Discrete image transforms (Castleman, 2/e, Ch. 13)

$$y_i = \sum_{j=0}^{N-1} t_{ij} x_j \qquad \text{or} \qquad \underline{y} = \underline{\underline{T}}\,\underline{x} \tag{1}$$

linear transformation of vector $\underline{x}$

$\underline{\underline{T}}$: kernel matrix

y_i: inner product of input vector $\underline{x}$ with i-th row of $\underline{\underline{T}}$

If $\underline{\underline{T}}$ is a unitary matrix

$$\underline{\underline{T}}^{-1} = \underline{\underline{T}}^{*t}$$

and $$\underline{\underline{T}}\,\underline{\underline{T}}^{*t} = \underline{\underline{T}}^{*t}\,\underline{\underline{T}} = \underline{\underline{I}}$$

If $\underline{\underline{T}}$ is unitary and real it is an orthogonal matrix

$$\underline{\underline{T}}^{-1} = \underline{\underline{T}}^{t}$$

and $$\underline{\underline{T}}\,\underline{\underline{T}}^{t} = \underline{\underline{T}}^{t}\,\underline{\underline{T}} = \underline{\underline{I}} \tag{2}$$

Note that the i,j-th element of $\underline{\underline{T}}\,\underline{\underline{T}}^{t}$ is the inner product of rows i and j. By (2) this is zero unless $i=j$ in which case it is zero. $\Rightarrow$ the rows of $\underline{\underline{T}}$ are a set of orthonormal vectors.

Example: Discrete Fourier Transform

$$F_k = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} f_i\, e^{-j 2\pi k \frac{i}{N}} \qquad \text{or} \qquad \underline{F} = \underline{\underline{W}}\,\underline{f}$$

$\underline{\underline{W}}$ is a unitary (not orthogonal)

$$w_{ik} = \frac{1}{\sqrt{N}}\, e^{-j 2\pi k \frac{i}{N}}$$

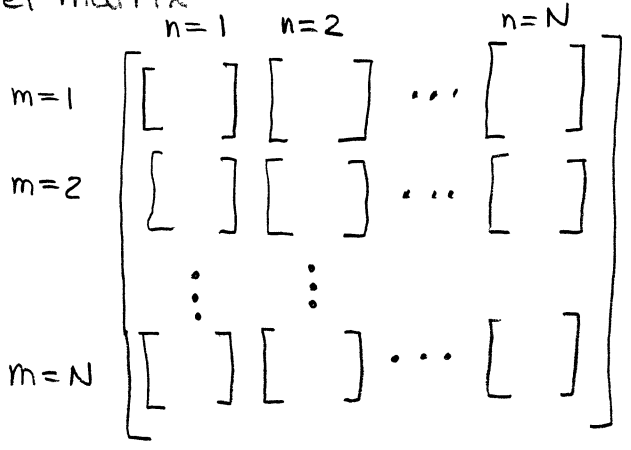
In general, the rows of $\underline{\underline{T}}$ (in this case $\underline{\underline{W}}$) form an orthonormal basis set for all $N \times 1$ vectors.

(1) is simply a coordinate transform which rotates $\underline{x}$ but does not change its length.

Two-dimensional discrete linear transformations

$$G_{mn} = \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} F_{ik} \underbrace{J(i,k,m,n)}_{\text{kernel function of the transform}}$$

kernel matrix is a block matrix



If $J(i,k,m,n) = T_r(i,m)\, T_c(k,m)$

then it is called separable and can be carried out as a rowwise operation followed by a column wise operation or vice versa.

$$G_{mn} = \sum_{i=0}^{N-1} T(i,m) \left[\sum_{k=0}^{N-1} F_{ik}\, T(k,n) \right] \quad \text{or} \quad \underline{G} = \underline{T}\,\underline{F}\,\underline{T}$$

Inverse transform is

$$\underline{F} = \underline{T}^{-1}\,\underline{G}\,\underline{T}^{-1} = \underline{T}^{*t}\,\underline{G}\,\underline{T}^{*t}$$

Example: 2-D DFT

$$\underline{T} = {}^{\alpha}\underline{W}$$

$$\underline{T}^{-1} = {}^{\alpha}\underline{W}^{*t}$$

$$\therefore \quad \underline{G} = {}^{\alpha}\underline{W}\,\underline{F}\,{}^{\alpha}\underline{W} \quad \text{and} \quad \underline{F} = {}^{\alpha}\underline{W}^{*t}\,\underline{G}\,{}^{\alpha}\underline{W}^{*t}$$

Orthogonal transforms

Many transforms used in image processing have only real elements in their kernel matrix.

