Computing DFT

Feb 15, 06

$N \times N$ image.

Direct $\longrightarrow$ N^4 operations.

Row/col $\longrightarrow$ $N^2 \log_2 N$ operation

$N = 1000$

$$\left.\begin{array}{ll} \text{Direct} & \sim 10^{12} \text{ op.} \\ \text{Row/col} & \approx 10^{6} \times 10 \approx 10^{7} \end{array}\right\} \Longrightarrow \frac{10^{12}}{10^{7}} \approx \underline{10^{5}}$$

~~$\frac{10^{5}}{60}$ sec $\frac{10^{3}}{60}$ min~~ = 30 hours $\longrightarrow$ $1\frac{1}{4}$ days.

1 sec $\longrightarrow$

- Row/Col decomposition requires a matrix Transpose?

How do we do matrix Transpose in an I/O efficient way?

Eklundh's Alg for Transposition

Recursive alg:

$N \times N \longrightarrow \log_2 N$ stages.

- Fast non I/O intensive way of doing matrix Transpose.
- Divide + Conquer..

Observation

$$\underset{N \times N}{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$$A^T = \begin{bmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{bmatrix}$$

simple example $A = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$ $A^T = \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{bmatrix}$

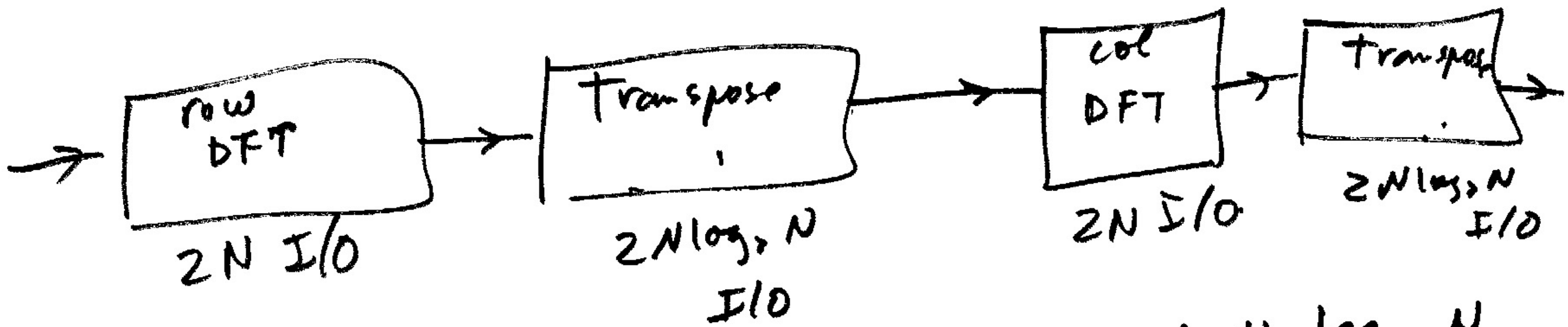
# of stages of decomposition $\log_2 N$ stages.

Each stage $2N$ I/O operations.

Example: first stage. : $\left.\begin{matrix} 4 \text{ I/O per row} \\ N/2 \text{ rows} \end{matrix}\right\} \rightarrow 4 \times \frac{N}{2} = 2N$ I/O per stage

Total I/O cont : $2N \log_2 N$ stage.

Excludes Alg for matrix Transpo.



total I/O: $4N + 4N \log_2 N \approx 4N \log_2 N$.

See Fig 3.20 + 3.21 of J. Lim.

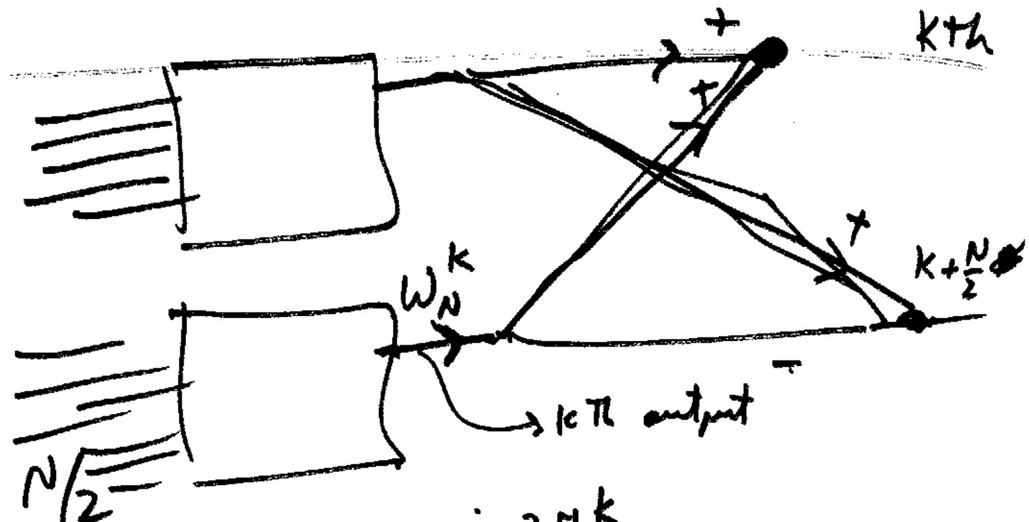
FFT

- Direct
- Row/col.
- Vector Radix FFT. (True Translation of Divide + Conquer in 2D).

