CHAPTER 2

DIGITAL IMAGE TRANSFORM ALGORITHMS

Contents

Introduction

- Image transforms are represented by transform matrices A:

$$\mathbf{X} = \mathbf{A}\mathbf{x}$$

where **x** and **X** are the original and transformed image respectively.

In most cases the transform matrices are *unitary*:

$$\mathbf{A}^{-1} = \mathbf{A}^{*T}$$

- The columns of $\mathbf{A}^{*T}$ are the *basis vectors* of the transform and in 2-d transforms they correspond to *basis images*.
- The most popular image transforms are:
  - *Discrete Fourier Transform* (DFT).
  - *Discrete Cosine Transform* (DCT).

Two-dimensional discrete Fourier transform

Let $\tilde{x}(n_1, n_2)$ be a 2-d rectangularly periodic sequence. Its periodicity is defined as:

$$\tilde{x}(n_1, n_2) = \tilde{x}(n_1 + N_1, n_2) = \tilde{x}(n_1, n_2 + N_2)$$

where N_1, N_2 denote the horizontal and vertical periods.

The fundamental period of this sequence is the rectangle R_{N1N2} having size $N_1 \times N_2$:

$$R_{N_1N_2} = \{(n_1, n_2) : 0 \le n_1 < N_1, 0 \le n_2 < N_2\}$$

Two-dimensional discrete Fourier transform

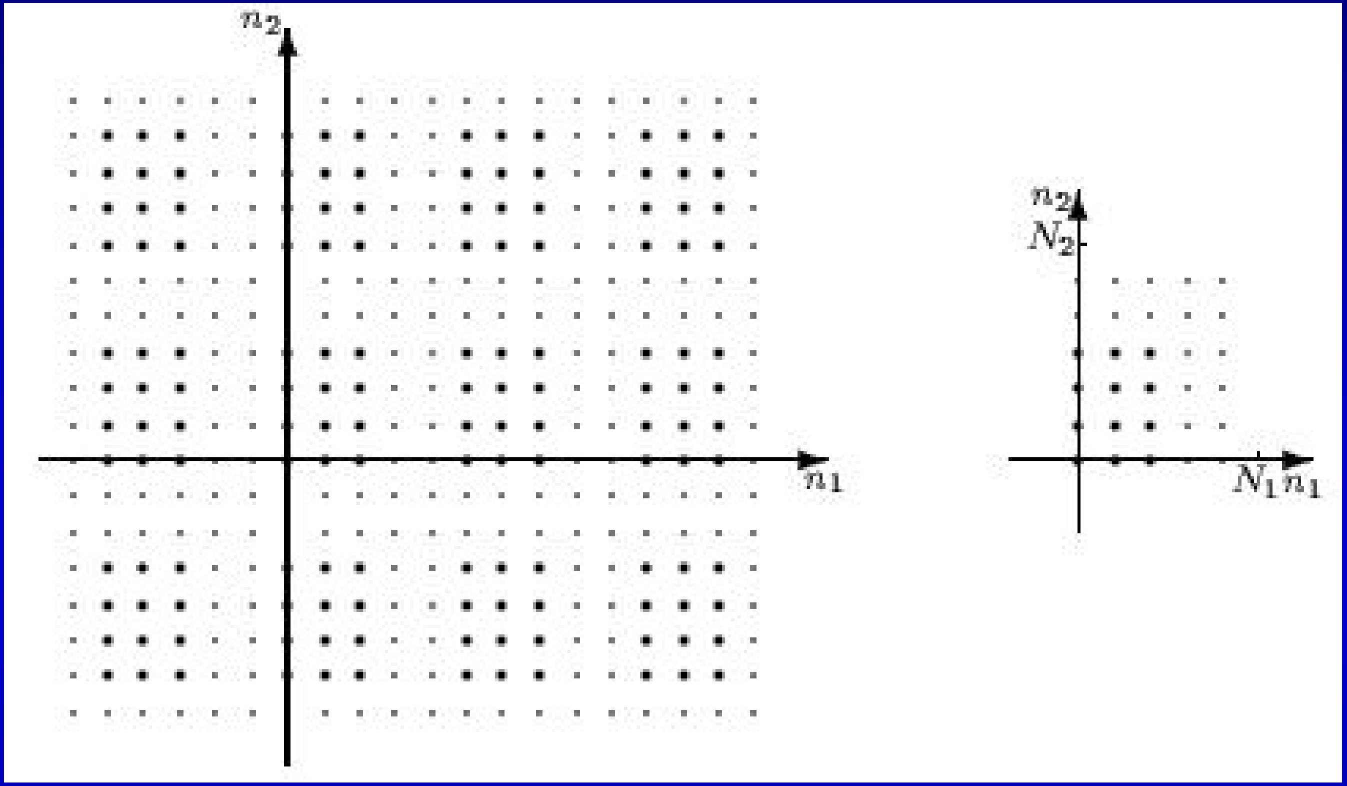


Figure 1: Rectangularly periodic sequence and its fundamental period.

Two-dimensional discrete Fourier transform

◆ A periodic sequence can be represented by a Fourier series.

◆ The 2-d DFS is used for the representation of a 2-d rectangularly periodic signal:

$$\tilde{x}(n_1,n_2)=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\tilde{X}(k_1,k_2)\exp\left(i\frac{2p\,n_1k_1}{N_1}+i\frac{2p\,n_2k_2}{N_2}\right)$$

◆ The Fourier series coefficients are :

$$\tilde{X}(k_1,k_2)=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\tilde{x}(n_1,n_2)\exp\left(-i\frac{2p\,n_1k_1}{N_1}-i\frac{2p\,n_2k_2}{N_2}\right)$$

Two-dimensional discrete Fourier transform

◆ A discrete 2-d space-limited signal can be represented by DFT as:

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) \exp\left(-i \frac{2p\, n_1 k_1}{N_1} - i \frac{2p\, n_2 k_2}{N_2} \right)$$

$$x(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) \exp\left(i \frac{2p\, n_1 k_1}{N_1} + i \frac{2p\, n_2 k_2}{N_2} \right)$$

Two-dimensional discrete Fourier transform

- Another commonly used form of the DFT is:

$$X(k_1,k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1,n_2) W_{N_1}^{n_1k_1} W_{N_2}^{n_2k_2}$$

$$x(n_1,n_2) = \frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} X(k_1,k_2) W_{N_1}^{-n_1k_1} W_{N_2}^{-n_2k_2}$$

where: $W_{N_j} = \exp\left(-i\frac{2p}{N_j}\right) \qquad j = 1, 2$

Two-dimensional discrete Fourier transform

◆ The DFT is related to the Fourier transform of a discrete 2-d signal which is defined as:

$$X(w_1,w_2)=\sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty}x(n_1,n_2)\exp(-iw_1n_1-iw_2n_2)$$

$$x(n_1,n_2)=\frac{1}{4p^2}\int_{-p}^{p}\int_{-p}^{p}X(w_1,w_2)\exp(iw_1n_1+iw_2n_2)dw_1dw_2$$

The DFT coefficients are a sampled version of the 2-d FT of a discrete sequence:

$$X(k_1,k_2)= \left. X(w_1,w_2)\right|_{w_1=\frac{2p\,k_1}{N_1},\,w_2=\frac{2p\,k_2}{N_2}}$$

over the unit bicircle z_1 = exp(iù$_1$), z_2 = exp(iù$_2$).

Two-dimensional discrete Fourier transform

◆ The *circular convolution* of two signals can be computed by means of the *circular shift* of a signal.

Circular Shift

$$y(n_1, n_2) = x(((n_1 + m_1))_{N_1}, ((n_2 + m_2))_{N_2})$$

$$((n))_N \overset{\Delta}{=} n \bmod N$$

Circular Convolution

$$y(n_1, n_2) \overset{\Delta}{=} x(n_1, n_2) \otimes\otimes h(n_1, n_2) =$$

$$= \sum_{m_1=0}^{N_1-1} \sum_{m_2=0}^{N_2-1} x(m_1, m_2) h(((n_1 - m_1))_{N_1}, ((n_2 - m_2))_{N_2})$$

Two-dimensional discrete Fourier transform

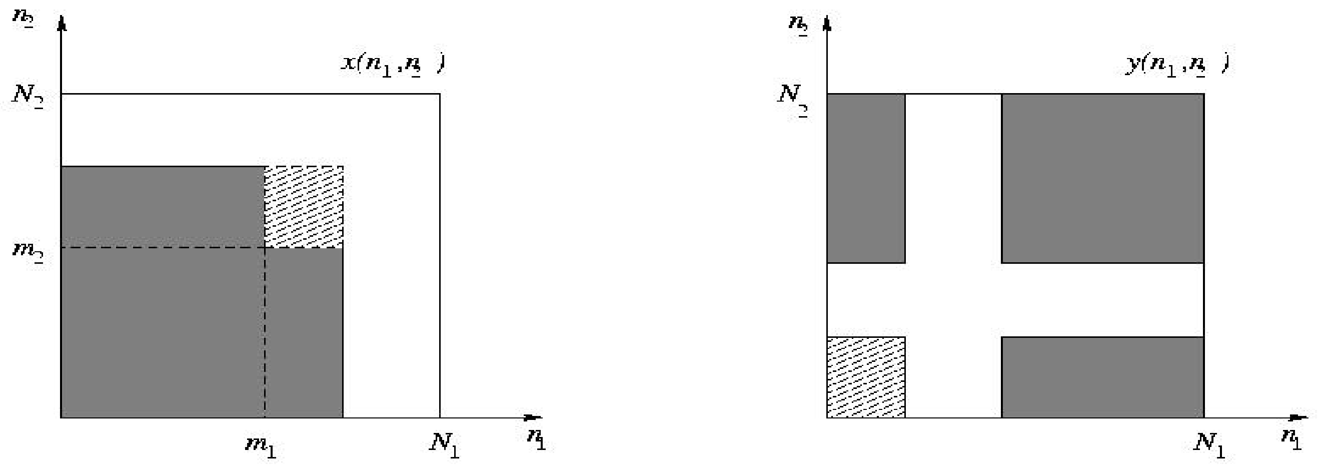


Figure 2: Circular shift of a 2-d sequence

Two-dimensional discrete Fourier transform

- In most applications the ***linear convolution*** of two signals is needed. It is defined as:

$$y(n_1,n_2)=\sum_{m_1=0}^{Q_1-1}\sum_{m_2=0}^{Q_2-1} h(m_1,m_2)x(n_1-m_1,n_2-m_2)$$

where R_{P1P2} = [0, P_1) x [0, P_2) and R_{Q1Q2} = [0, Q_1) x [0, Q_2) are the regions of support of x, h.

