Image transforms

Group(8)

Asmaa Gamal Abd-Elrahman

Hajar Mohammed Hassan

Nosiba Gamal Elbanna

Wavelet and other image transforms

The discrete Fourier transform is a member of an important class of linear transforms that include the Hartley, sine, cosine, Walsh-Hadamard, Slant, Haar, and wavelet transforms. These transforms, which are the subject of this chapter, decompose functions into weighted sums of orthogonal or biorthogonal basis functions, and can be studied using the tools of linear algebra and functional analysis. When approached from this point of view, images are vectors in the vector space of all images. Basis functions determine the nature and usefulness of image transforms. Transforms are the coefficients of linear expansions. And for a given image and transform (or set of basis functions), both the orthogonality of the basis functions and the coefficients of the resulting transform are computed using inner products. All of an image's transforms are equivalent in the sense that they contain the same information and total energy. They are reversible and differ only in the way that the information and energy is distributed among the transform's coefficients.

2D-Discrete Wavelet Transformation and its applications in Digital Image Processing using MATLAB

Spatial domain refers to the normal image space represented as a matrix of pixels. Transformation techniques in this domain are directly operated on image pixel values. The values are manipulated to achieve desired enhancement.

Frequency domain deals with the rate at which these pixel values change in spatial domain. Frequency simply refers to the rate of change of color components in an image. Areas of high frequencies experience rapid color changes, whereas parts that change gradually contain low frequencies.

-

Wavelet Transformation

- Okay, so, what exactly are wavelets, and why do we need this transformation? According to Wikipedia,

- *A **wavelet** is a wave-like oscillation with an amplitude that begins at zero, increases, and then decreases back to zero. It can typically be visualized as a "brief oscillation" like one recorded by a seismograph or heart monitor.*

Wavelet Transformation

- Wavelets are functions that are concentrated in time and frequency around a certain point. This transformation technique is used to overcome the drawbacks of fourier method. Fourier transformation, although it deals with frequencies, does not provide temporal details. According to *Heisenberg's Uncertainty Principle*, we can

Wavelet Transformation

- either have high frequency resolution and poor temporal resolution or vice versa.
- This wavelet transform finds its most appropriate use in non-stationary signals. This transformation achieves good frequency resolution for low-frequency components and high temporal resolution for high-frequency components.
- This method starts with a mother wavelet such as Haar, Morlet, Daubechies, etc. The signal is then essentially translated into scaled and shifted versions of mother wavelet.

Wavelet based Compression of Images

- Wavelet compression is a form of data compression well suited for image compression. The idea is to store image data in as little space as possible in a file. Wavelet compression either be lossless or lossy.

- In this section we will use wavelet decomposition for compression of images. We use thresholding on detail components after performing a 4-level decomposition. This is done by sorting the wavelet coefficients and retain the first 20%, 10%, 1% and 0.5% largest coefficients and threshold everything else out to zero.

- The code below shows the implementation of image compression using a 4-level wavelet decomposition in MATLAB. This can be done easily with the help of inbuilt methods such as wavedec2() and waverec2() to decompose and reconstruct images to/from sub-signals.

```
clear all;
close all;
clc;

A = imread('poly.png');
B = rgb2gray(A);

%% Wavelet Compression
[C,S] = wavedec2(B, 4, 'db1');
Coeff_sort =sort(abs(C(:)));

count = 1;
for keep = [.1 .05 .01 .005]
     subplot(2,2,count)
     thresh = Coeff_sort(floor((1-keep)*length(Coeff_sort)));
     index = abs(C)>thresh;
     C_filter = C.*index;

     % Reconstruction
     Areacon = uint8(waverec2(C_filter, S, 'db1'));
     imshow(uint8(Areacon))
     title(['', num2str(keep*100),'%'], 'FontSize',12)
     count = count+1;
end
set(gcf, 'Position', [1750 100 1750 2000])
```

The effect of various values of thresholding can be observed below.