This is then an orthogonal matrix

The inverse transform is

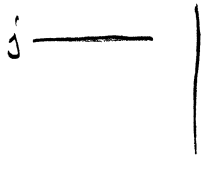
$$\underline{F} = \underline{T}^t \underline{G} \underline{T}^t$$

If T is symmetric

$$\underline{G} = \underline{T}\underline{F}\underline{T} \quad \text{and} \quad \underline{F} = \underline{T}\underline{G}\underline{T}$$

The rows of the kernel matrix form a set of basis vectors in an N-dimensional vector space.

The rows are orthonormal



$$\underline{T}\,\underline{T}^{*t} = I \quad \text{or} \quad \sum_{i=0}^{N-1} T_{ji}\, T^{*}_{ki} = \delta_{jk}$$

A basis image is generated by inverse transforming a coefficient matrix containing only one non-zero element which is one,

i.e. $\underline{G}^{p,q} = \{\delta_{i-p,j-q}\}$ This is the coefficient matrix

only non-zero for $p=q$ which is 1

$$F_{mn} = \sum_{i=0}^{N-1} T(i,m) \sum_{k=0}^{N-1} \delta_{i-p,k-q}\, T(k,n) = T(p,m)\, T(q,n)$$

this is the outer product of two rows of T

Sinusoidal transforms

matrix kernel for the DFT

$$\underline{\underline{W}} = \begin{bmatrix} w_{00} & \cdots & w_{0,N-1} \\ \vdots & \ddots & \vdots \\ w_{N-1,0} & \cdots & w_{N-1,N-1} \end{bmatrix}$$

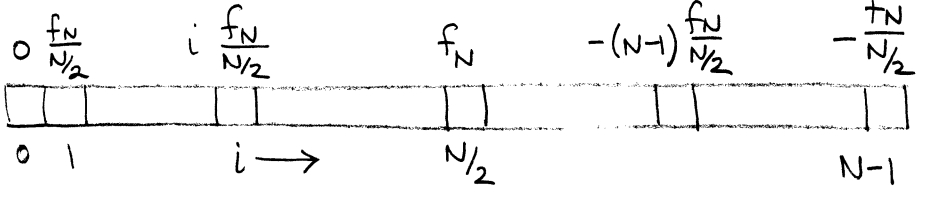
where $w_{ik} = \frac{1}{\sqrt{N}} e^{-j2\pi \frac{ik}{N}}$

$\underline{\underline{W}}$ is unitary

In one dimension

$$\underline{F} = \underline{\underline{W}}\,\underline{f} \quad \text{and} \quad \underline{f} = \underline{\underline{W}}^{*t}\,\underline{F}$$

spectrum vector



This is usually shifted

$$F(u) \Leftrightarrow f(x) \Rightarrow F(u-u_0) \Leftrightarrow e^{j2\pi x \frac{u_0}{N}} f(x)$$

$$= e^{j\pi x} f(x)$$

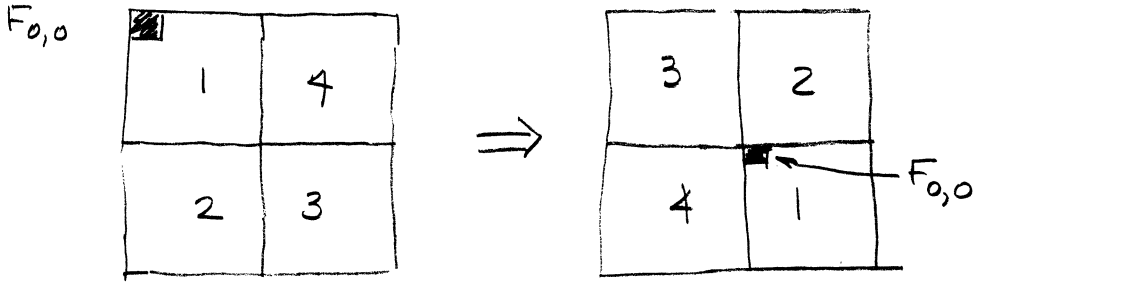
$$= (-1)^x f(x)$$

where $u_0 = \frac{N}{2}$

In two dimensions

$$\underline{G} = \underline{W}\,\underline{F}\,\underline{W} \quad \text{and} \quad \underline{F} = \underline{W}^{*t}\,\underline{G}\,\underline{W}^{*t}$$