- The region of support of the convolution is:

$$R_{L_1L_2}=[0,L_1)\times[0,L_2) \quad L_i=P_i+Q_i-1 \quad i=1,2$$

Two-dimensional discrete Fourier transform

◆ The algorithm consists of the following steps:

1. Choose N_1, N_2 such that $N_i \geq P_i + Q_i - 1$, $i = 1, 2$.
2. Pad the sequences $x(n_1, n_2)$ and $h(n_1, n_2)$ with zeros, so that:

$$x_p(n_1, n_2) = \begin{cases} x(n_1, n_2) & (n_1, n_2) \in R_{P_1 P_2} \\ 0 & (n_1, n_2) \in R_{N_1 N_2} - R_{P_1 P_2} \end{cases}$$

$$h_p(n_1, n_2) = \begin{cases} h(n_1, n_2) & (n_1, n_2) \in R_{Q_1 Q_2} \\ 0 & (n_1, n_2) \in R_{N_1 N_2} - R_{Q_1 Q_2} \end{cases}$$

Two-dimensional discrete Fourier transform

3. Calculate the DFTs of the new sequences $x_p(n_1, n_2)$ and $h_p(n_1, n_2)$.

4. Calculate the DFT $Y_p(k_1, k_2)$, as the product of $X_p(k_1, k_2)$ and $H_p(k_1, k_2)$.

5. Calculate $y_p(n_1, n_2)$ by using the inverse DFT. The result of the linear convolution is:

$$y(n_1,n_2) = y_p(n_1,n_2) \qquad (n_1,n_2) \in R_{L_1 L_2}$$

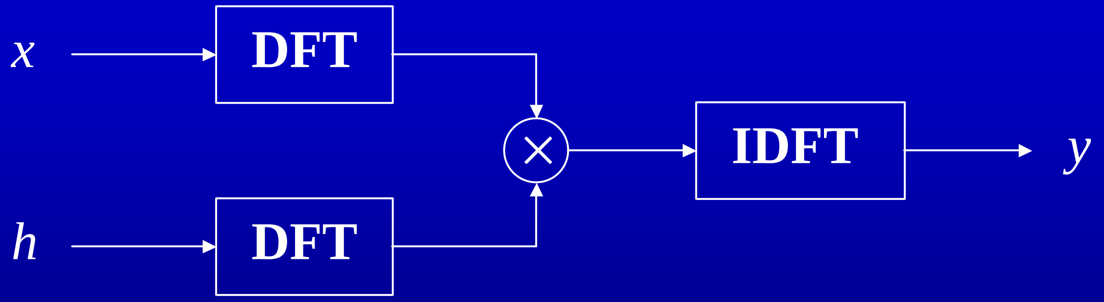


Figure 3: Convolution calculation using the DFTs

Two-dimensional discrete Fourier transform

- The DFT also supports the **2-d correlation**:

$$R_{xy}(m_1,m_2)=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)y(n_1+m_1,n_2+m_2)$$

- If both sequences are real then in the frequency domain:

$$P_{xy}(k_1,k_2)=X^*(k_1,k_2)Y(k_1,k_2)$$

- The **autocorrelation** of an image is defined as:

$$R_{xx}(m_1,m_2)=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)x(n_1+m_1,n_2+m_2)$$

Two-dimensional discrete Fourier transform

Properties of the 2-d DFT

1. Separable sequence:

$$x(n_1,n_2)=x_1(n_1)x_2(n_2)\leftrightarrow X(k_1,k_2)=X_1(k_1)X_2(k_2)$$

2. Linearity:

$$x(n_1,n_2)=a\cdot v(n_1,n_2)+b\cdot w(n_1,n_2)\leftrightarrow$$
$$X(k_1,k_2)=a\cdot V(k_1,k_2)+b\cdot W(k_1,k_2)$$

3. Circular shift:

$$y(n_1,n_2)=x(((n_1-m_1))_{N_1},((n_2-m_2))_{N_2})\leftrightarrow$$
$$Y(k_1,k_2)=W_{N_1}^{\;m_1k_1}W_{N_2}^{\;m_2k_2}X(k_1,k_2)$$

Two-dimensional discrete Fourier transform

4. Modulation:

$$y(n_1,n_2) = W_{N_1}^{-n_1 l_1} W_{N_2}^{-n_2 l_2} x(n_1,n_2) \leftrightarrow$$
$$Y(k_1,k_2) = X(((k_1 - l_1))_{N_1}, ((k_2 - l_2))_{N_2})$$

5. Complex conjugate property:

$$x^*(((N_1 - n_1))_{N_1}, ((N_2 - n_2))_{N_2}) \leftrightarrow X^*(k_1,k_2)$$

If $x(n_1, n_2)$ is a real-valued signal:

$$X^*(k_1,k_2) = X(((N_1 - k_1))_{N_1}, ((N_2 - k_2))_{N_2})$$

6. Reflection:

$$x(n_2,n_1) \leftrightarrow X(k_2,k_1)$$

Two-dimensional discrete Fourier transform

7. Initial value and DC value:

$$x(0,0)=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}X(k_1,k_2) \qquad X(0,0)=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)$$

8. Parseval's theorem:

$$\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)y^*(n_1,n_2)=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}X(k_1,k_2)Y^*(k_1,k_2)$$

$$\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\left|x(n_1,n_2)\right|^2=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\left|X(k_1,k_2)\right|^2$$

9. Circular convolution and multiplication:

$$x(n_1,n_2)\otimes\otimes h(n_1,n_2)\leftrightarrow X(k_1,k_2)\cdot H(k_1,k_2)$$

$$x(n_1,n_2)h(n_1,n_2)\leftrightarrow\frac{1}{N_1N_2}X(k_1,k_2)\otimes\otimes H(k_1,k_2)$$

Row-column FFT algorithm

◆ The simplest algorithm for the calculation of the 2-d DFT is the Row - Column FFT (RCFFT) algorithm.

◆ The 2-d DFT can be decomposed in N_1 DFTs along rows and N_2 DFTs along columns:

$$X'(n_1,k_2) = \sum_{n_2=0}^{N_2-1} X(n_1,n_2) W_{N_2}^{n_2 k_2}$$

$$X(k_1,k_2) = \sum_{n_1=0}^{N_1-1} X'(n_1,k_2) W_{N_1}^{n_1 k_1}$$

Row-column FFT algorithm

- The transform magnitude is:

$$X_M(k_1,k_2)=\sqrt{X_R(k_1,k_2)^2+X_I(k_1,k_2)^2}$$

- The transform phase is:

$$f(k_1,k_2)=\tan^{-1}\left(\frac{X_I(k_1,k_2)}{X_R(k_1,k_2)}\right)$$

- The magnitude of the transform provides useful information about the frequency content of an image.

Row-column FFT algorithm

◆ The number of complex multiplications for RCFFT is:

$$C = N_1 \frac{N_2}{2} \log_2 N_2 + N_2 \frac{N_1}{2} \log_2 N_1 = \frac{N_1 N_2}{2} \log_2 (N_1 N_2)$$

◆ If radix-2 FFT is used then the number of complex additions for RCFFT is:

$$A = N_1 N_2 \log_2 (N_1 N_2)$$

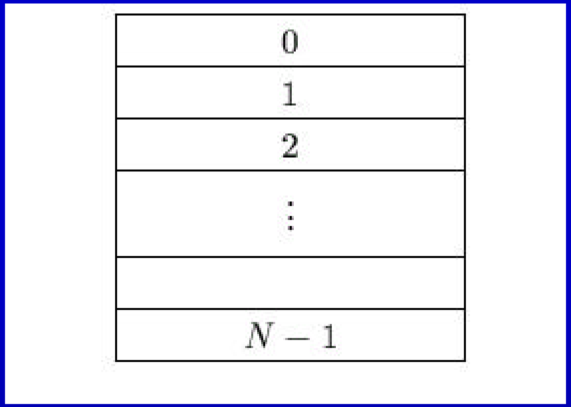
◆ The computational complexity is of the order :

$$O(kN^2 \log_2 N)$$

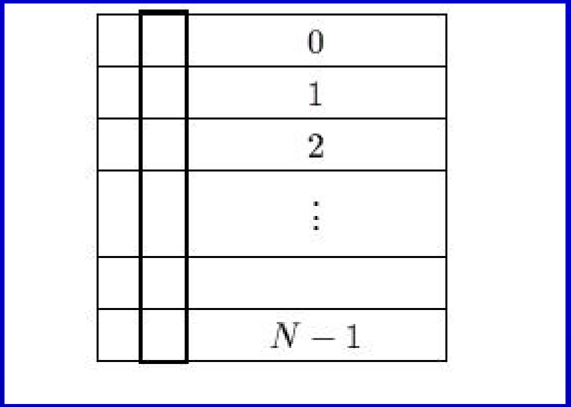
Memory problems in 2-d DFT calculations

- There are memory problems in the calculation of 2-d DFT even for moderately sized images.