Recall 1-D.

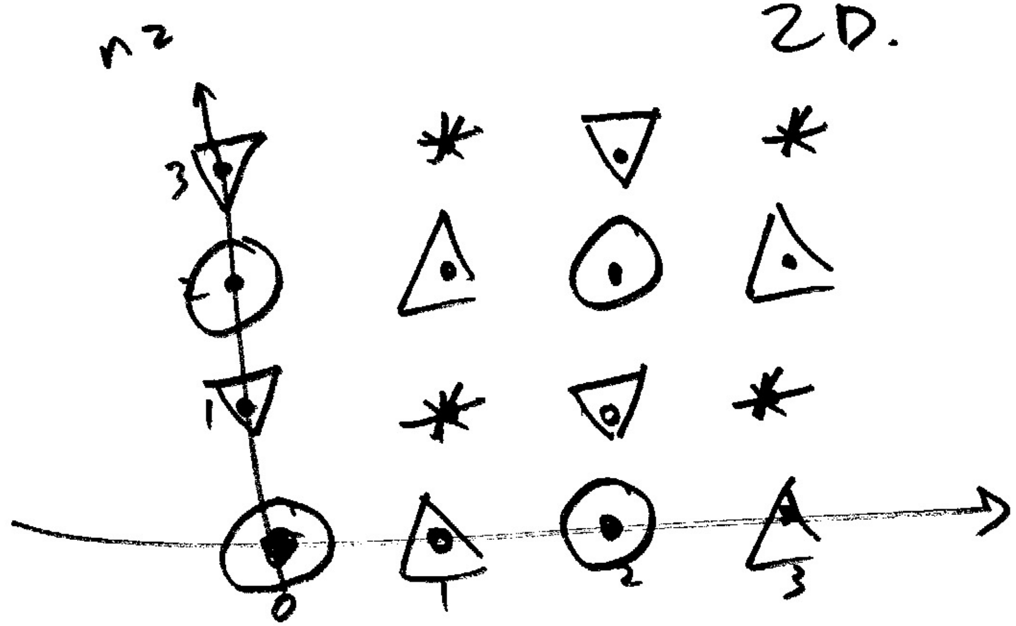
$x(n)$ Npt. → even point $X_e(n)$ $N/2$.

→ odd point $X_o(n)$



$$W_N^k \triangleq e^{-j\frac{2\pi k}{N}}$$

2D.



$x(n_1, n_2)$

e = even
o = odd.

Divide $x_{4\times 4}$ into 4 subsequences:

$$\left\{ \begin{array}{l} g_{ee}(n_1, n_2) \\ g_{eo}(n_1, n_2) \\ g_{oe}(n_1, n_2) \\ g_{oo}(n_1, n_2) \end{array} \right.$$

g_{ee} = even index in n_1 and in n_2

g_{eo} = even in n_1 odd in n_2

g_{oe} = odd in n_1 even in n_2.

g_{oo} $*$ = odd in n_1 odd in n_2.

Q how is DFT of x related to DFT of geo, gee goe goo ?

Show Fig 3.23, 3.24, 3.25 J.S. Lim

$$\underset{N\times N}{X(k_1,k_2)} = G_{ee}(k_1,k_2) + G_{eo}(k_1,k_2)\,e^{-j\frac{2\pi k_2}{N}} + G_{oe}(k_1,k_2)\,e^{-j\frac{2\pi k_1}{N}} + G_{oo}(k_1,k_2)\,e^{-j\frac{2\pi k_1}{N}}\,e^{-j\frac{2\pi k_2}{N}}$$

G_{ee}	:	$N/2 \times N/2$ PT DFT of g_{ee}
G_{oe}		" " " " " g_{oe}
G_{eo}		" " " " " g_{eo}
G_{oo}		" " " " " g_{oo}

$N \times N \longrightarrow 4 \quad \frac{N}{2} \times N/2.$

$N/2 \times N/2 \longrightarrow 4 \quad N/4 \times N/4$

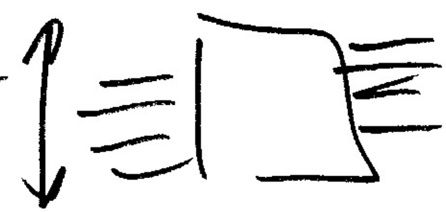
$N/4 \times N/4 \longrightarrow 4 \quad N/8 \times N/8$

$\vdots$

# of stage $N \times N$ image. $\log_2 N$

Each stage, how many butterflies: each butterfly 4 input → 4 outputs.