$$F(u,v) \Leftrightarrow f(x,y) \Rightarrow F\left(u-\tfrac{N}{2}, v-\tfrac{N}{2}\right) \Leftrightarrow (-1)^{x+y} f(x,y)$$



transformation rearranges spectrum to be symmetric for plotting

discrete cosine transform

$$G_c(m,n) = \alpha(m)\alpha(n)\sum_{i=0}^{N-1}\sum_{k=0}^{N-1} g(i,k)\cos\left[\frac{\pi(2i+1)m}{2N}\right]\cos\left[\frac{\pi(2k+1)n}{2N}\right]$$

$$g(i,k) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} \alpha(m)\alpha(n)G_c(m,n)\cos\left[\frac{\pi(2i+1)m}{2N}\right]\cos\left[\frac{\pi(2k+1)n}{2N}\right]$$

where $\alpha(0) = \sqrt{\frac{1}{N}}$ and $\alpha(m) = \sqrt{\frac{2}{N}}$ for $1 \le m \le N$

In matrix form, the DCT can be written as a unitary matrix operation

$$\underline{G_c} = \underline{C}\,\underline{g}\,\underline{C}$$

where

$$C_{im} = \alpha(m)\cos\left[\frac{\pi(2i+1)m}{2N}\right]$$

Notes

- can be computed by a fast algorithm
- real valued
- widely used in image compression

discrete sine transform

- similar to DCT but $N = 2^P$
- used in certain image compression applications

$$G_s(m,n) = \frac{2}{N+1}\sum_{i=0}^{N-1}\sum_{k=0}^{N-1} g(i,k)\sin\left[\frac{\pi(i+1)(m+1)}{N+1}\right]\sin\left[\frac{\pi(k+1)(n+1)}{N+1}\right]$$

$$g(i,k) = \frac{2}{N+1}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} G_s(m,n)\sin\left[\frac{\pi(i+1)(m+1)}{N+1}\right]\sin\left[\frac{\pi(k+1)(n+1)}{N+1}\right]$$

$$T_{i,k} = \sqrt{\frac{2}{N+1}}\sin\left[\frac{\pi(i+1)(k+1)}{N+1}\right]$$

Discrete Hartley Transform (DHT)

use the basis function $cas(\theta) = \cos(\theta) + \sin(\theta) = \sqrt{2}\cos\left(\theta - \frac{\pi}{4}\right)$

Forward

$$G_{mn} = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{N-1} g_{ik}\, cas\left[\frac{2\pi(im+kn)}{N}\right]$$

$$g_{ik} = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} G_{mn}\, cas\left[\frac{2\pi(im+kn)}{N}\right]$$

identical transforms

$$T_{ik} = \frac{1}{\sqrt{N}}\, cas\left[\frac{2\pi ik}{N}\right]$$

DFT N numbers $\longrightarrow$ N complex symmetric numbers

DHT N numbers $\longrightarrow$ N real numbers

computational alternative to Fourier transform avoiding complex arithmetic

convolution theorem

$$g(x) = f(x) * h(x) \iff F(u)H_e(u) + F(-u)H_o(u) = G(u)$$

$F(u)$ – Hartley transform of f

$G(u)$ – Hartley transform of g

$H_e(u)$ – even components of Hartley transform of h

$H_o(u)$ – odd components of Hartley transform of h

Hadamard transform (also called the Walsh transform)

- symmetric
- seperable
- unitary
- only +1 and -1 as elements.
- exists for $N = 2^n$

$$\frac{1}{\sqrt{2}} \underline{H}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Can be successively generated as

$$\frac{1}{\sqrt{N}} \underline{H}_N = \frac{1}{\sqrt{N}} \begin{bmatrix} \underline{H}_{N/2} & \underline{H}_{N/2} \\ \underline{H}_{N/2} & -\underline{H}_{N/2} \end{bmatrix}$$