- A solution to this problem is the storage of the image on a hard disk in a row-wise manner.



(a)



(b)

Figure 4: (a) Storage of an image in a row-wise manner. Each image row occupies one data record. b) Access of image in a column-wise manner.

Memory problems in 2-d DFT calculations

- The total number of I/O operations is:

$$N_{IO} = 2N + 3N + 2N(N-1) = 2N^2 + 3N$$

- If K signal rows (or columns) can be stored in RAM, the number of I/O operations required are:

$$N_{IO} = \frac{2N^2}{K} + 3N$$

Memory problems in 2-d DFT calculations

◆ A further speed-up is obtained by combining the row transform computation with a part of the column transform computation.

◆ The number of I/O operations is:

$$N_{IO} = 2N\lfloor n / k \rfloor$$

Memory problems in 2-d DFT calculations

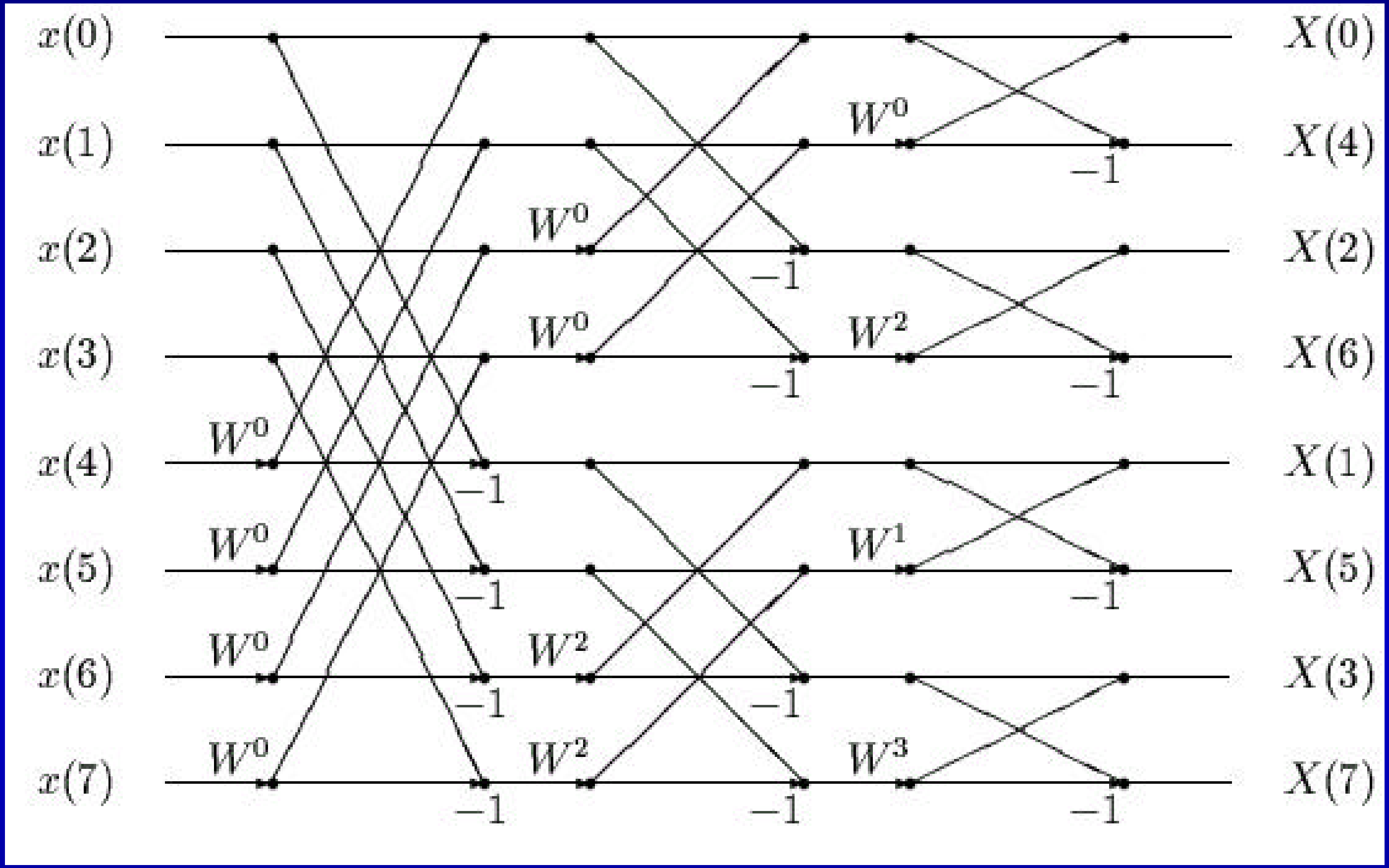


Figure 5: Radix-2 FFT algorithm for *N*=8.

Memory problems in 2-d DFT calculations

- ◆ Another approach for the reduction of the I/O operations is to use fast matrix transposition algorithms.

- ◆ If the matrix has size $N \times N$, where $N = 2^n$, it can be split into four submatrices of size $(N/2)\times(N/2)$ each.

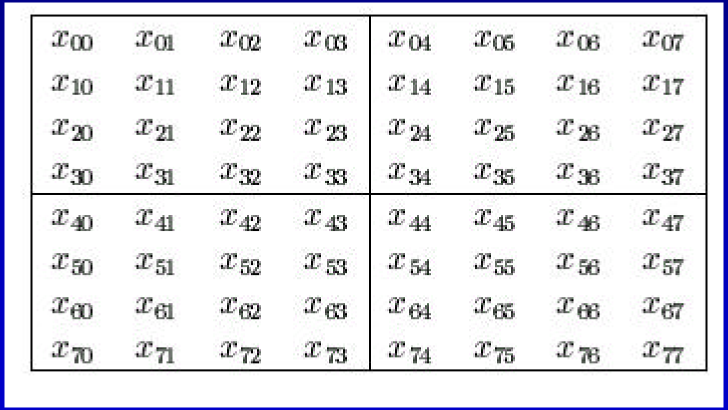
- ◆ This procedure is repeated until submatrices of size 2×2 are reached.

Memory problems in 2-d DFT calculations

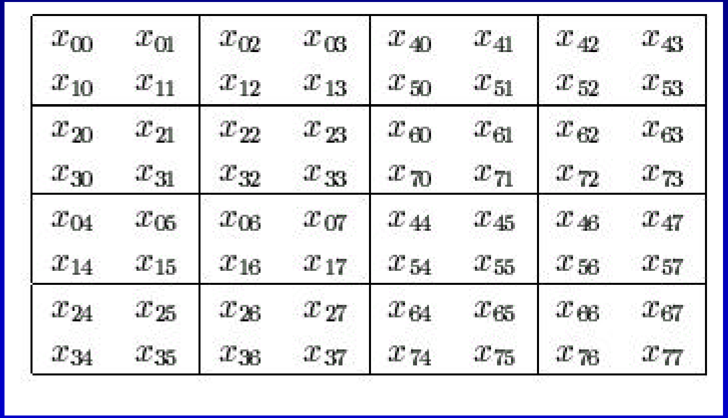
- This algorithm has $n = \log_2 N$ steps.
- The transposition can be performed in place.
- The first stage needs $2N$ I/O operations.
- The number of stages is $\log_2 N$.
- The total number of I/O operations is:

