FOURIER TRANSFORM

Alternative Representation and Meaning

Why should we need an Alternative representation?

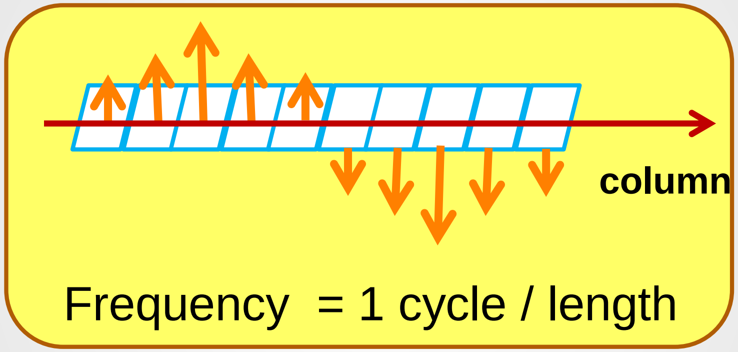
Need a New pattern to represent Information

which the **Normal representation cannot provide**

Pixel vs Frequency Representation

Pixel Representation (Normal Representation)

- Intensity Amplitude (level) @ pixel position (row,col)
- No relationship between the particular pixel and its neighbors



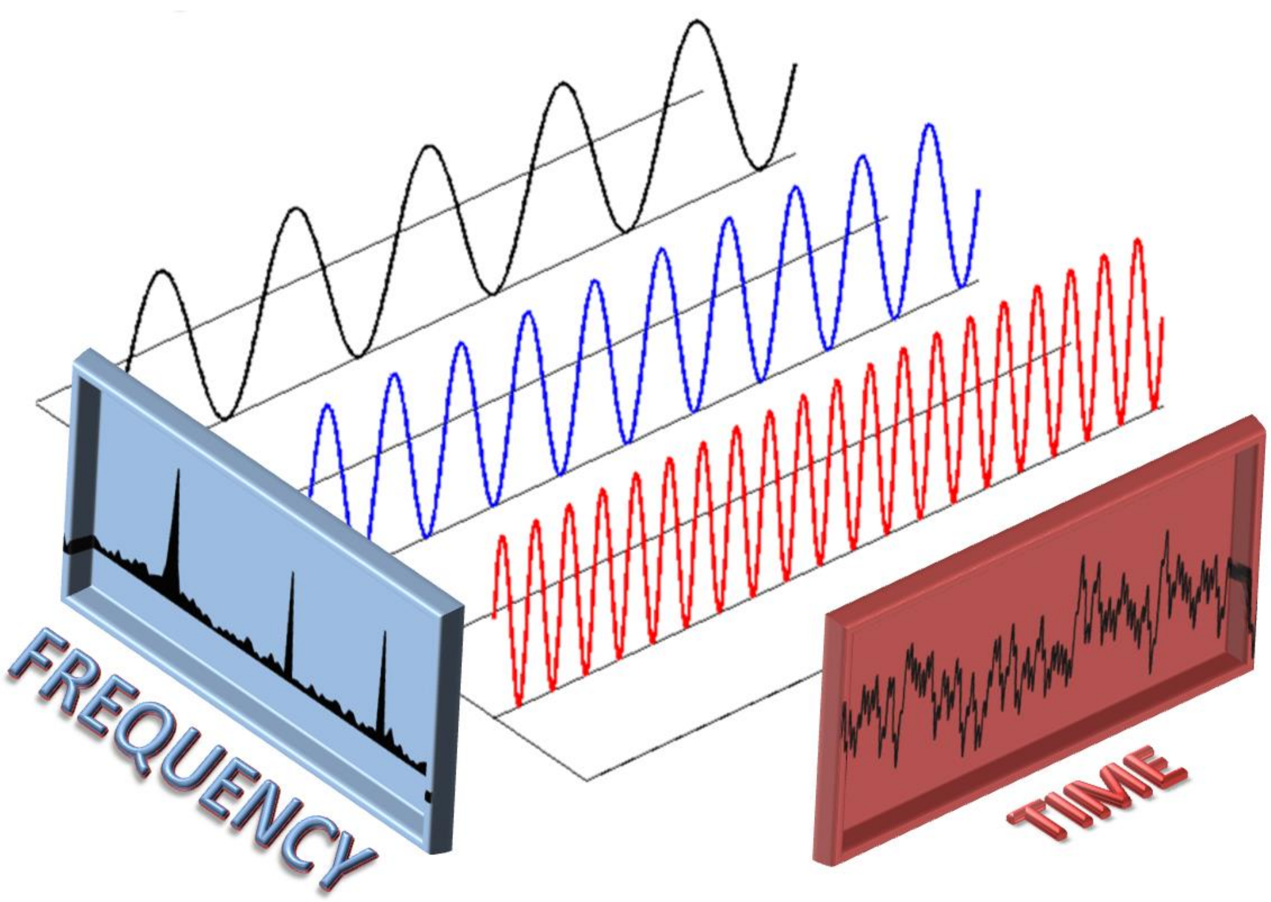
Frequency Representation (Alternative Representation)