For example for $N = 8$

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

sequency (sign changes)

$$H_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{matrix} 0 \\ 7 \\ 3 \\ 4 \\ 1 \\ 6 \\ 2 \\ 5 \end{matrix}$$

ordered Hadamard transform

$$H_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix}$$

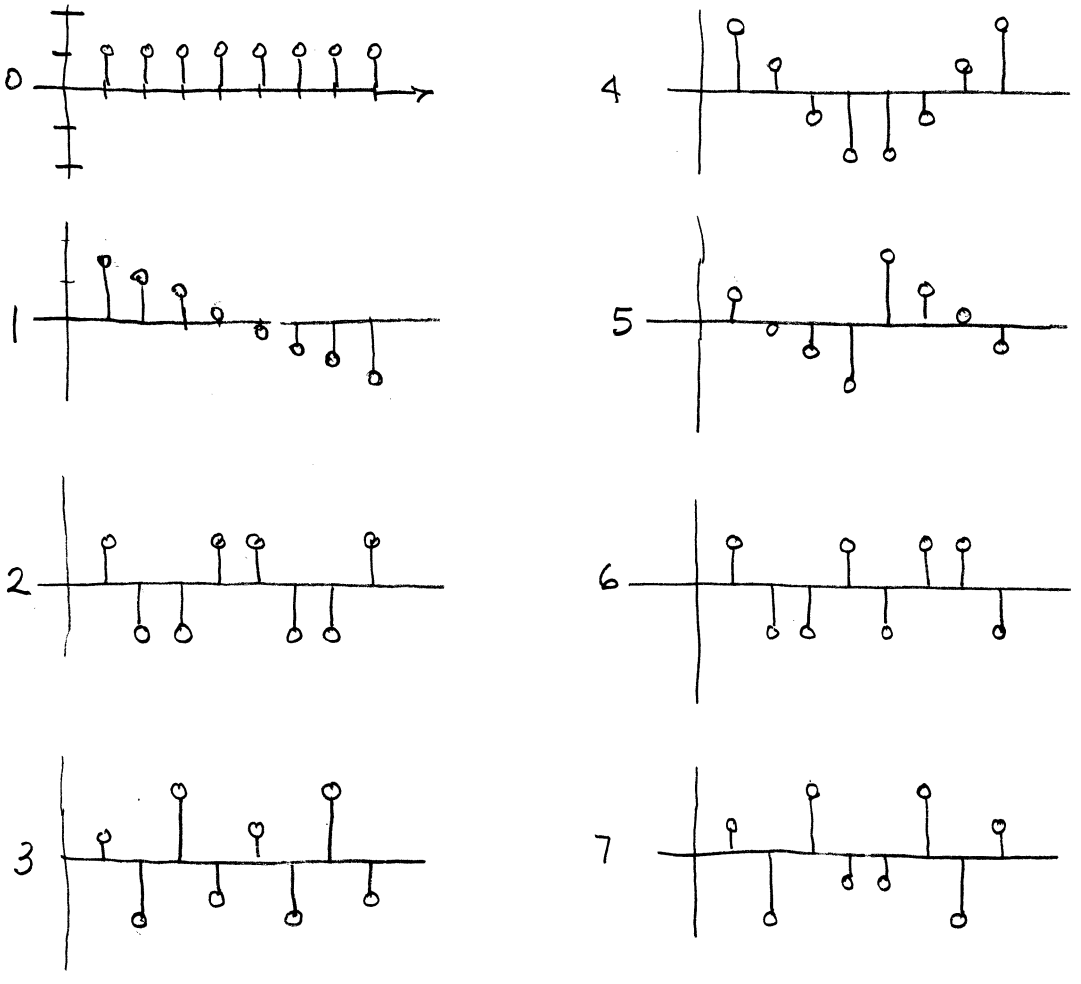
The slant transform — has a fast transform, used for image compression