$$N_{IO} = 2N \log_2 N$$

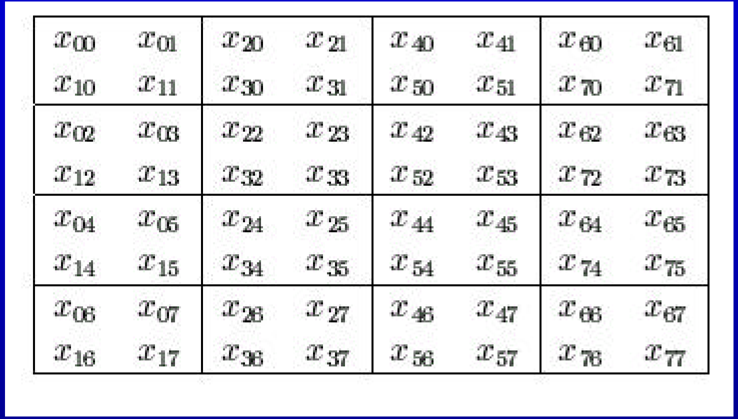
Memory problems in 2-d DFT calculations



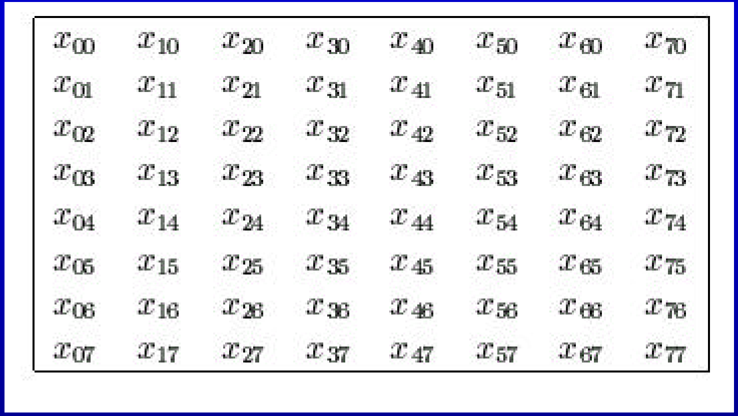
x_{00}	x_{01}	x_{02}	x_{03}	x_{04}	x_{05}	x_{06}	x_{07}
x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}
x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}
x_{30}	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}
x_{40}	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}
x_{50}	x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	x_{57}
x_{60}	x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}	x_{67}
x_{70}	x_{71}	x_{72}	x_{73}	x_{74}	x_{75}	x_{76}	x_{77}



x_{00}	x_{01}	x_{02}	x_{03}	x_{40}	x_{41}	x_{42}	x_{43}
x_{10}	x_{11}	x_{12}	x_{13}	x_{50}	x_{51}	x_{52}	x_{53}
x_{20}	x_{21}	x_{22}	x_{23}	x_{60}	x_{61}	x_{62}	x_{63}
x_{30}	x_{31}	x_{32}	x_{33}	x_{70}	x_{71}	x_{72}	x_{73}
x_{04}	x_{05}	x_{06}	x_{07}	x_{44}	x_{45}	x_{46}	x_{47}
x_{14}	x_{15}	x_{16}	x_{17}	x_{54}	x_{55}	x_{56}	x_{57}
x_{24}	x_{25}	x_{26}	x_{27}	x_{64}	x_{65}	x_{66}	x_{67}
x_{34}	x_{35}	x_{36}	x_{37}	x_{74}	x_{75}	x_{76}	x_{77}



x_{00}	x_{01}	x_{20}	x_{21}	x_{40}	x_{41}	x_{60}	x_{61}
x_{10}	x_{11}	x_{30}	x_{31}	x_{50}	x_{51}	x_{70}	x_{71}
x_{02}	x_{03}	x_{22}	x_{23}	x_{42}	x_{43}	x_{62}	x_{63}
x_{12}	x_{13}	x_{32}	x_{33}	x_{52}	x_{53}	x_{72}	x_{73}
x_{04}	x_{05}	x_{24}	x_{25}	x_{44}	x_{45}	x_{64}	x_{65}
x_{14}	x_{15}	x_{34}	x_{35}	x_{54}	x_{55}	x_{74}	x_{75}
x_{06}	x_{07}	x_{26}	x_{27}	x_{46}	x_{47}	x_{66}	x_{67}
x_{16}	x_{17}	x_{36}	x_{37}	x_{56}	x_{57}	x_{76}	x_{77}



x_{00}	x_{10}	x_{20}	x_{30}	x_{40}	x_{50}	x_{60}	x_{70}
x_{01}	x_{11}	x_{21}	x_{31}	x_{41}	x_{51}	x_{61}	x_{71}
x_{02}	x_{12}	x_{22}	x_{32}	x_{42}	x_{52}	x_{62}	x_{72}
x_{03}	x_{13}	x_{23}	x_{33}	x_{43}	x_{53}	x_{63}	x_{73}
x_{04}	x_{14}	x_{24}	x_{34}	x_{44}	x_{54}	x_{64}	x_{74}
x_{05}	x_{15}	x_{25}	x_{35}	x_{45}	x_{55}	x_{65}	x_{75}
x_{06}	x_{16}	x_{26}	x_{36}	x_{46}	x_{56}	x_{66}	x_{76}
x_{07}	x_{17}	x_{27}	x_{37}	x_{47}	x_{57}	x_{67}	x_{77}

Figure 6: Three stages in the transposition of an 8×8 matrix.

Memory problems in 2-d DFT calculations

- If the image is real, the complex conjugate property can be used for reducing memory requirements.

- Suppose that the dimensions N_1, N_2 of the image are powers of 2 and that the image is stored on a matrix **A** of size $N_1 \times N_2$.

- The first stage of the RCFFT processes the conjugate symmetry.

Memory problems in 2-d DFT calculations

- The columns $X'(n_1, 0)$, $X'(n_1, N_2/2)$ are real numbers.

- They are used to store the real and imaginary parts of $X'(n_1, k_2)$ in place as:

$$\begin{aligned} \mathrm{Re}\left[X'(n_1,k_2)\right] &\longrightarrow \mathbf{A}(n_1,k_2) \\ \mathrm{Im}\left[X'(n_1,k_2)\right] &\longrightarrow \mathbf{A}(n_1,N_2-k_2) \end{aligned} \qquad 1\le k_2 < N_2/2 \quad ,0\le n_1 \le N_1-1$$

- This storage requires only 50% of the memory required for the storage of the complete complex signal and is performed in place.

Memory problems in 2-d DFT calculations

◆ The column transform also satisfies the conjugate property.

◆ The samples $X(0, 0)$, $X(N_1 / 2, 0)$, $X(0, N_2 / 2)$, $X(N_1 / 2, N_2 / 2)$ are real and can be stored in the corresponding positions of **A**. The samples $X(k_1, 0)$, $X(k_1, N_2 / 2)$ can be stored as:

$$\mathrm{Re}[X(k_1,0)] \longrightarrow \mathbf{A}(k_1,0)$$

$$\mathrm{Im}[X(k_1,0)] \longrightarrow \mathbf{A}(N_1-k_1,0)$$

$$\mathrm{Re}[X(k_1,N_2/2)] \longrightarrow \mathbf{A}(k_1,N_2/2) \qquad 1 \le k_1 < N_1/2$$

$$\mathrm{Im}[X(k_1,N_2/2)] \longrightarrow \mathbf{A}(N_1-k_1,N_2/2)$$

Memory problems in 2-d DFT calculations

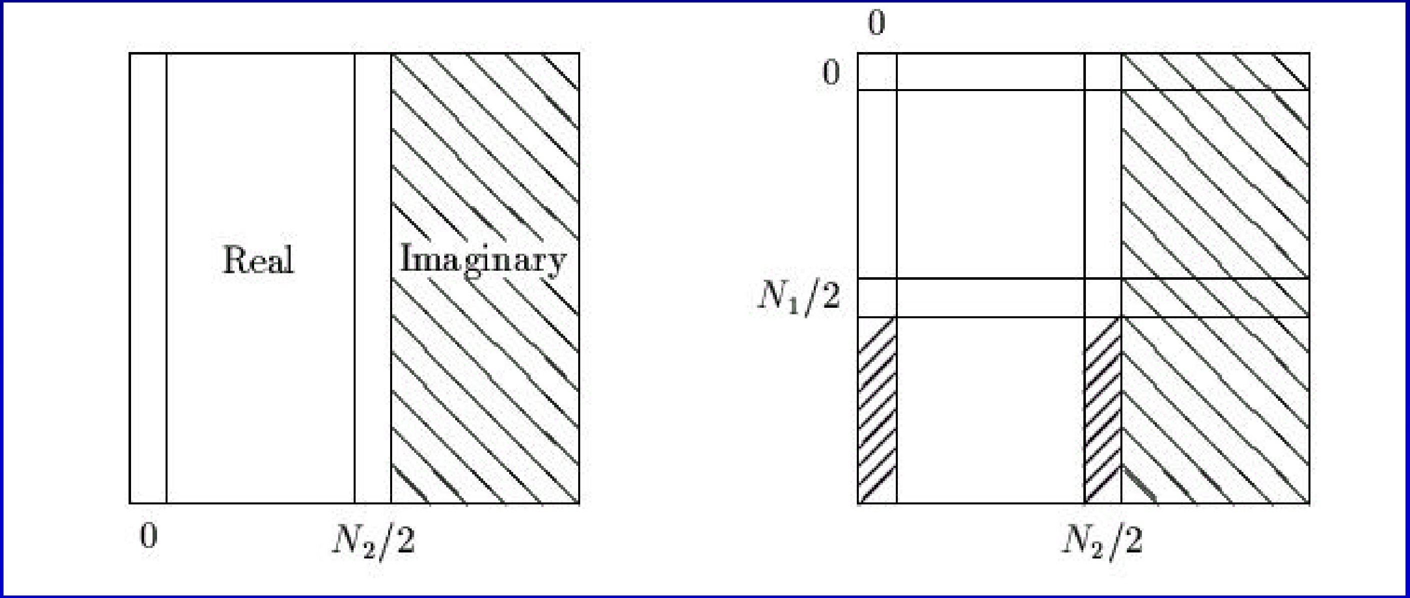
- The real and imaginary parts of the samples $X(k_1, k_2)$, $1 \le k_1 \le N_1$-1, $1 \le k_2 < N_2/2$ are stored as:

$$\mathrm{Re}\left[X(k_1,k_2)\right] \longrightarrow \mathbf{A}(k_1,k_2)$$

$$\mathrm{Im}\left[X(k_1,k_2)\right] \longrightarrow \mathbf{A}(((N_1-k_1))_{N_1},((N_2-k_2))_{N_2})$$

for $\quad 1 \le k_1 \le N_1 - 1, \quad 1 \le k_2 < N_2/2$

Memory problems in 2-d DFT calculations



(a) (b)

Figure 7: In-place storage of the 2-d DFT of a real-valued signal. (a) Storage after row transform. (b) Storage after column transform. The cross-hatched areas denote the storage places for the imaginary part of the transform.

