf(rinput, iinput, routput, ioutput, rwork, iwork, lgppp,
                 lgfftsize, step, rtwid, itwid)
plural float *rinput, *iinput, *routput, *ioutput, *rwork, *iwork,
rtwid[10],
             itwid[10];
int lgppp, step, lgfftsize;

{
    register int stage, i;

    stage = 0;

    compute_fftrow_stage(rinput, iinput, routput, ioutput, lgppp, lgfftsize,
                         step, stage, rtwid, itwid);

    for(stage = 1; stage < lgfftsize; stage++)
    {
       compute_fftrow_stage(routput, ioutput, routput, ioutput, lgppp,
```

```
                        lgfftsize, step, stage, rtwid, itwid);
    }

row_bit_reverse_permute(routput,ioutput,rwork,iwork,lgppp,lgfftsize,step);

}

row_bit_reverse_permute(routput, ioutput, rwork, iwork, lgppp, lgfftsize,
step)
plural float *routput, *ioutput, *rwork, *iwork;
int lgppp, lgfftsize, step;

{
    register plural int jyproc, jxproc, group, groupadd, groupdif;
    register plural float rswap, iswap;
    register plural int groupind, groupnum, index, rindex, destpe, destmem;
    register int bit, pppm1, zerooff, ppp;

    ppp      = 1 << lgppp;
    pppm1    = ppp - 1;
    group    = nxproc >> lgppp;
    jyproc   = iyproc * nxproc;
    jxproc   = ixproc;
    groupind = jxproc&(group-1);
    groupnum = jxproc/group;
    groupadd = jxproc * ppp;

    for(zerooff = 0; zerooff < ppp; zerooff++)
    {
        groupdif = (groupnum + zerooff) & pppm1;
        rswap    = *(routput + step*groupdif);
        iswap    = *(ioutput + step*groupdif);
        index    = groupadd + groupdif;
        rindex   = 0;

        for(bit = 0; bit < lgfftsize; bit++)
        {
            rindex <<= 1;
            rindex  += index & 1;
            index  >>= 1;
        }

        destpe  = (rindex >> lgppp) + jyproc;
        destmem =  rindex & pppm1;
        router[destpe].rswap    = rswap;
        router[destpe].iswap    = iswap;
        router[destpe].destmem  = destmem;
        *(rwork + step*destmem) = rswap;
        *(iwork + step*destmem) = iswap;
    }

    bit = ppp*step;

    for(zerooff = 0; zerooff < bit; zerooff += step)
    {
        *(routput + zerooff) = *(rwork + zerooff);
        *(ioutput + zerooff) = *(iwork + zerooff);
    }

}
```

```
compute_fftrow_stage(rinput, iinput, routput, ioutput, lgppp, lgfftsize,
                     step, stage, rtwid, itwid)
plural float *rinput, *iinput, *routput, *ioutput, rtwid[10], itwid[10];
int  lgppp, lgfftsize, step, stage;

{
   register plural float tempr, tempi, temprx, tempix, tempro, tempio;
   register plural float tempx;
   register plural unsigned int jhcommd;
   register plural int tb, subindex, index, jxproc;
   register plural float zero, one;
   register int i, sub1, hcommd, layer, ppp, base_layer, off_layer,
                commd, sub2;
   unsigned int comtst1, comtst2;

   zero    = 0.0;
   one     = 1.0;
   ppp     = 1 << lgppp;
   jxproc  = ixproc*ppp;
   commd   = 1 << (lgfftsize - stage - 1);
   comtst1 = (1 << (lgfftsize - stage)) - 1;
   comtst2 = commd - 1;
   hcommd  = commd/ppp;
   jhcommd = ixproc%(hcommd << 1) < hcommd;

   if(hcommd > 0)
   {
      sub1 = 0;
      for(layer = 0; layer < ppp; layer++)
      {
         temprx = *(rinput + sub1);
         tempix = *(iinput + sub1);

         xnetW[hcommd].tempx = temprx;
         xnetE[hcommd].tempr = temprx;
         if (jhcommd)  tempr = tempx;