$$S_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

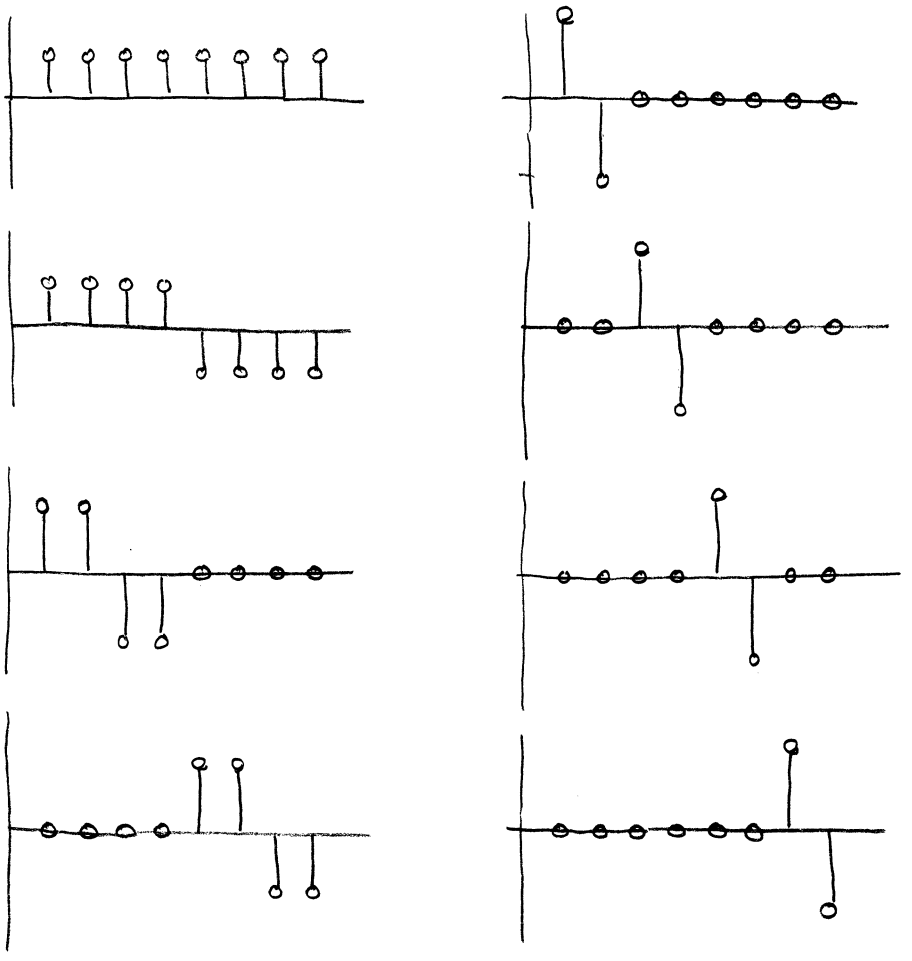
$$S_N = \frac{1}{\sqrt{2}} \left[\begin{array}{cc|c|cc|c} 1 & 0 & & 1 & 0 & \\ a_N & b_N & \underline{0} & -a_N & b_N & \underline{0} \\ \hline & \underline{0} & \underline{I} & & \underline{0} & \underline{I} \\ \hline 0 & 1 & & 0 & -1 & \\ -b_N & a_N & \underline{0} & b_N & a_N & \underline{0} \\ \hline & \underline{0} & \underline{I} & & \underline{0} & -\underline{I} \end{array}\right] \left[\begin{array}{c|c} \underline{S}_{N/2} & \underline{0} \\ \hline \underline{0} & \underline{S}_{N/2} \end{array}\right]$$

$$a_{2N} = \sqrt{\frac{3N^2}{4N^2-1}} \quad \text{and} \quad b_{2N} = \sqrt{\frac{N^2-1}{4N^2-1}}$$

called slant because it has linear basis functions



Haar
- symmetric
- separable
- unitary
- $N = 2^P$
- vary in both scale (width) and position; used for wavelets



N = 8 Haar transform basis functions
p specifies scale, q specifies width

Mathematically,

$$h_0(x) = \frac{1}{\sqrt{N}}$$

$$h_k(x) = \frac{1}{\sqrt{N}} \begin{cases} 2^{P/2} & \frac{q-1}{2^P} \le x \le \frac{q-\frac{1}{2}}{2^P} \\ -2^{P/2} & \frac{q-\frac{1}{2}}{2^P} \le x \le \frac{q}{2^P} \\ 0 & \text{otherwise} \end{cases}$$

where $k = 2^P + q - 1$

2^P is the largest power of 2 such that $2^P \le k$ and $q-1$ is the remainder.

$$\underline{Hr} = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}$$

Haar functions are scaled, shifted versions of an odd rectangular pulse.