20%

10%

1%

0.5%

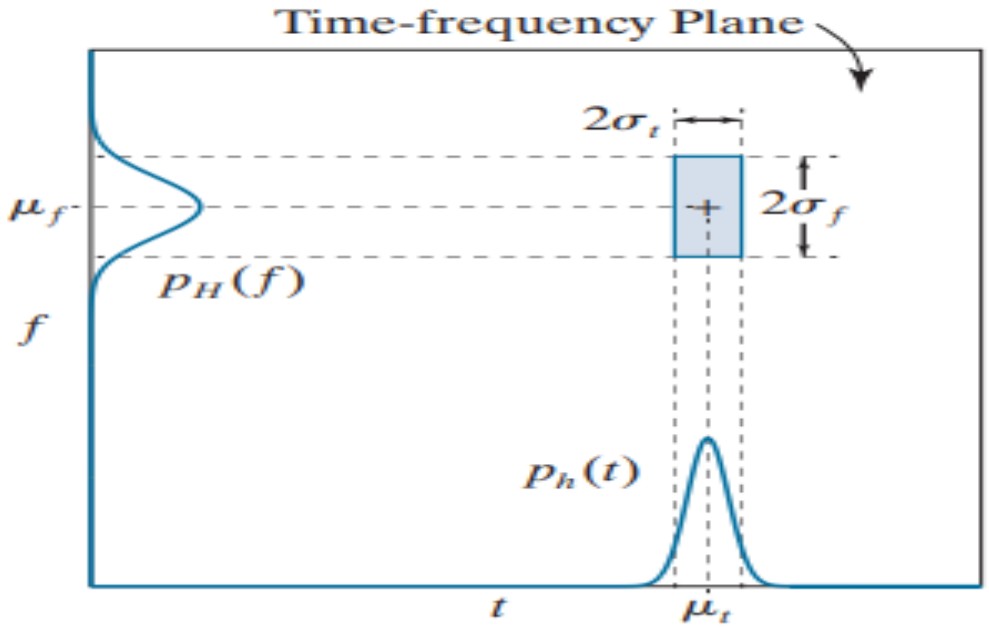
Wavelet Transformation

- Since we are retaining only a set of coefficients at each level, this indirectly acts as a frequency domain filter (low-pass, high-pass, band-pass, etc.) which is generally used in fourier transformations. Wavelet compression is a better version of fourier compression since we’re dealing with wavelets.

BASIS FUNCTIONS IN THE TIME-FREQUENCY PLANE

- The majority of the energy falls in a rectangular region ,called a Heisenberg box or cell
- Since the support of a function can be defined as the set of points where the function is nonzero, Heisenberg's uncertainty principle tells us that it is impossible for a function to have finite support in both time and frequency.
- basis functions are scaled and shifted small waves, called wavelets
- each wavelet basis function is characterized by a unique spectrum and location in time.

BASIS FUNCTIONS IN THE TIME-FREQUENCY PLANE



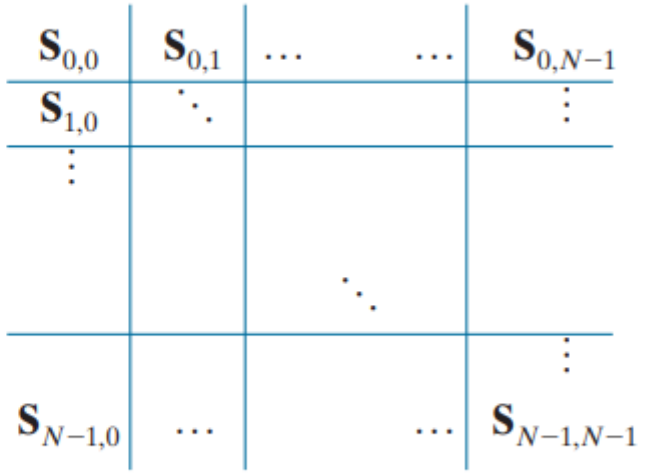
where f denotes frequency and H(f) is the Fourier transform of h(t).
Then the energy of basis function h, as illustrated in Fig is concentrated at (μt, μf) on the time-frequency plane.