- Rate of Intensity Change
- Measure how fast the intensity change over pixel position
  - In a period of width and height of an image

Why do we need

Fourier Transform?

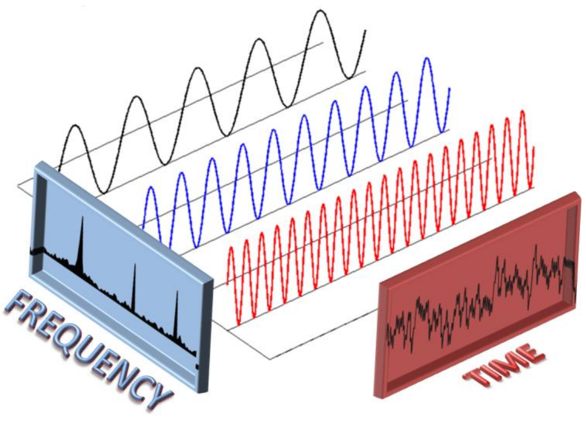
From Time-series to Frequency Pattern Ingredients



http://haitham.ece.illinois.edu/Pictures/sFFT.png

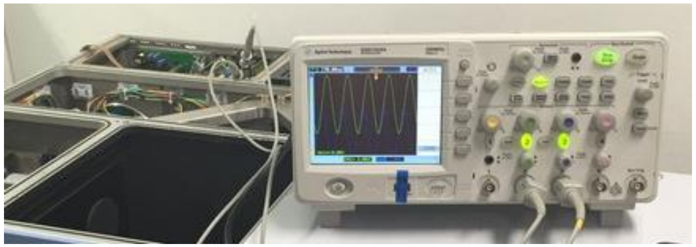
Why do we need Fourier Transform?

- A complicated signal or pattern
  - difficult to understand or analyze



- Better understanding
  - If complicated signal or pattern can be broken down into simple waves with different frequency.
  - ข้อมูลความถี่ สามารถอธิบายถึงความสัมพันธ์ระหว่างค่าของข้อมูลในเวลา หรือ ตำแหน่งที่ใกล้กัน ได้

Fourier Transform in our lives



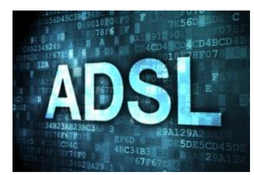
Chemistry
Material science

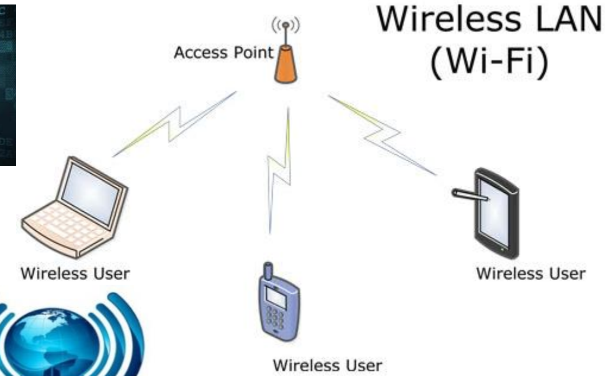
เครื่อง FTIR นสำหรับการตรวจสอบเอกลักษณ์ของสาร โดยใช้รังสีอินฟราเรดฉายผ่านไปที่วัตถุ เมื่อโมเลกุลของสารแต่ละชนิดจะดูดกลืนแสงอินฟราเรดในช่วงความถี่ที่มีค่าเฉพาะแตกต่างกันแล้ว ทำให้เกิดการเคลื่อนไหวของพันธะ ปรากฏการณ์ที่เกิดขึ้นจะถูกบันทึกเป็นสเปกตรัมซึ่งแสดงลักษณะเฉพาะตัวของสาร ที่เรียกว่า finger print ออกมา นำมาใช้ในการพิสูจน์เอกลักษณ์ของอัญมณีและเพชรแท้ในปัจจุบัน