Vector-radix fast Fourier transform algorithm

- Let $x(n_1, n_2)$ be a square image $N\times N$.

- Its DFT can be split into four 2-d DFTs of size $(N/2)\times(N/2)$, by following a *decimation-in-time* approach:

$$X(k_1,k_2)=\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1}x(n_1,n_2)W_N^{n_1k_1}W_N^{n_2k_2}$$

$$=G_{ee}(k_1,k_2)+G_{eo}(k_1,k_2)W_N^{k_2}+$$

$$G_{oe}(k_1,k_2)W_N^{k_1}+G_{oo}(k_1,k_2)W_N^{k_1+k_2}$$

Vector-radix fast Fourier transform algorithm

Where:

$$S_{00}(k_1,k_2)=\sum_{l_1=0}^{N/2-1}\sum_{l_2=0}^{N/2-1}x(2l_1,2l_2)W_N^{2l_1k_1}W_N^{2l_2k_2}$$

$$S_{01}(k_1,k_2)=\sum_{l_1=0}^{N/2-1}\sum_{l_2=0}^{N/2-1}x(2l_1,2l_2+1)W_N^{2l_1k_1}W_N^{2l_2k_2}$$

$$S_{10}(k_1,k_2)=\sum_{l_1=0}^{N/2-1}\sum_{l_2=0}^{N/2-1}x(2l_1+1,2l_2)W_N^{2l_1k_1}W_N^{2l_2k_2}$$

$$S_{11}(k_1,k_2)=\sum_{l_1=0}^{N/2-1}\sum_{l_2=0}^{N/2-1}x(2l_1+1,2l_2+1)W_N^{2l_1k_1}W_N^{2l_2k_2}$$

Vector-radix fast Fourier transform algorithm

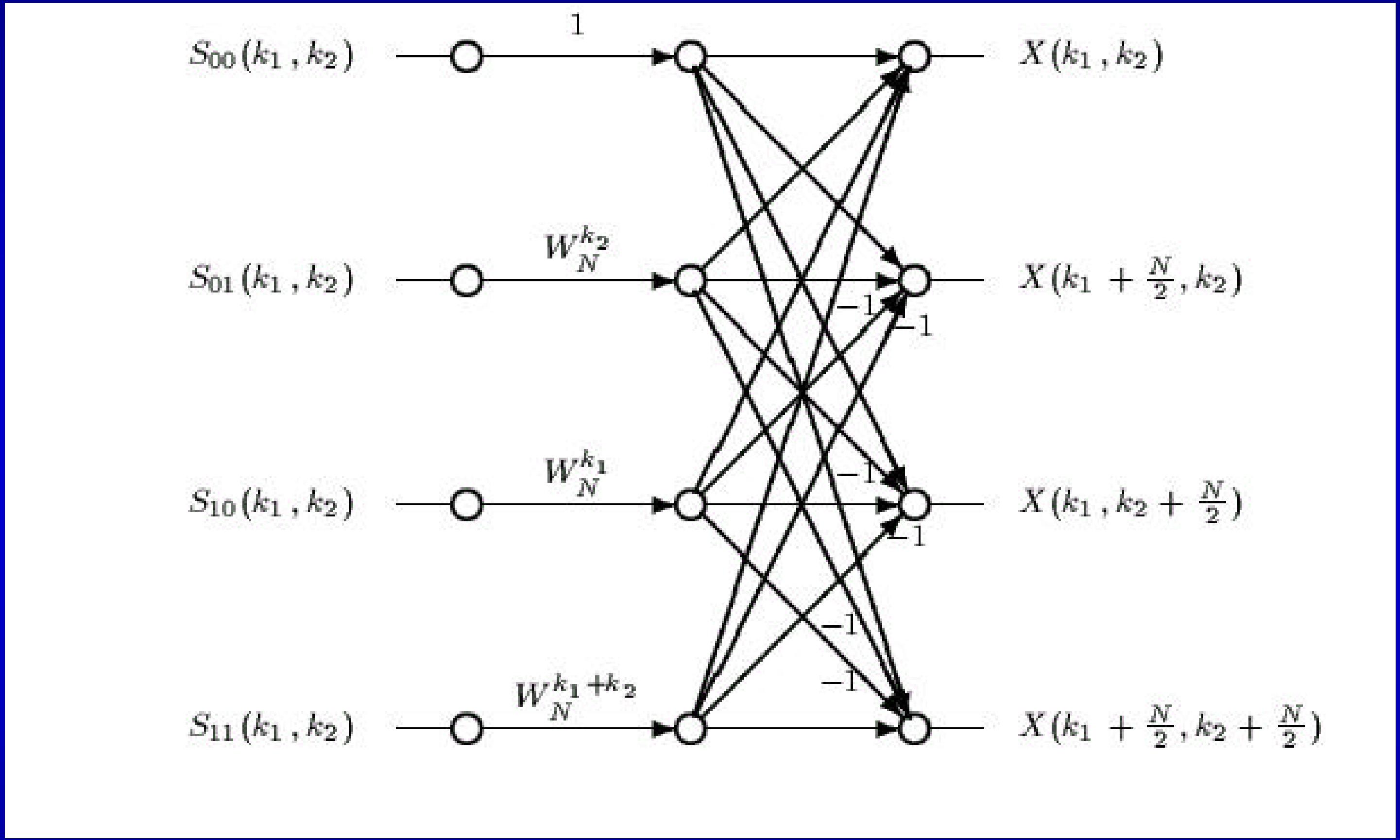


Figure 8: Radix-2×2 butterfly.

Vector-radix fast Fourier transform algorithm

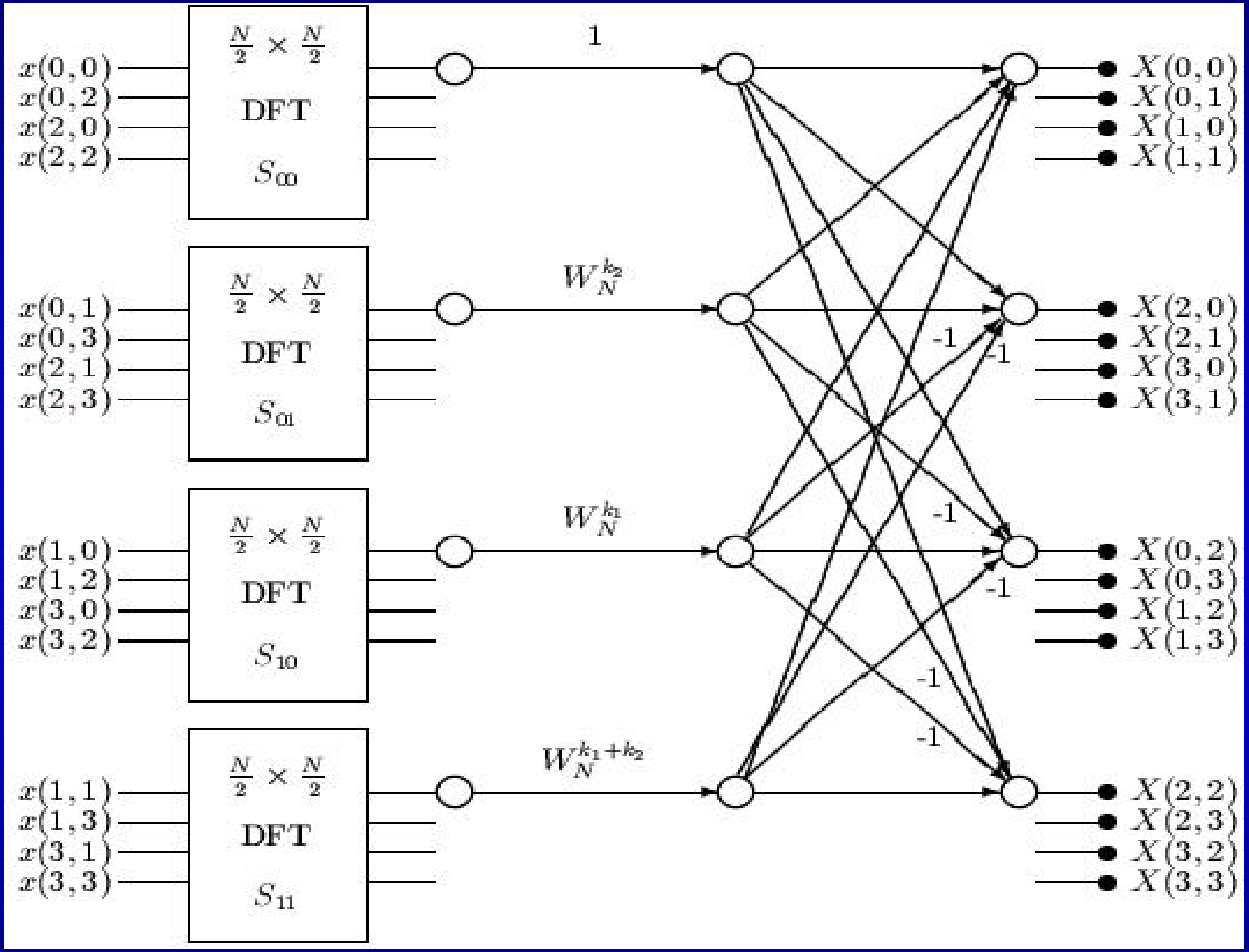


Figure 9: Flow diagram of a 4×4 vector-radix FFT.