         xnetW[hcommd].tempx = tempix;
         xnetE[hcommd].tempi = tempix;
         if (jhcommd)  tempi = tempx;

         tempro = temprx;
         tempio = tempix;

         if(!jhcommd)
         {
            tempro = -tempro;
            tempio = -tempio;
         }

         tempro += tempr;
         tempio += tempi;

         index = jxproc + layer;
         tb = index & comtst1;
         subindex = index & comtst2;

         tempr = one;
         tempi = zero;

         if(tb != subindex)
         {
```

```
            subindex <<= stage;

            for(i = 0; i < lgfftsize; i++)
            {
               if((subindex >> i) & 1)
               {
                  tempx = tempr;
                  tempr = tempr * rtwid[i] - tempi * itwid[i];
                  tempi = tempi * rtwid[i] + tempx * itwid[i];
               }
            }
         }

         if(!jhcommd)
         {
            tempx  = tempro;
            tempro = tempr*tempx  - tempio*tempi;
            tempio = tempr*tempio + tempi*tempx;
         }

         *(routput + sub1) = tempro;
         *(ioutput + sub1) = tempio;
         sub1              += step;
      }
   }

   else
   {
      for(base_layer = 0; base_layer < ppp; base_layer += 2*commd)
      {
         for(off_layer = 0; off_layer < commd; off_layer++)
         {
            sub1 = base_layer + off_layer;
            sub2 = sub1 + commd;
            sub1 *= step;
            sub2 *= step;

            *(routput + sub1) = *(rinput + sub1) + *(rinput + sub2);
            *(ioutput + sub1) = *(iinput + sub1) + *(iinput + sub2);

            tempix = *(iinput + sub1) - *(iinput + sub2);
            temprx = *(rinput + sub1) - *(rinput + sub2);

            layer = base_layer + off_layer + commd;
            index = jxproc + layer;
            tb = index & comtst1;
            subindex = index & comtst2;

            tempr = one;
            tempi = zero;

            if(tb != subindex)
            {
               subindex <<= stage;

               for(i = 0; i < lgfftsize; i++)
               {
                  if((subindex >> i) & 1)
                  {
                     tempx = tempr;
                     tempr = tempr * rtwid[i] - tempi * itwid[i];
                     tempi = tempi * rtwid[i] + tempx * itwid[i];
```

```
                    }
                 }
              }

              *(routput + sub2 ) = tempr * temprx - tempi * tempix;
              *(ioutput + sub2 ) = tempi * temprx + tempr * tempix;
           }
        }
     }

}

compute_fftcol_stage(rinput, iinput, routput, ioutput, lgppp, lgfftsize,
                     step, stage, rtwid, itwid)
plural float *rinput, *iinput, *routput, *ioutput, rtwid[10], itwid[10];
int  lgppp, lgfftsize, step, stage;

{
   register plural float tempr, tempi, temprx, tempix, tempro, tempio;
   register plural float tempx;
   register plural unsigned int jhcommd;
   register plural int tb, subindex, index, jyproc;
   register plural float zero, one;
   register int i, sub1, hcommd, layer, ppp, base_layer, off_layer,
                commd, sub2;
   unsigned int comtst1, comtst2;

   ppp     = 1 << lgppp;
   zero    = 0.0;
   one     = 1.0;
   jyproc  = iyproc*ppp;
   commd   = 1 << (lgfftsize - stage - 1);
   comtst1 = (1 << (lgfftsize - stage)) - 1;
   comtst2 = commd - 1;
   hcommd  = commd/ppp;
   jhcommd = iyproc%(hcommd << 1) < hcommd;

   if(hcommd > 0)
   {
      sub1 = 0;
      for(layer = 0; layer < ppp; layer++)
      {
         temprx = *(rinput + sub1);
         tempix = *(iinput + sub1);

         xnetN[hcommd].tempx = temprx;
         xnetS[hcommd].tempr = temprx;
         if (jhcommd)  tempr = tempx;

         xnetN[hcommd].tempx = tempix;
         xnetS[hcommd].tempi = tempix;
         if (jhcommd)  tempi = tempx;

         tempro = temprx;
         tempio = tempix;

         if(!jhcommd)
         {
            tempro = -tempro;
            tempio = -tempio;
         }
```