OFDM Applications

- ADSL and VDSL broadband access via telephone network copper wires.
- IEEE 802.11a and 802.11g Wireless LANs.
- The Digital audio broadcasting systems EUREKA 147, Digital Radio Mondiale, HD Radio, T-DMB and ISDB-TSB.
- The terrestrial digital TV systems DVB-T, DVB-H, T-DMB and ISDB-T.
- The IEEE 802.16 or WiMax Wireless MAN standard.
- The IEEE 802.20 or Mobile Broadband Wireless Access (MBWA) standard.
- The Flash-OFDM cellular system.
- Some Ultra wideband (UWB) systems.

Telecommunications

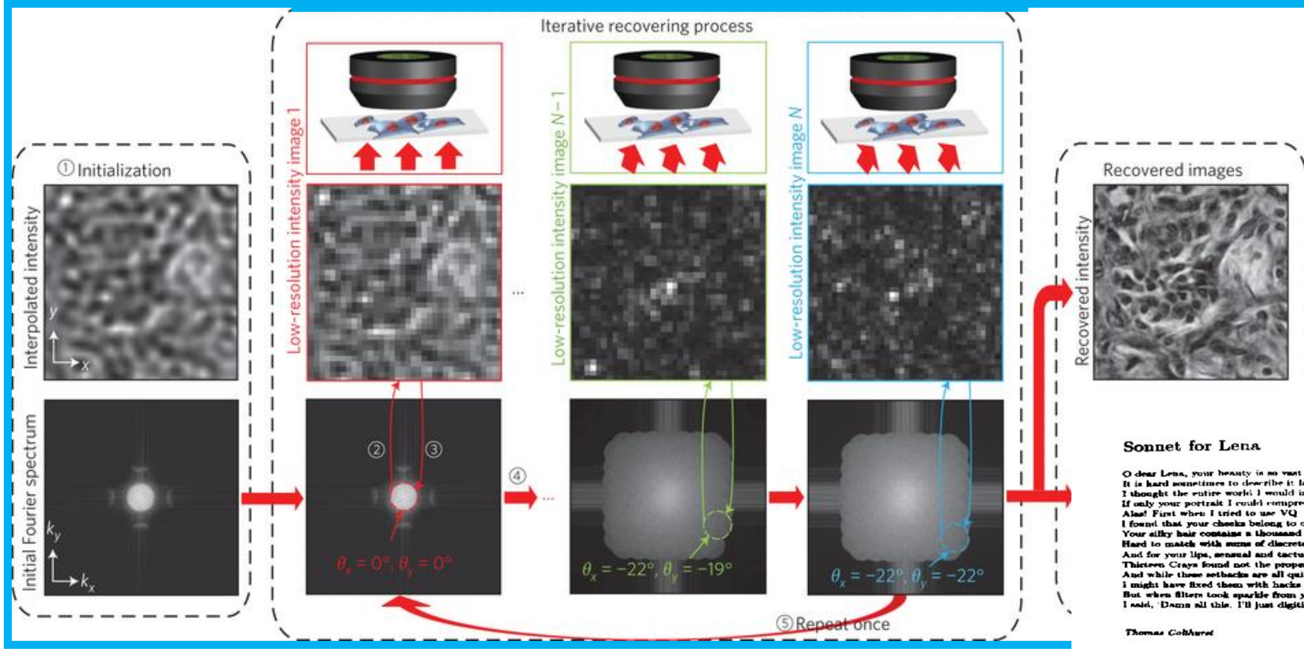




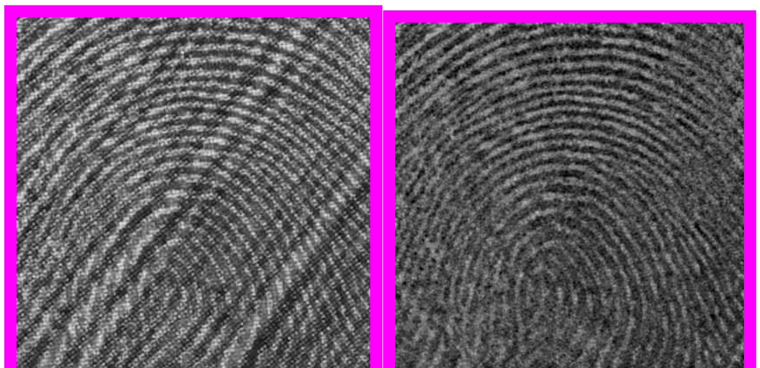
WiMAX

Fourier Transform in our lives

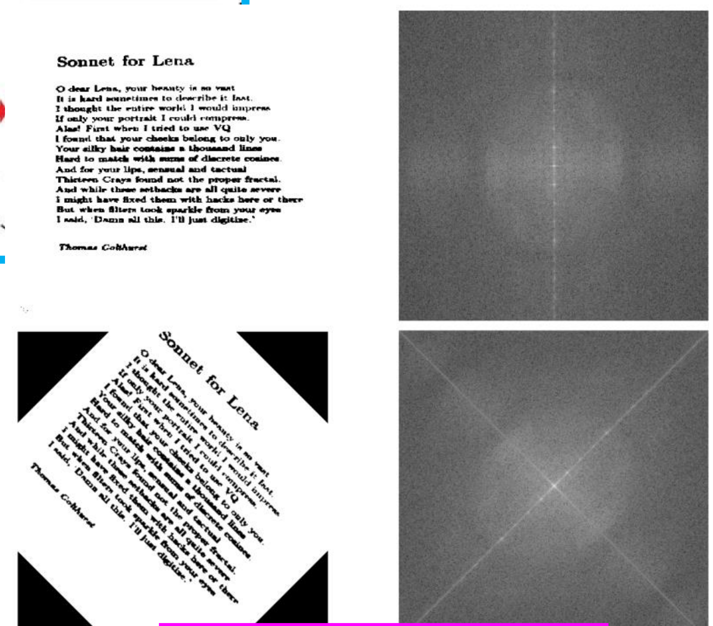
Enhancement (super resolution)



Movie Special Effects



Noise Filtering



Rotation Detection

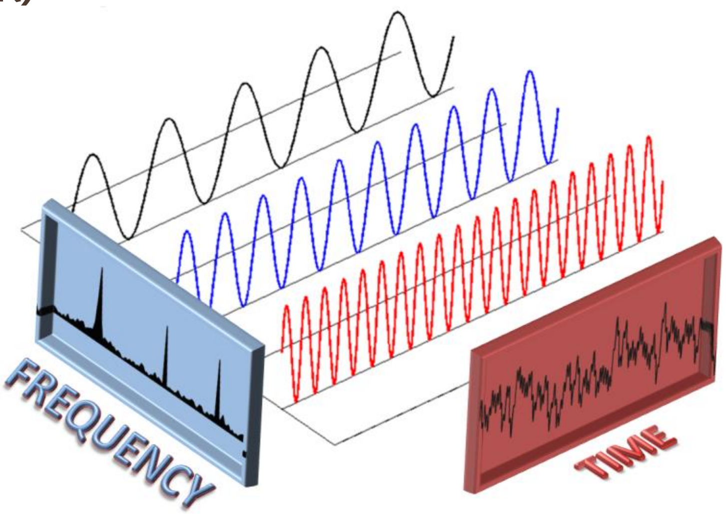
FOURIER TRANSFORM

Mathematical Pattern Detection for Decomposition

What are the Fourier Patterns?