Notes

- not as useful to plot in terms of k since p and q refer to scale and shift
- addresses lines and edges directly so it can be used to call attention to line and edge features

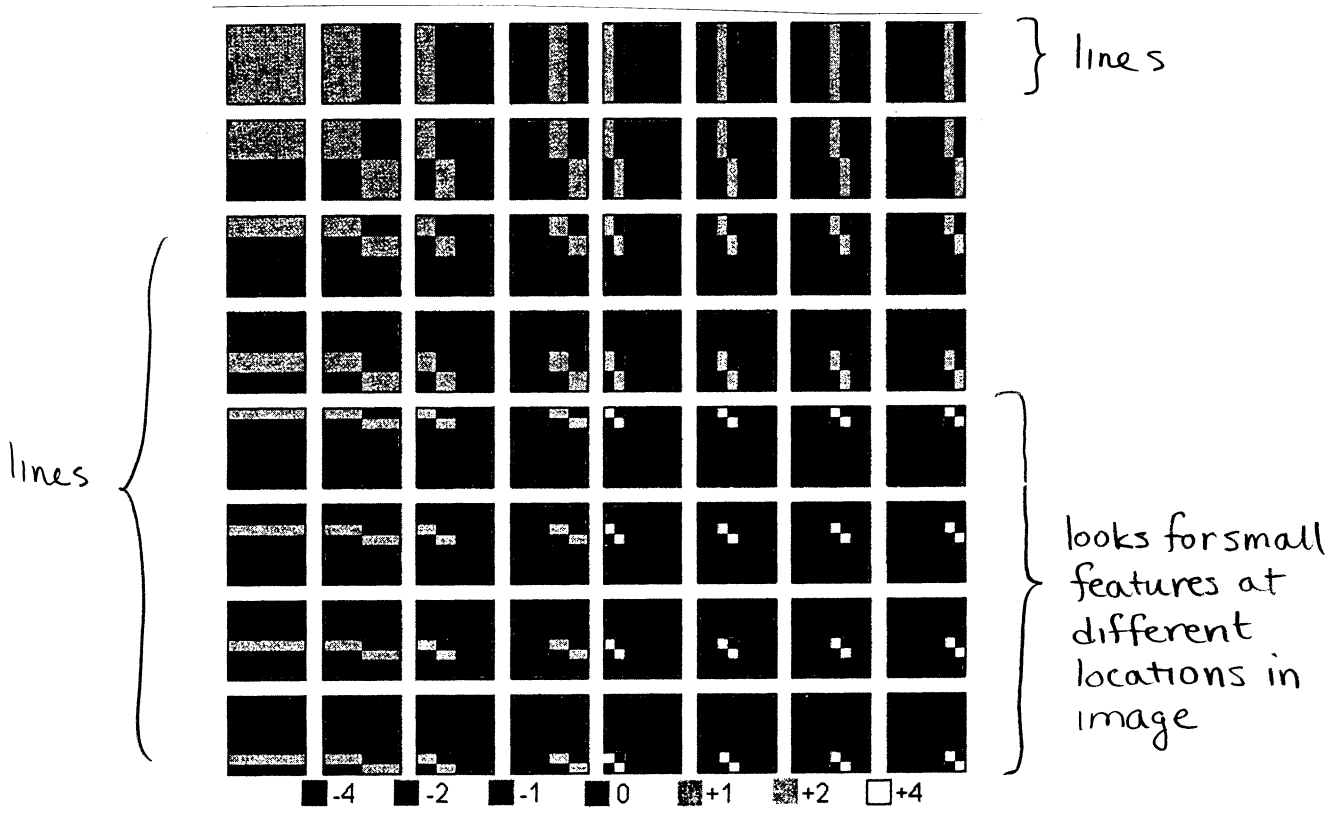


Figure 13–6 The Haar transform basis images for $N = 8$

Eigenvector based transforms

For an $N \times N$ matrix $\underline{A}$ there are N scalars λ_k, $k = 0, \ldots, N-1$

$$|\underline{A} - \lambda_k \underline{I}| = 0$$

These are called the eigenvalues. There are N corresponding eigenvectors $\underline{v}_k$

$$\underline{A}\,\underline{v}_k = \lambda_k \underline{v}_k$$

The eigenvectors form an orthonormal basis set.