Vector-radix fast Fourier transform algorithm

- VRFFT has $\log_2 N$ stages.
- Each stage contains $N^2/4$ butterflies.
- Each butterfly requires 3 complex multiplications.
- The total number of complex multiplications is:

$$C = \frac{3N^2}{4}\log_2 N$$

- The number of complex additions is:

$$A = 2N^2 \log_2 N$$

- All computations of the VRFFT can be performed in place.

Polynomial transform FFT

- A polynomial transform can be defined as:

$$\hat{X}_k(z) = \sum_{m=0}^{N-1} X_m(z)[G(z)]^{mk} \bmod P(z)$$

$$X_m(z) = \frac{1}{N}\sum_{k=0}^{N-1} \hat{X}_k(z)[G(z)]^{-mk} \bmod P(z)$$

where $P(z)$ is an irreducible polynomial and

$$G(z) = 1 \bmod P(z)$$

is a polynomial N-th root of unity. The polynomial of degree N-1 is:

$$X_m(z) = \sum_{n=0}^{N-1} x(m,n) z^n$$

Polynomial transform FFT

- The following polynomial transform is used for a 2-d FFT:

$$\hat{X}_k(z) = \sum_{m=0}^{N-1} X_m(z)(z^2)^{mk} \bmod (z^N + 1)$$

- Only $2N^2\log_2 N$ real additions are needed for the calculation of the polynomial transform.

- A polynomial transform can be employed for the fast calculation of the 2-d DFT.

Polynomial transform FFT

- **Polynomial transform FFT (PTFFT).**

$$X_{n_1}(z) = \sum_{n_2=0}^{N-1} x(n_1, n_2) W^{-n_2} z^{n_2}$$

$$\hat{X}_{(2k_2+1)k_1}(z) = \sum_{n_1=0}^{N-1} X_{n_1}(z) z^{2n_1k_1} \bmod (z^N + 1)$$

$$X((2k_2+1)k_1, k_2) = \sum_{l=0}^{N-1} y(k_1, l) W^{l} W^{2lk_2}$$

where $y(k_1,l)$ are the coefficients of the polynomial:

$$\hat{X}_{(2k_2+1)k_1}(z) = \sum_{l=0}^{N-1} y(k_1, l) z^{l}$$

Polynomial transform FFT

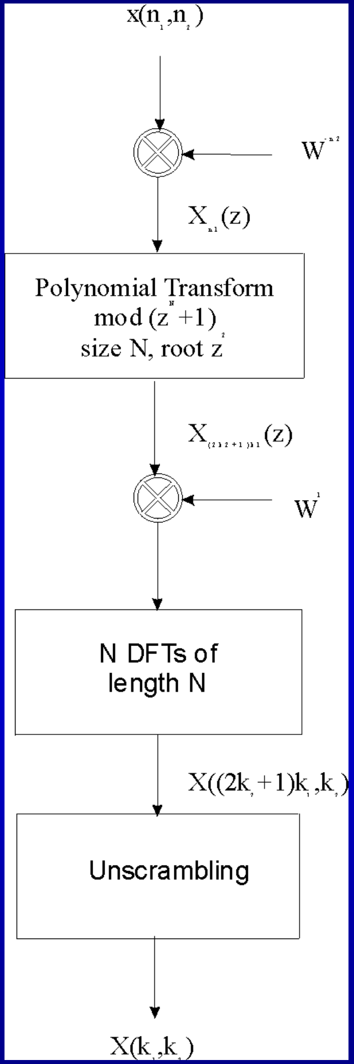


Figure 10: Polynomial transform FFT

Polynomial transform FFT

- If a radix-2 FFT is used for the 1-d DFTs then the total number of real multiplications is:

$$C = 2N^2(4 + \log_2 N)$$

- The total number of real additions is:

$$A = N^2(4 + 5\log_2 N)$$

- A comparison between RCFFT and PTFFT shows that PTFFT is better than RCFFT for N>16.

Two dimensional power spectrum estimation

◆ The power spectrum carries important information about the 2-d signal content.

◆ The squared magnitude $\hat{P}_{xx}(k_1, k_2)$ is the ***periodogram*** of the discrete signal and can be used as an estimator of its 2-d power spectrum.

$$\hat{P}_{xx}(k_1, k_2) = \frac{1}{N_1 N_2} \left| X(k_1, k_2) \right|^2$$

$$= \frac{1}{N_1 N_2} \left| \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2} \right|^2$$

Two dimensional power spectrum estimation

- The periodogram $\hat{P}_{xx}(w_1, w_2)$ is the Fourier transform of the estimator $\hat{R}_{xx}(n_1, n_2)$ of the autocorrelation function:

$$\hat{R}_{xx}(n_1,n_2) = \frac{1}{(2N_1+1)(2N_2+1)} \sum_{k_1=-N_1}^{N_1} \sum_{k_2=-N_2}^{N_2} x^*(k_1,k_2)x(k_1+n_1,k_2+n_2)$$

- The periodogram is a smoothed version of the actual 2-d power spectrum. It is very poor for small images and it is a noisy power spectrum estimator.

Two dimensional power spectrum estimation

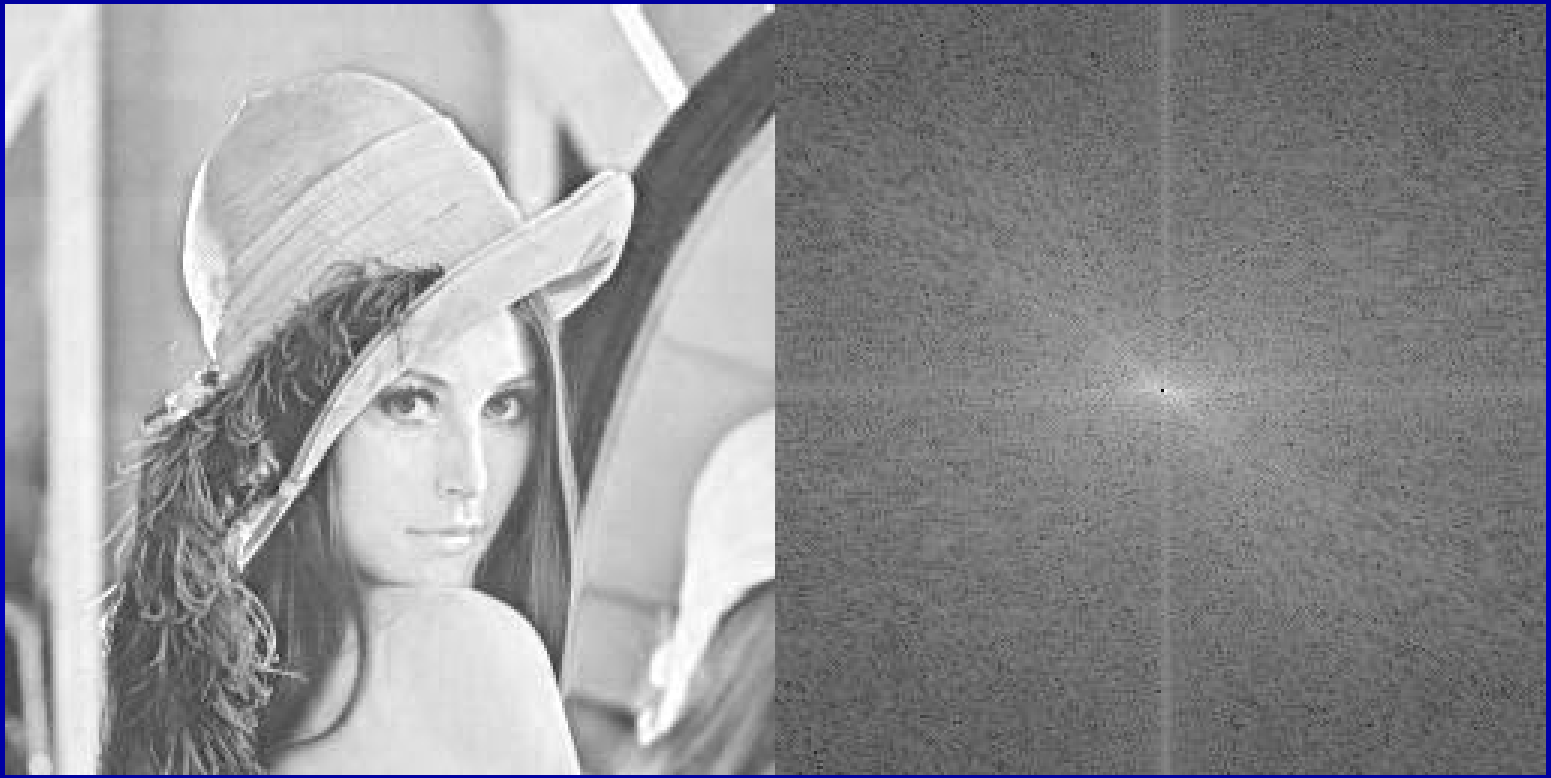


Figure 11: (a) Test image LENNA; (b) periodogram of LENNA.