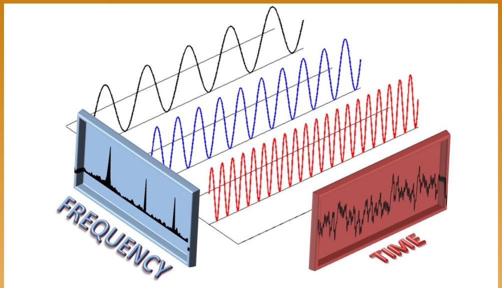
Fourier Representation (Fourier Patterns)

* Mathematical Pattern (Fixed Mask)
  * Based on sine and cosine

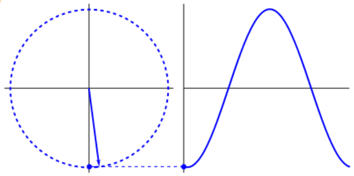


* Multi-Pattern Detection
  * Multiple masks
  * Pattern properties:
    * Must be reversable
      * (Forward transform <-> Inverse transform)
        * -> Orthonormal patterns

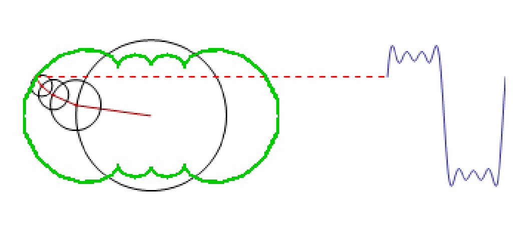
Fourier Representation (Fourier Patterns)



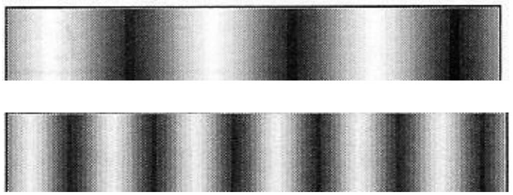
- 1D-Frequency Pattern (Frequency-Time)
  - Speed in time
    - Pattern of how fast amplitude change over time



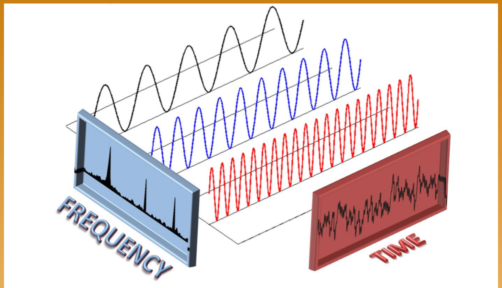
- Speeding in Circular path
  - ความเร็วในการก้าวเดิน (frequency) บนวงกลม a หน่วย (amplitude)



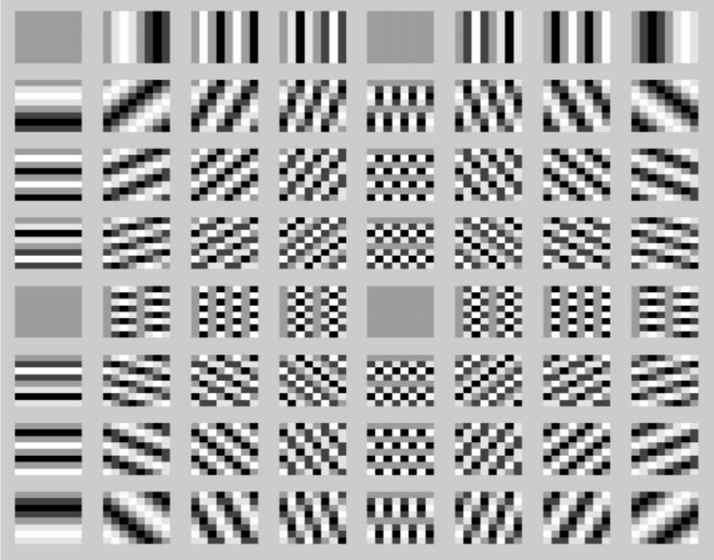
- Encoding
  - Pattern encoding
    - Frequency pattern / encoded amplitude



Fourier Representation (Fourier Patterns)



* 2D - Frequency Pattern
  * Encoding
    * Pattern encoding
      * Frequency pattern / encoded amplitude