Let $\underline{x}$ be a $N \times 1$ random vector, i.e., each x_i is a random variable

estimate mean $\quad \underline{m}_x \approx \frac{1}{L} \sum_{l=1}^{L} \underline{x}_l$

↑ set of L random vectors

estimate covariance matrix $\quad \underline{C}_x = E\left\{(\underline{x} - \underline{m}_x)(x - \underline{m}_x)^t\right\} \approx \frac{1}{L} \sum_{l=1}^{L} (\underline{x}_l \underline{x}_l^t - \underline{m}_x \underline{m}_x^t)$

$N \times N$, real, symmetric

define $\quad \underline{y} = \underline{A}(\underline{x} - \underline{m}_x)$

↑ rows are eigenvectors of $\underline{C}_x$

Compute $\underline{C}_y = \underline{A}\, C_x \underline{A}^t = \left[\lambda_k \underline{v}_k\right] \underline{A}^t = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{bmatrix}$

↑ but $\underline{v}_k$ are the rows (eigenvalues) of A

IMPORTANT RESULTS

(1) A removes correlation among the variables

(2) λ_k is the variance of the k-th transformed variable

Dimensional Reduction

Assume $\underline{m} = \underline{0}$

Drop the lower $N-M$ rows of $\underline{A}$ to give $\underline{B}$

$$\hat{y} = Bx$$

$\hat{y}$: estimate of variable y.

Invert to get

$$\hat{x} = B^t \hat{y}$$

where $\hat{x}$ is an approximation of x using only M components.

The error is only $\sum_{k=M+1}^{N} \lambda_k = MSE$

Karhunen-Loéve Transform.

$$y = A(x - m_x)$$

A: eigenvectors of C_x

By dropping $m-N$ components we can dramatically reduce the dimensionality of a set of vectors (images)

Some comments

- As correlation between adjacent pixels approach one, the K-L basis functions approach those of the DCT

Example: 24 band multispectral image, 1000 x 1000 pixels/band

Compute covariance matrix for 1,000,000 24-element random variables.

Transform, keeping only m largest eigenvalues.

If M is small this is a terrific savings in memory.

SVD (singular value decomposition)

Write $N \times N$ matrix $\underline{A}$ as

$$\underline{A} = \underline{U}\,\underline{\Lambda}\,\underline{V}^t \qquad (1)$$

columns $\underline{U}$ are eigenvectors of $\underline{A}\,\underline{A}^t$

columns $\underline{V}$ are eigenvectors of $\underline{A}^t\,\underline{A}$

} orthogonal.

$\underline{\Lambda}$ is a diagonal matrix with the singular values of $\underline{A}$ along the diagonal

$$\underline{\Lambda} = \underline{U}^t A\, \underline{V} \qquad (2)$$

Equations (1) and (2) define a transform pair.

- eigenvectors must be computed for each image
- has at most N non zero elements and, if some are zero, this gives data compression
- usually several of the singular values are small enough to ignore for even more data compression.

SVD example.

$$A = \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 4 & 3 & 1 \\ 2 & 4 & 5 & 4 & 2 \\ 1 & 3 & 4 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

$$\underline{A}\underline{A}^t = \begin{bmatrix} 6 & 14 & 18 & 14 & 6 \\ 14 & 36 & 48 & 36 & 14 \\ 18 & 48 & 65 & 48 & 18 \\ 14 & 36 & 48 & 36 & 14 \\ 6 & 14 & 18 & 14 & 6 \end{bmatrix}$$

Now compute eigenvectors and eigenvalues of this matrix.

$$\underline{U} = \begin{bmatrix} 0.186 & 0.638 & 0.241 & -0.695 & -0.695 \\ 0.476 & 0.058 & -0.52 & -0.133 & -0.128 \\ 0.691 & -0.422 & 0.587 & 0 & 0 \\ 0.476 & 0.058 & -0.52 & 0.133 & 0.128 \\ 0.186 & 0.638 & 0.241 & 0.695 & 0.695 \end{bmatrix} \quad \underline{\lambda} = \begin{bmatrix} 147.07 \\ 1.872 \\ 0.058 \\ 0 \\ 0 \end{bmatrix}$$

Forward transform

$$\underline{\Lambda} = \underline{U}^t\underline{A}\underline{U} = \begin{bmatrix} 12.585 & 0 & 0 & 0 & 0 \\ 0 & -1.142 & 0 & 0 & 0 \\ 0 & 0 & 0.557 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Inverse transform

$$\underline{A} = \underline{U}\underline{\Lambda}\underline{U}^t = \begin{bmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 4 & 3 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 3 & 4 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{bmatrix}$$

as expected.

Transform domain filtering

- modify weighting coefficients prior to reconstructing image via inverse transform
- if either (desirable) signal or (undesirable) noise components of the image resemble one or a few of the basis images of a particular transform, then that transform will be useful in separating the two.
- The Haar transform is a good candidate for detecting vertical and horizontal lines and edges since several of its basis images match those features.