Two dimensional power spectrum estimation

- The ***Bartlett*** estimator can reduce the noise. The 2-d signal of size $N_1 \times N_2$ is split into $K_1 \times K_2$ non-overlapping sections each having size $M_1 \times M_2$:

$$x_{ij}(n_1, n_2) = x(n_1 + iM_1, n_2 + jM_2)$$

where $i = 0, \ldots, K_1 - 1, \; j = 0, \ldots, K_2 - 1$

- The periodogram of each section is:

$$\hat{P}_{xx}^{(ij)}(k_1, k_2) = \frac{1}{M_1 M_2} \left| \sum_{n_1=0}^{M_1-1} \sum_{n_2=0}^{M_2-1} x_{ij}(n_1, n_2) W_{M_1}^{n_1 k_1} W_{M_2}^{n_2 k_2} \right|^2$$

Two dimensional power spectrum estimation

- The new spectral estimator is the average of the periodograms of all sections:

$$\hat{P}_{xx}^{B}(K_1,K_2)=\frac{1}{K_1K_2}\sum_{i=0}^{K_1-1}\sum_{j=0}^{K_2-1}\hat{P}_{xx}^{(ij)}(k_1,k_2)$$

- *Blackman-Tukey* 2-d power spectrum estimator: An estimator of the 2-d autocorrelation function $R_{xx}(m_1,m_2)$ is used:

$$P_{xx}^{BT}(k_1,k_2)=\sum_{m_1=0}^{2N_1-2}\sum_{m_2=0}^{2N_2-2}R_{xx}(m_1,m_2)w(m_1,m_2)W_{2N_1-1}^{m_1k_1}W_{2N_2-1}^{m_2k_2}$$

Two dimensional power spectrum estimation

◆ A separable 2-d window can be used. The triangular 1-d window or the 1-d Hamming window can be used for each of the dimensions.

◆ The BT estimator can be obtained by using the 2-d DFT of the windowed autocorrelation function:

$$P_{xx}^{BT}(k_1,k_2) = DFT\left[w(m_1,m_2)R_{xx}(m_1,m_2)\right]$$

Two dimensional power spectrum estimation

- Technique based on 2-d AR modeling of images.

$$x(n_1,n_2) = \sum_{(i,j)\in A}\sum a(i,j)x(n_1-i,n_2-j)+w(n_1,n_2)$$

A is the *prediction window*

á(*i*,*j*) are the *predictor coefficients*

$w(n_1,n_2)$ is a white noise random process, having variance $ó_w^2$.

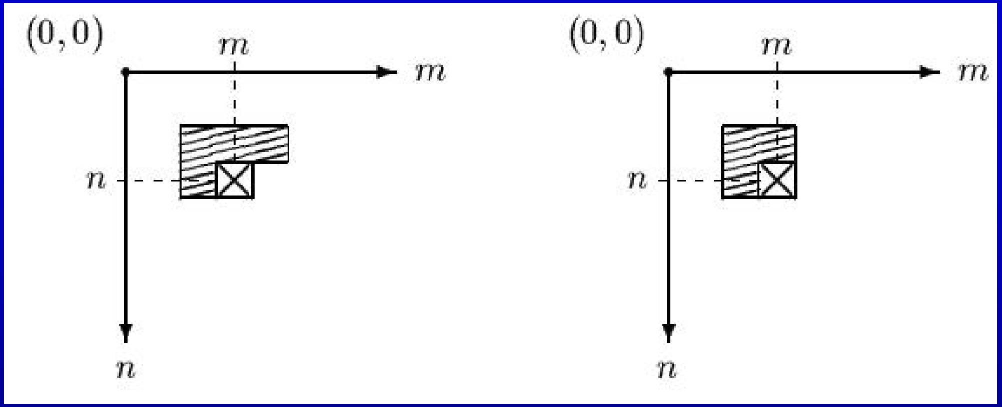


Figure 12: Examples of prediction windows.

Two dimensional power spectrum estimation

- The power spectrum can be estimated by the following formula:

$$P_{xx}^{AR}(k_1,k_2)=\frac{s_w^2}{\left|\sum_{(m_1,m_2)}\sum_{\in A}a(m_1,m_2)W_{N_1}^{m_1k_1}W_{N_2}^{m_2k_2}\right|^2}$$

Discrete cosine transform

- DCT is used in the JPEG and MPEG standards.

- Forward DCT transform:

$$C(0) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n)$$

$$C(k) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos \frac{(2n+1)kp}{2N}$$

- Inverse DCT:

$$x(n) = \frac{1}{\sqrt{N}} C(0) + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} C(k) \cos \frac{(2n+1)kp}{2N}$$

Discrete cosine transform

- Let *x*(*n*) be a signal. An *even sequence* *f*(*n*) can be produced by this signal as:

$$f(n) = \begin{cases} x(n) & 0 \le n \le N-1 \\ x(2N-1-n) & N \le n \le 2N-1 \end{cases}$$

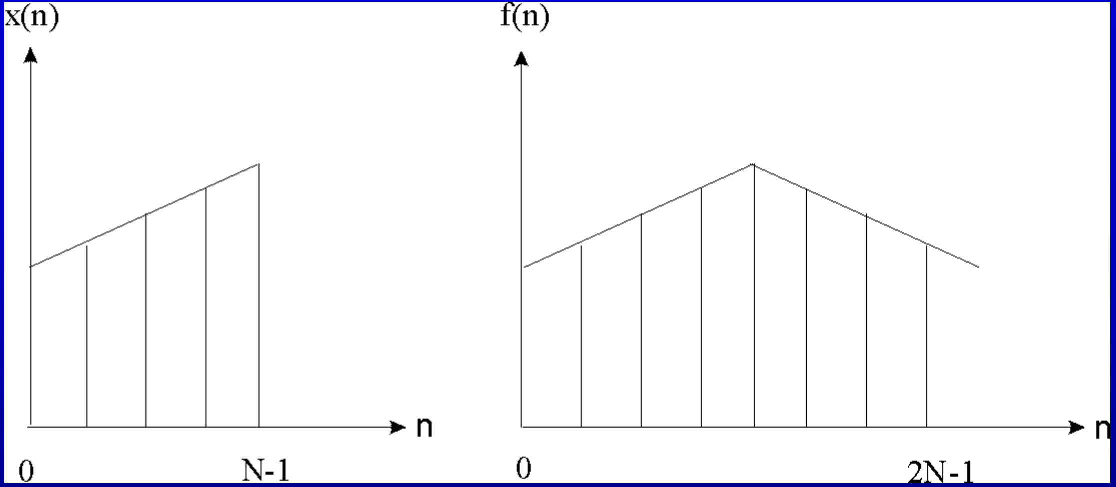


Figure 13: Creation of an even sequence

Discrete cosine transform

◆ Algorithm for the fast calculation of DCT using DFT:

1. Formation of the even sequence $f(n)$.
2. The DFT of length $2N$ is calculated by using a 1-d FFT of length $2N$.
3. Calculation of the DCT coefficients $C(k)$ using the relations:

$$C(k) = \frac{W_{2N}^{k/2}}{\sqrt{2N}} F(k) \quad for\ 0 \le k \le N-1$$

where $F(k)$ are the DFT coefficients.

Discrete cosine transform

- Alternative method for the fast calculation of the 1-d DCT:

$$C(k)=\sqrt{\frac{2}{N}}\sum_{n=0}^{N-1}x(n)\,\mathrm{Re}\left\{\exp\left(-i\,\frac{(2n+1)kp}{2N}\right)\right\}$$

- If $x(n)$ is a real sequence then:

$$C(k)=\sqrt{\frac{2}{N}}\,\mathrm{Re}\left\{\exp\left(-i\,\frac{kp}{2N}\right)\sum_{n=0}^{2N-1}y(n)\exp\left(-i\,\frac{nkp}{N}\right)\right\}$$

where: $$y(n)=\begin{cases}x(n) & 0\le n\le N-1\\ 0 & N\le n\le 2N-1\end{cases}$$

Discrete cosine transform

- Alternative definition for the DCT pair:

$$C(k)=\sum_{n=0}^{N-1} 2x(n)\cos\frac{(2n+1)kp}{2N}$$

$$x(n)=\frac{1}{N}\sum_{k=0}^{N-1} w(k)C(k)\cos\frac{(2n+1)kp}{2N}$$

where: $$w(k)=\begin{cases}1/2 & k=0\\ 1 & 1\le k\le N-1\end{cases}$$

- Thus, the DCT coefficients are related to the DFT coefficients by:

$$C(k)=W_{2N}^{k/2}F(k)$$

Discrete cosine transform

- Fast DCT algorithm:

1. Formation of a sequence *í*(*n*) as:

$$v(n) = \begin{cases} x(2n) & 0 \le n \le \left[\frac{N-1}{2}\right] \\ x(2N-2n-1) & \left[\frac{N+1}{2}\right] \le n \le N-1 \end{cases}$$