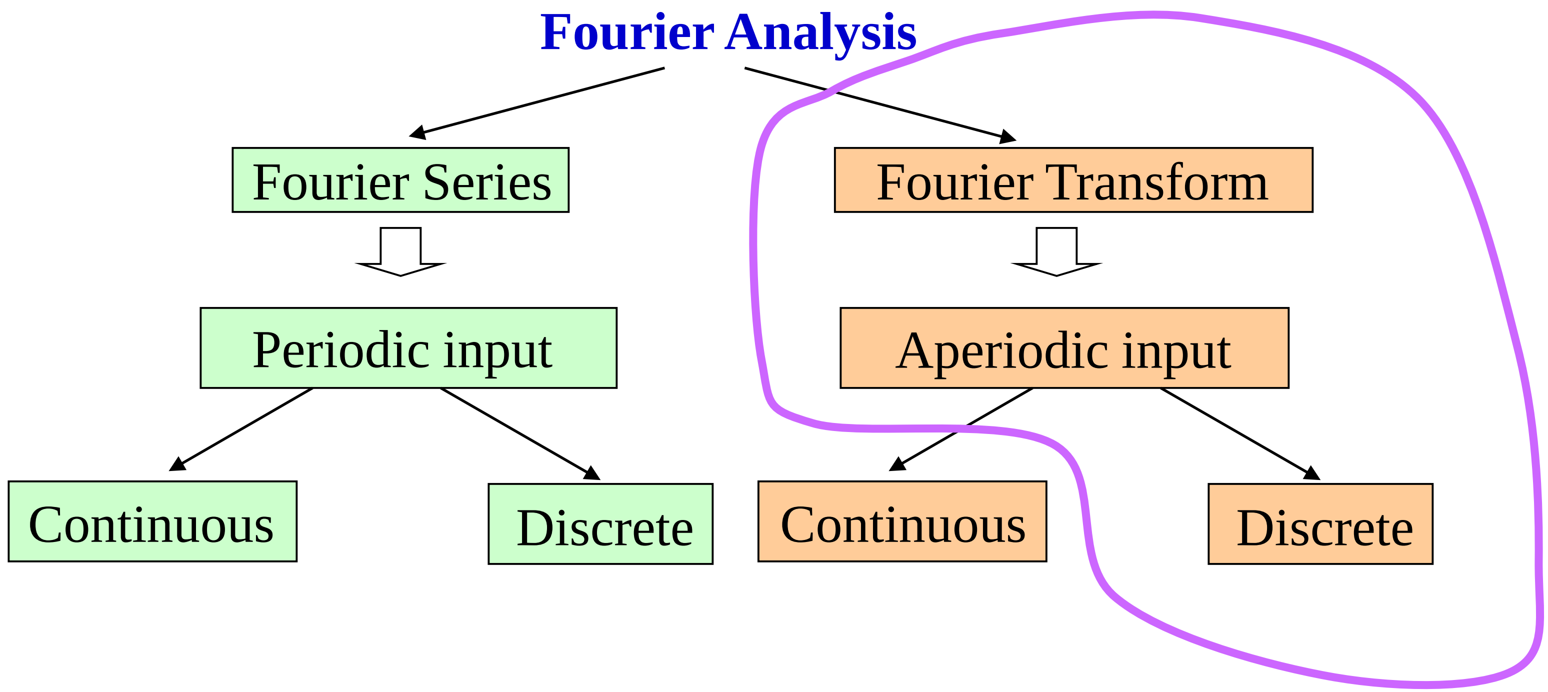


What are the values inside Fourier Patterns?

Fourier Transform Pattern (Filtering Mask)

- Fourier masks (Fixed mask)
  - Derived from mathematical models
    - Complex exponential set of multiple frequencies (f1, f2, …)
  - Fourier Operation
    - Dot product between input and each frequency mask
- Transform Results
  - Amplitude response (similarity between input and each frequency mask)
    - A(f1) , A(f2), A(f3), ….