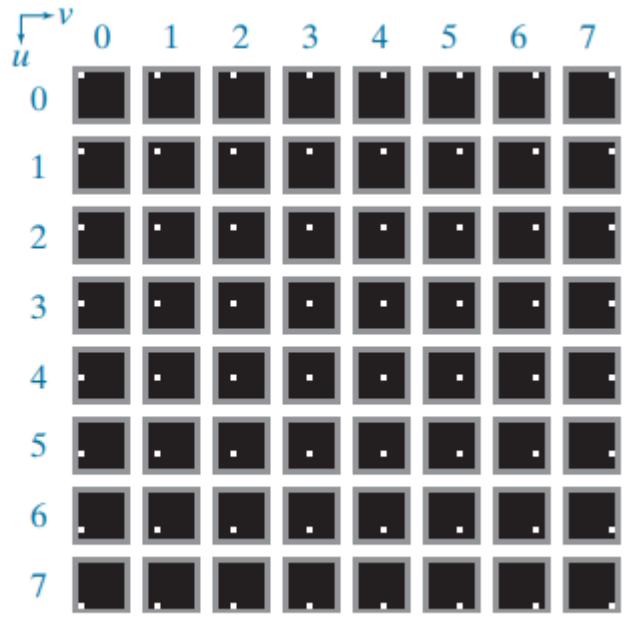
BASIS IMAGES

In the context of digital image processing, F is a 2-D image, and the S(u,v) are called basis images. They can be arranged in an N*N array, as shown in Fig. 7.6(a), to provide a concise visual representation of the 2-D basis functions they represent.

a b

FIGURE 7.6
(a) Basis image organization and (b) a standard basis of size 8×8. For clarity, a gray border has been added around each basis image. The origin of each basis image (i.e., $x = y = 0$) is at its top left.





FOURIER-RELATED TRANSFORMS

- three Fourier-related transforms that are real rather than complex-valued the discrete Hartley transform, discrete cosine transform, and discrete sine transform.

- All three transforms avoid the computational complexity of complex numbers and can be implemented via fast FFT-like algorithms.

❑ THE DISCRETE HARTLEY TRANSFORM

- Its main distinction from the DFT is that it transforms real inputs to real outputs, with no intrinsic involvement of complex numbers.

- 8 8 * basis images of the two transforms are also similar. The basis images of maximum frequency occur when u and v are N/2 or 4.

FOURIER-RELATED TRANSFORMS

❑ THE DISCRETE COSINE TRANSFORM

- A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies.
- where small high-frequency components can be discarded .The use of cosine rather than sine functions is critical for compression
- compresses data in sets of discrete DCT blocks. DCT blocks can have a number of sizes, including 8x8 pixels for the standard DCT, and varied integer DCT sizes between 4x4 and 32x32 pixels.
- The DCT has a strong "energy compaction" property , capable of achieving high quality at high data compression ratios .However , blocky compression artifacts اثار/عيوبcan appear when heavy DCT compression is applied.

FOURIER-RELATED TRANSFORMS

- THE DISCRETE SINE TRANSFORM

DSTs are widely employed in solving partial differential equations by spectral methods, where the different variants of the DST correspond to slightly different odd/even boundary conditions at the two ends of the array

Hadamard Transform

The Hadamard Transform is also known as :

- ❑ Walsh-Hadamard transform
- ❑ Hadamard-Rademacher-Walsh transform

- It is an example of a generalized class of Fourier transforms.
- It performs an orthogonal , symmetric operations.
- The Hadamard transform H_m is a 2^m x 2^m matrix.
- Hadamard matrix transforms 2^m real numbers x_n into 2^m real numbers X_k

The Hadamard transform can be defined in two ways:

- ❑ Recursively
- ❑ Binary representation

Applications of Hadamard transform:

- signal Processing
- Data Compression
- Data Encryption
- In many scientific methods such as NMR, mass spectroscopy and crystallography etc.

Hadamard transform for 1D functions

- the kernel of 1D Hadamard transform is :

$$g(x,u)=\frac{1}{\sqrt{N}}(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)}$$

- 1D Hadamard transform H(u) can be return as :

$$H(u)=\frac{1}{N}\sum_{x=0}^{N-1} f(x)\ \ (-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)}$$

$$where\ N=2^n$$

- The inverse Hadamard transform :

$$f(x)=\frac{1}{N}\sum_{x=0}^{N-1} H(u)\ \ (-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)}$$

inverse kernel $h(x,u)$

Hadamard transform for 2D functions

- the kernel of 1D Hadamard transform is :

$$\boldsymbol{H(u,v)} = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \boldsymbol{f(x,y)}\ (-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)+b_i(y)b_i(u)]}$$