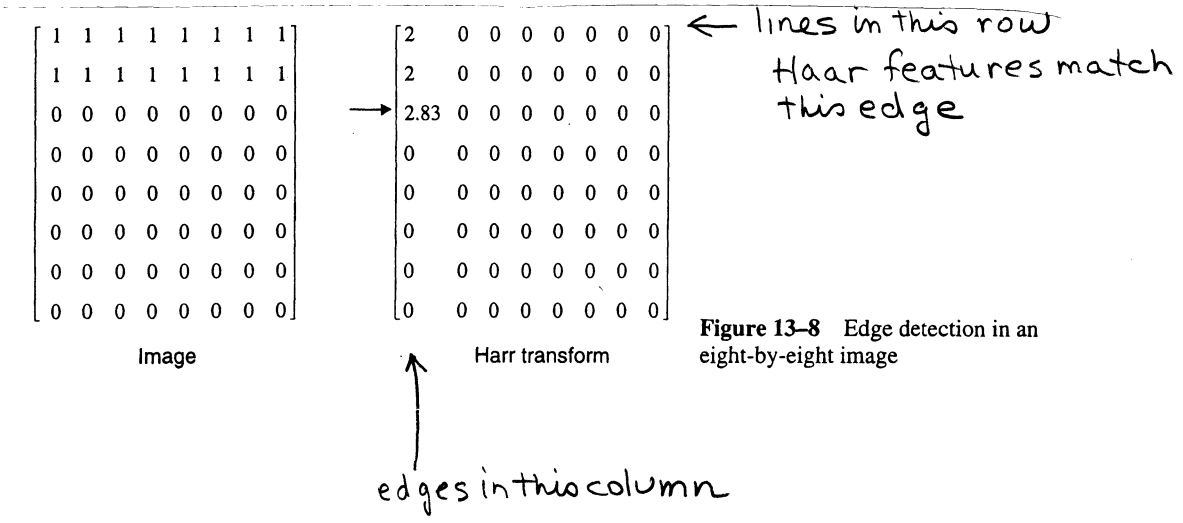


Figure 13–8 Edge detection in an eight-by-eight image

Examples of various transforms of a impulse at (0,0)

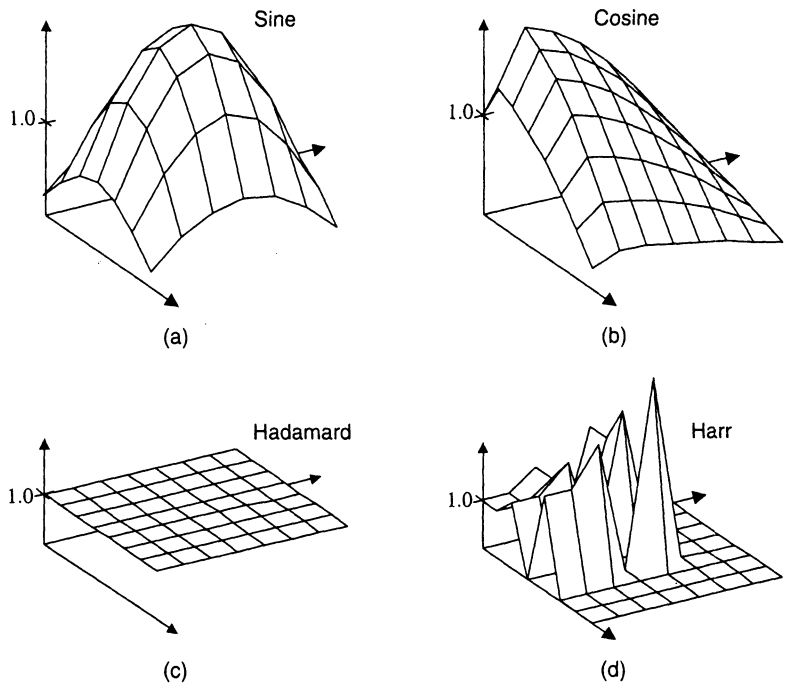


Figure 13–9 Transforms of an image containing an impulse: (a) DST; (b) DCT; (c) Hadamard: (d) Haar. The input is an eight-by-eight matrix, zero everywhere except the upper left element, which has value eight