Fourier Theory



Fourier Coefficients -> Amplitude response (Similarity)

Discrete Fourier Transform (DFT)

1D – DFT (Vector)	2D – DFT (Matrix)

Forward 1D DFT:

(Time (x) -> Frequency (u))

$$\vec{T}_u = \frac{1}{\sqrt{M}} e^{-j2\pi\left(\frac{ux}{M}\right)}$$

Inverse 1D DFT (IDFT):

(Frequency (u) -> Time (x))

$$\vec{T}_u^{-1} = \frac{1}{\sqrt{M}} e^{j2\pi\left(\frac{ux}{M}\right)}$$

Forward 2D DFT:

(Position (x,y) -> Frequency (u,v))

$$\vec{T}_{u,v} = \frac{1}{\sqrt{MN}} e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

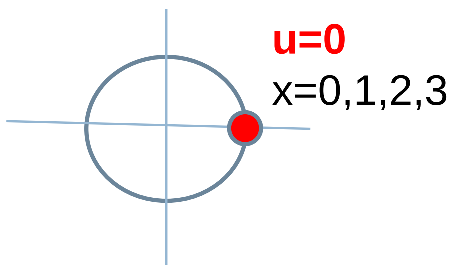
Inverse 2D DFT (IDFT):

(Frequency (u,v) -> Position (x,y))

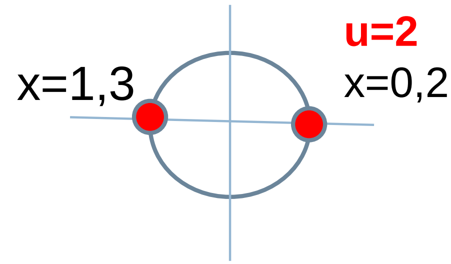
$$\vec{T}_{u,v}^{-1} = \frac{1}{\sqrt{MN}} e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Walking in circular path / Frequency Couple

ถ้ามีข้อมูลจำนวน N=4 ตำแหน่ง เทียบเท่ากับมีจำนวนก้าว 4 ก้าวเดิน บนวงกลม

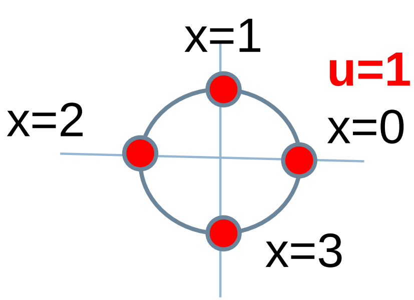


Complex Fourier pattern vector
[1 1 1 1]
Real



Complex Fourier pattern vector
[1 -1 1 -1]
Real

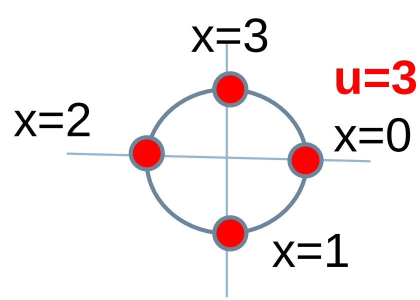
$$\vec{T}_u = e^{-j2\pi\left(\frac{ux}{M}\right)}$$



Complex Fourier pattern vector
[1 i -1 – i]

= [1 0 -1 0] + i [0 1 0 -1]

Real Imaginary



Complex Fourier pattern vector
[1 -i -1 i]

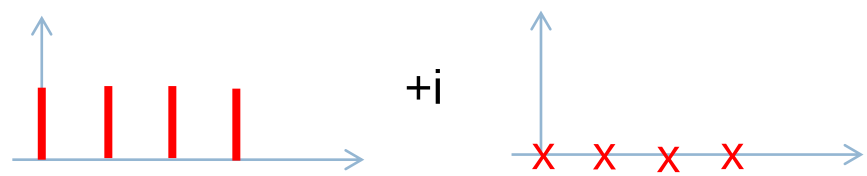
= [1 0 -1 0] + i [0 -1 0 1]

Real Imaginary

Walking in circular path / Frequency Masks

N = 4: (x=0:N-1) -> (u=0:N-1)

F(u=0) = [1 1 1 1]



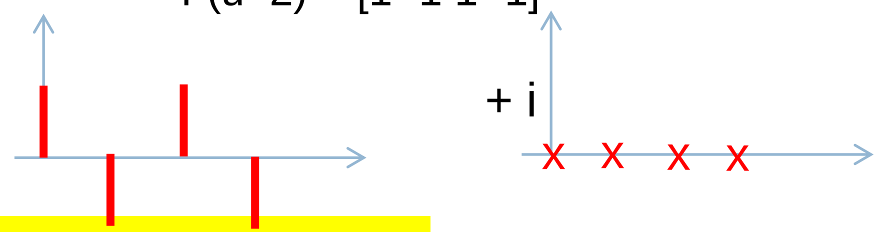
Cosine domain vector (Fourier pattern)