- The inverse Hadamard transform :

$$\boldsymbol{f(x,y)} = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \boldsymbol{H(u,v)}\ [\ \prod_{i=0}^{n-1}(-1)^{\sum_{i=0}^{n-1}[b_i(x)b_i(u)+b_i(y)b_i(u)]}]$$

- Hadamard kernel is:
  - ❑ Separable
  - ❑ Symmetric

It means $g(x,y,u,v) = g_1(x,u)\ g_2(y,v) = \ h_1(x,u)\ h_2(y,v)$

- The Hadamard transform of an image f is denoted as :

$$g = A \times f \times A$$

where the matrix A is related to Hadamard matrix as

$$A = \frac{1}{\sqrt{N}} H_{q_1 q}$$

$N \rightarrow$ dimension of an image

- The basic Hadamard matrix is $H_N = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

- The higher order Hadamard matrix is $H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$

EX: $H_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$

Hadamard Transform in MATLAB

- Hadamard transform:

```
function t=getHadamardTransform(im,N)
h=hadamard(N);
m=log2(N);
t=(1/(2^m))*h*im*h';
```

- Inverse Hadamard transform:

```
function t=getInvHadamardTransform(im,N)
h=hadamard(N);
m=log2(N);
t=(1/(2^m))*h'*im*h;
```

Walsh transform

- The Walsh transform is non-sinusoidal , orthogonal transformation technique that decomposes a signal into a set of basis functions.
- Sequence means number of sign changes in row.
- Walsh transform is related to Hadamard transform.
- Walsh transform applications
  - Power spectrum analysis
  - Filtering
  - Speech processing
  - Medical signals
  - Multiplexing and coding in communication
  - Logical design and analysis
  - Solving nonlinear differential equations etc.

- For 4×4 Hadamard matrix ,let us check sign change in row:

$$H_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Row 1= 1 1 1 1 ,sign changes=0

Row 2= 1 -1 1 -1 ,sign changes=3

Row 3= 1 1 -1 -1 ,sign changes=1

Row 4= 1 -1 -1 1 ,sign changes=2

- By arranging rows according to sign changes we got Walsh matrix

$$W_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

- **Walsh transform for 1D function:**

$$F = W \times f$$

- **Walsh transform for 2D function:**

$$F = W \times f \times W^T = W \times f \times W$$

since W matrix is symmetric

EX: $f(x) = \{1,2,0,4\}$ apply Walsh transform

For 1D $F = W \times f$

$$F = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \\ 2 \\ 4 \end{bmatrix}$$

Walsh transform in MATLAB

- Walsh transform:

```
function t=getWalshTransform(im,N)
h=walsh(N);
m=log2(N);
t=(1/(2^m))*h*im*h';
```

- Inverse Walsh transform:

```
function t=getInvWalshTransform(im,N)
h=walsh(N);
m=log2(N);
t=(1/(2^m))*h'*im*h;
```

Slant transform

- widely used in image compression.
- Orthogonal and very fast.
- Its kernel can be generated recursively like Hadamard transform.
- The slant transform of order 2×2 is defined as:

$$s_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- In general, an $N \times N$ matrix can be recursively given as:

$$s_N = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ a_N & b_N & & -a_N & b_N & \\ 0 & & I_{(\frac{N}{2})-2} & 0 & & I_{(\frac{N}{2})-2} \\ 0 & 1 & 0 & 0 & -1 & 0 \\ -b_N & a_N & & b_N & a_N & \\ 0 & & I_{(\frac{N}{2})-2} & 0 & & I_{(\frac{N}{2})-2} \end{bmatrix} \times \begin{bmatrix} s_{\frac{N}{2}} & 0 \\ 0 & s_{\frac{N}{2}} \end{bmatrix} \quad \text{eq(1)}$$

Here $b_N = (1 + 4a_{N-1}^2)^{-1/2}$ eq(2)

$a_N = 2 \times b_N \times a_{N-1}$ eq(3)