N^2 input



N^2 output

$\Longrightarrow \frac{N^2}{4}$ butterflies/stage.

- how much computation per butterfly is needed.

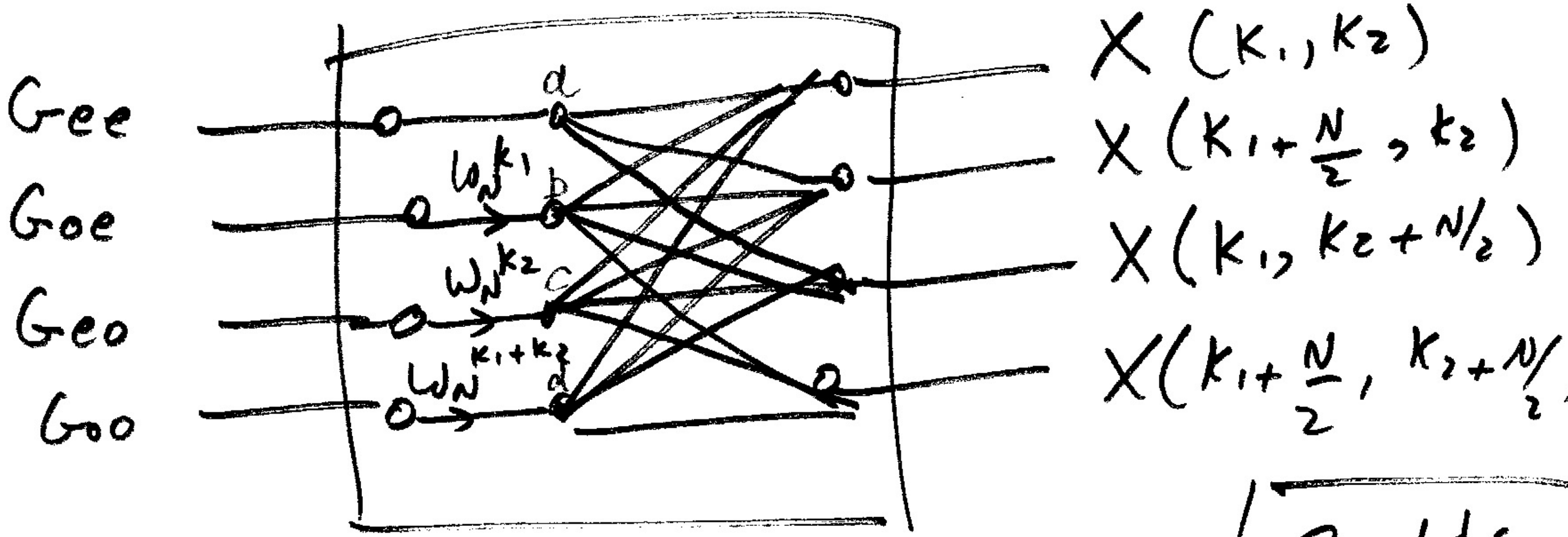
12 adds
3 mults

per butterfly

adds : $\frac{N^2}{4} \log_2 N \cdot 12 \simeq 3N^2 \log_2 N$

mults $\frac{N^2}{4} \log_2 N \cdot 3 \simeq \frac{3}{4} N^2 \log_2 N$

adds in butterfly change. $12 \longrightarrow 8$



$\Longrightarrow$ $\boxed{8 \text{ adds}}$

Define new intermediate variables.

$$A = a+b$$
$$B = a-b$$
$$C = c+d$$
$$D = c-d$$

then

$$X(k_1, k_2) = A + C$$
$$X(k_1+\tfrac{N}{2}, k_2) = B + D$$
$$X(k_1, \tfrac{N}{2}+k_2) = A - C$$
$$X(k_1+\tfrac{N}{2}, k_2+\tfrac{N}{2}) = B - D$$

→ adds. $12 \longrightarrow 8$

$$\frac{N^2}{4} \cdot \log_2 N \cdot 8 = 2N^2 \log_2 N.$$

adds.

	# of mult	# of adds
Row/Col.	$N^2 \log_2 N$	$2N^2 \log_2 N$
Vector Radix	$\frac{3}{4} N^2 \log_2 N.$	$2N^2 \log_2 N$
Direct	$\cancel{N^4}$	$\cancel{N^4}$

2D. **FIR** Filter Design

FIR → Finite Impulse Response.

1. Filter spec → application dependent
2. Design of Filter → determine coeffs of $h(n)$
3. Implementation
   - ASIC
   - DSP chip / TI, Analog.
     VSP ≈ TI / Equator
   - general purpose computing in C on PC, workstation.

easier to implement, Takes less time to implement