```
        tempro += tempr;
        tempio += tempi;

        index = jyproc + layer;
        tb = index & comtst1;
        subindex = index & comtst2;

        tempr = one;
        tempi = zero;

        if(tb != subindex)
        {
           subindex <<= stage;

           for(i = 0; i < lgfftsize; i++)
           {
              if((subindex >> i) & 1)
              {
                 tempx = tempr;
                 tempr = tempr * rtwid[i] - tempi * itwid[i];
                 tempi = tempi * rtwid[i] + tempx * itwid[i];
              }
           }
        }

        if(!jhcommd)
        {
           tempx  = tempro;
           tempro = tempr*tempx  - tempio*tempi;
           tempio = tempr*tempio + tempi*tempx;
        }

        sub1 += step;
     }
  }

  else
  {
     for(base_layer = 0; base_layer < ppp; base_layer += 2*commd)
     {
        for(off_layer = 0; off_layer < commd; off_layer++)
        {
           sub1 = base_layer + off_layer;
           sub2 = sub1 + commd;
           sub1 *= step;
           sub2 *= step;
           *(routput + sub1) = *(rinput + sub1) + *(rinput + sub2);
           *(ioutput + sub1) = *(iinput + sub1) + *(iinput + sub2);

           tempio = *(iinput + sub1) - *(iinput + sub2);
           tempro = *(rinput + sub1) - *(rinput + sub2);

           layer = base_layer + off_layer + commd;
           index = jyproc + layer;
           tb = index & comtst1;
           subindex = index & comtst2;

           tempr = one;
           tempi = zero;

           if(tb != subindex)
           {
```

```
                subindex <<= stage;

                for(i = 0; i < lgfftsize; i++)
                {
                   if((subindex >> i) & 1)
                   {
                      tempx = tempr;
                      tempr = tempr * rtwid[i] - tempi * itwid[i];
                      tempi = tempi * rtwid[i] + tempx * itwid[i];
                   }
                }
             }

             *(routput + sub2 ) = tempr * temprx - tempi * tempio;
             *(ioutput + sub2 ) = tempi * temprx + tempr * tempio;
          }
       }
    }

}

initialize_twiddles(rtwiddles, itwiddles, lgfftsize)
plural float rtwiddles[10], itwiddles[10];
int lgfftsize;

{
   register int i, j;
   register float two;
   float mipi, rbase, ibase, base;

   mipi = 3.14159265358979323;
   two  = 2.0;
   base = 1.0/((float)(1 << (lgfftsize - 1)));
   j    = 0;
   base *= mipi;
   rbase =  cos(base);
   ibase =  sin(base);

   rtwiddles[0] = rbase;
   itwiddles[0] = ibase;

   for(i = 1; i < lgfftsize; i++)
   {
      rtwiddles[i] = rtwiddles[j]*rtwiddles[j]-itwiddles[j]*itwiddles[j];
      itwiddles[i] = two * itwiddles[j] * rtwiddles[j];
      j           += 1;
   }

}
```

ÖZGEÇMİŞ

Doğum tarihi	31.01.1979	
Doğum yeri	İstanbul	
Lise	1992-1995	Özel Notre Dame De Sion Fransız Kız Lisesi
Lisans	1995-1999	Yıldız Teknik Üniversitesi Kimya-Metalurji Fak. Matematik Mühendisliği Bölümü
Yüksek Lisans	2000-2003	Yıldız Teknik Üniversitesi Kimya-Metalurji Fak. Matematik Mühendisliği Bölümü

Çalıştığı kurum(lar)

1999-2000 İSKO Makine
2000-Devam deiyor YTÜ Fen Bilimleri Enstitüsü Araştırma Görevlisi