- I_N identity matrix

Let us generate kernel for $N = 4$

$$N = 2^n \ , \quad n = 2 \ , \ s_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$b_2 = (1 + 4(1))^{-1/2} = \frac{1}{\sqrt{5}}$$

$$a_2 = 2 \times \frac{1}{\sqrt{5}} \times 1 = \frac{2}{\sqrt{5}}$$

$$A_2 = \begin{bmatrix} 1 & 0 & & 1 & 0 & \\ a_N & b_N & 0 & -a_N & b_N & 0 \\ 0 & & I_{(\frac{N}{2})-2} & 0 & & I_{(\frac{N}{2})-2} \\ 0 & 1 & & 0 & -1 & \\ -b_N & a_N & 0 & b_N & a_N & 0 \\ 0 & & I_{(\frac{N}{2})-2} & 0 & & I_{(\frac{N}{2})-2} \end{bmatrix} \times \begin{bmatrix} A_1 & 0 \\ 0 & A_1 \end{bmatrix}$$

$$A_2 = \frac{1}{\sqrt{2}} \times \begin{bmatrix} 1 & 0 & 1 & 0 \\ \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ 0 & 1 & 0 & -1 \\ \frac{-1}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} \times \frac{1}{\sqrt{2}} \times \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Therefore, the final slant transform of order 4×4 is given as

$$A_2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ \frac{3}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} \\ 1 & -1 & -1 & 1 \\ \frac{1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} & \frac{3}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \end{bmatrix}$$

Matrix A is real , not symmetric ant unitary

- The forward slant transform given as

$$\mathrm{F} = A \times f$$

- The inverse slant transform given as

$$f = A^T \times F$$

Slant transform for 2D

- This logic can be extended to 2D images also
- The forward slant transform for 2D given as

$$\mathrm{F} = A \times f \times A^T$$

- The inverse slant transform for 2D given as

$$f = A^T \times F \times A$$

slant Transform in MATLAB

- Slant transform:

```
function t=getSlantTransform(im,N)
s=sltmtx(log2(N));
t=s*im*s';
```

- Inverse Slant transform:

```
function t=getInvSlantTransform(im,N)
s=sltmtx(log2(N));
t=s'*im*s;
```

Haar transform

- Haar proposes the Haar transform in 1910
- Due to its low computing requirement, the Haar transform has been mainly used for image processing and pattern recognition.
- Therefor two-dimensional signal processing is an area of efficient applications of Haar transforms due to their wavelet like structure.
- Haar transform is real and orthogonal
- It is very fast
- It is separable and symmetric.
- The basis vectors are sequency ordered.
- It has poor energy compaction for images.

- Elements +1 ,-1 ,0

Procedure to generate kernel of Haar transform:

1) Find the order N let $n = \log N$

2) Determine p & q

   - p ranges from 0 to n-1
   - if p=0 then q=0 or q=1 , else $1 \leq q \leq 2^p$

3) Determine value of K

   $K = 2^p + q - 1$

4) if k = 0

$$h_0(z) = h_{00}(z) = \frac{1}{\sqrt{N}} \; for \; z \in (0,1)$$

else $$h_{pq}(z) = \frac{1}{\sqrt{N}} \begin{cases} 2^p & if \; \frac{q-1}{2^p} \le z \le \frac{q-1/2}{2^p} \\ -2^p & if \; \frac{q-1/2}{2^p} \le z \le \frac{q}{2^p} \\ 0 & if z \in (0,1) \end{cases}$$

EX: base for N=2

If N=2, then n=log N =log 2 =1

So, the value of p=0 , therefore q=0,1

$$K = 2^p + q - 1 = \begin{cases} 0 \text{ for } p = 0, q = 0 \\ 1 \text{ for } p = 0, q = 1 \end{cases}$$

So, where k=0 , $h_0(z) = \frac{2^0}{\sqrt{2}} = \frac{1}{\sqrt{2}}$

where k=1 , $h_0(z) = \frac{-2^0}{\sqrt{2}} = \frac{-1}{\sqrt{2}}$

- Therefore 2×2 Haar matrix is given as :

$$A_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- 4×4 Haar matrix is given as :

$$A_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -\sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & -\sqrt{2} & -\sqrt{2} \end{bmatrix}$$

Haar Transform in MATLAB

- Haar Transform:

```
function t=getHaarTransform(im,N)
h=haarmtx(N);
m=log2(N);
t=h*im*h';
```

- Inverse Haar Transform :

```
function t=getInvHaarTransform(im,N)
h=haarmtx(N);
t=h'*im*h;
```