2. Computation of the DFT $V(k)$, $0 \le k \le N$-1 by using an FFT algorithm of length N.

3. Computation of $C(k)$, $0 \le k \le [N/2]$, and $C(N\text{-}k)$ as:

$$C(k) = 2\,\mathrm{Re}\left\{W_{4N}^{k} \sum_{n=0}^{N-1} v(n) W_N^{nk}\right\} = 2\,\mathrm{Re}\left\{W_{4N}^{k} V(k)\right\}$$

$$C(N-k) = -2\,\mathrm{Im}\left\{W_{4N}^{k} V(k)\right\}$$

Discrete cosine transform

- Fast calculation of the inverse DCT:

1. Computation of $V(k)$ as:

$$V(k)=\frac{1}{2}W_{4N}^{-k}\left[C(k)-iC(N-k)\right] \qquad 0\le k\le N-1$$

2. Computation of $í(n)$ from $V(k)$ by means of an inverse FFT algorithm of length N.
3. Retrieval of $x(n)$ from $í(n)$ by using the formula:

$$v(n)=\begin{cases} x(2n) & 0\le n\le\left[\frac{N-1}{2}\right] \\ x(2N-2n-1) & \left[\frac{N+1}{2}\right]\le n\le N-1 \end{cases}$$

Two-dimensional discrete cosine transform

- A 2-d $N_1 \times N_2$ DCT is defined as:

$$C(k_1,k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} 4x(n_1,n_2)\cos\frac{(2n_1+1)k_1 p}{2N_1}\cos\frac{(2n_2+1)k_2 p}{2N_2}$$

for $0 \le k_1 \le N_1\text{-}1,\ 0 \le k_2 \le N_2\text{-}1.$

$$x(n_1,n_2) = \frac{1}{N_1 N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} w_1(k_1)w_2(k_2)C(k_1,k_2)\cos\frac{(2n_1+1)k_1 p}{2N_1}\cos\frac{(2n_2+1)k_2 p}{2N_2}$$

where:

$$w_1(k_1) = \begin{cases} 1/2 & k_1 = 0 \\ 1 & 1 \le k_1 \le N_1 - 1 \end{cases} \qquad w_2(k_2) = \begin{cases} 1/2 & k_2 = 0 \\ 1 & 1 \le k_2 \le N_2 - 1 \end{cases}$$

Two-dimensional discrete cosine transform

- The expansion of the signal forms a new one as:

$$f(n_1,n_2)=\begin{cases} x(n_1,n_2) & 0\le n_1\le N_1-1 & 0\le n_2\le N_2-1 \\ x(2N_1-n_1-1,n_2) & N_1\le n_1\le 2N_1-1 & 0\le n_2\le N_2-1 \\ x(n_1,2N_2-n_2-1) & 0\le n_1\le N_1-1 & N_2\le n_2\le 2N_2-1 \\ x(2N_1-n_1-1,2N_2-n_2-1) & N_1\le n_1\le 2N_1-1 & N_2\le n_2\le 2N_2-1 \end{cases}$$

- The DCT coefficients C(k_1, k_2) are related to the DFT ones F(k_1, k_2) as:

$$C(k_1,k_2)=W_{2N_1}^{k_1/2}W_{2N_2}^{k_2/2}F(k_1,k_2)$$

Two-dimensional discrete cosine transform

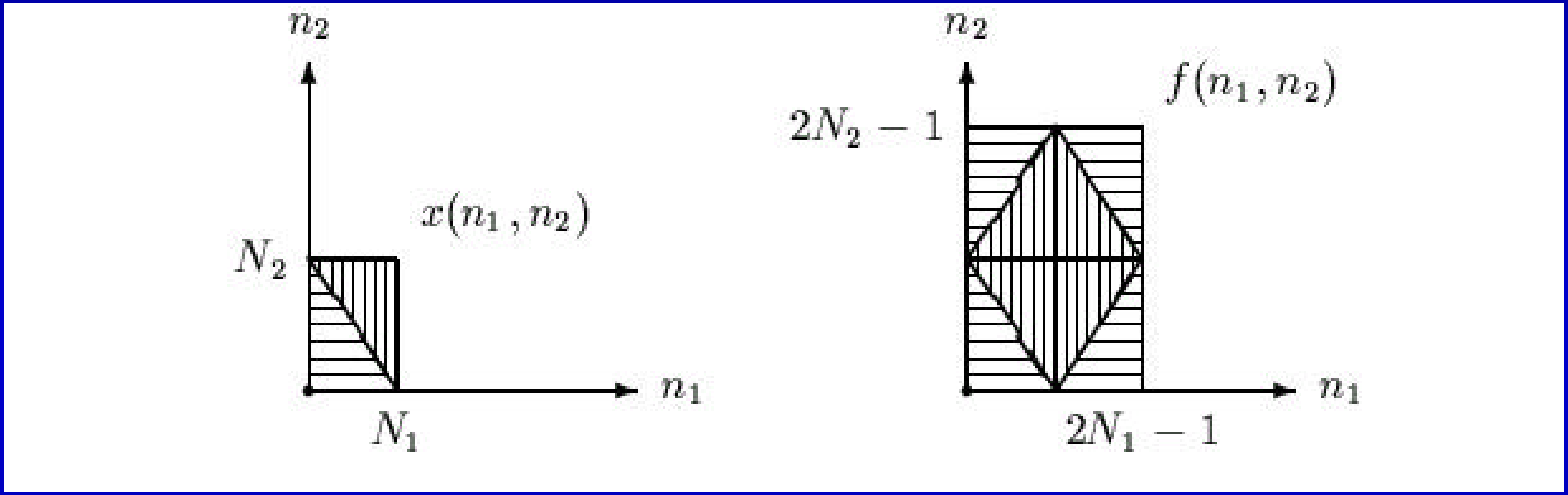


Figure 14: (a) Original sequence $x(n_1,n_2)$.
(b) Expanded sequence $f(n_1,n_2)$.

Two-dimensional discrete cosine transform

- Fast calculation of the 2-d DCT follows:

1. Formation of a sequence $í(n_1,n_2)$ as:

$$v(n_1,n_2)=\begin{cases} x(2n_1,2n_2) \\ x(2N_1-2n_1-1,2n_2) \\ x(2n_1,2N_2-2n_2-1) \\ x(2N_1-2n_1-1,2N_2-2n_2-1) \end{cases}$$

for:

$$\begin{array}{ll} 0\le n_1\le\left[\frac{N_1-1}{2}\right] & 0\le n_2\le\left[\frac{N_2-1}{2}\right] \\ \left[\frac{N_1+1}{2}\right]\le n_1\le N_1-1 & 0\le n_2\le\left[\frac{N_2-1}{2}\right] \\ 0\le n_1\le\left[\frac{N_1-1}{2}\right] & \left[\frac{N_2+1}{2}\right]\le n_2\le N_2-1 \\ \left[\frac{N_1+1}{2}\right]\le n_1\le N_1-1 & \left[\frac{N_2+1}{2}\right]\le n_2\le N_2-1 \end{array}$$

Two-dimensional discrete cosine transform

2. The DFT $V(k_1,k_2)$ is calculated by using a $N_1 \times N_2$ FFT algorithm.

3. The DCT coefficients are calculated as:

$$\begin{aligned} C(k_1,k_2) &= 2\,\mathrm{Re}\left\{W_{4N_1}^{k_1}\left[W_{4N_2}^{k_2}V(k_1,k_2)+W_{4N_2}^{-k_2}V(k_1,N_2-k_2)\right]\right\} \\ &= 2\,\mathrm{Re}\left\{W_{4N_2}^{k_2}\left[W_{4N_1}^{k_1}V(k_1,k_2)+W_{4N_1}^{-k_1}V(N_1-k_1,k_2)\right]\right\} \end{aligned}$$

Two-dimensional discrete cosine transform

◆ There is also an algorithm for the fast calculation of the inverse 2-d DCT.

1. Computation of $V(k_1,k_2)$ as:

$$V(k_1,k_2)=\frac{1}{4}W_{4N_1}^{-k_1}W_{4N_2}^{-k_2}\{[C(k_1,k_2)-C(N_1-k_1,N_2-k_2)]$$
$$-i[C(N_1-k_1,k_2)+C(k_1,N_2-k_2)]\}$$

2. Calculation of $\acute{\imath}(n_1,n_2)$ by using an inverse $N_1\times N_2$ 2-d FFT algorithm.

Two-dimensional discrete cosine transform

3. Retrieval of $x(n_1,n_2)$ from $í(n_1,n_2)$ by using the following formula:

$$v(n_1,n_2)=\begin{cases} x(2n_1,2n_2) \\ x(2N_1-2n_1-1,2n_2) \\ x(2n_1,2N_2-2n_2-1) \\ x(2N_1-2n_1-1,2N_2-2n_2-1) \end{cases}$$

for:

$$0 \le n_1 \le \left\lfloor \frac{N_1-1}{2} \right\rfloor \qquad 0 \le n_2 \le \left\lfloor \frac{N_2-1}{2} \right\rfloor$$

$$\left\lceil \frac{N_1+1}{2} \right\rceil \le n_1 \le N_1-1 \qquad 0 \le n_2 \le \left\lfloor \frac{N_2-1}{2} \right\rfloor$$

$$0 \le n_1 \le \left\lfloor \frac{N_1-1}{2} \right\rfloor \qquad \left\lceil \frac{N_2+1}{2} \right\rceil \le n_2 \le N_2-1$$

$$\left\lceil \frac{N_1+1}{2} \right\rceil \le n_1 \le N_1-1 \qquad \left\lceil \frac{N_2+1}{2} \right\rceil \le n_2 \le N_2-1$$