Sine domain vector (Fourier pattern)

[1 1 1 1]

[0 0 0 0]

F(u=2) = [1 -1 1 -1]



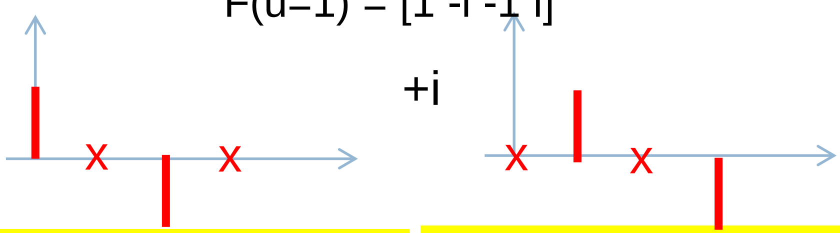
Cosine domain vector (Fourier pattern)

Sine domain vector (Fourier pattern)

[1 -1 1 -1]
Real

[0 0 0 0]
Imaginary

F(u=1) = [1 -i -1 i]



Cosine domain vector (Fourier pattern)

Sine domain vector (Fourier pattern)

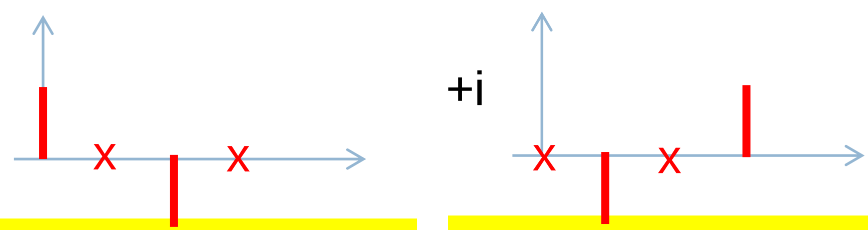
[1 0 -1 0]

Real

[0 1 0 -1]

Imaginary

F(u=3) = [1 -i -1 i]



Cosine domain vector (Fourier pattern)

Sine domain vector (Fourier pattern)

[1 0 -1 0]
Real

[0 -1 0 1]
Imaginary

FOURIER TRANSFORM

Pattern Encoding

(Fourier coefficients: Fourier Amplitude response)

Discrete Fourier Transform (DFT)

1D – DFT (Vector)

$$F(u) = \left(\vec{f} \cdot \vec{T}_u^*\right)$$

Forward 1D DFT:

$$\vec{T}_u = \frac{1}{\sqrt{M}} e^{-j2\pi\left(\frac{ux}{M}\right)}$$

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j2\pi\left(\frac{ux}{M}\right)}$$

Inverse 1D DFT (IDFT):

$$f(x) = \sum_{u=0}^{M-1} F(u) e^{j2\pi\left(\frac{ux}{M}\right)}$$

2D – DFT (Matrix)

$$F(u,v) = \left(\vec{f} \cdot \vec{T}_{u,v}^*\right)$$

Forward 2D DFT:

$$\vec{T}_{u,v} = \frac{1}{\sqrt{MN}} e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Inverse 2D DFT (IDFT):

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

การกรองและผลลัพธ์ของการกรองด้วยตัวกรองของ Fourier (Fourier Domain Pattern)

- ทำหน้าที่ **ค้นหาข้อมูล** (กรองสัญญาณ) ที่มีการเปลี่ยนแปลง **คล้ายกับรูปแบบ Fourier pattern** ที่ต้องการ หรือไม่

  - ถ้า**ไม่คล้ายเลย** ผลลัพธ์ที่ได้จะมีค่า**เป็น 0**
    - For 1D -> $F(u) = 0$
    - For 2D -> $F(u,v) = 0$

  - ถ้า**คล้าย** ผลลัพธ์ที่ได้จากการกรองจะมีค่า**ไม่เป็น 0**
    - For 1D -> $F(u) \neq 0$
    - For 2D -> $F(u,v) \neq 0$

1D Fourier Encoding

$$\vec{T}_u = e^{-j2\pi\left(\frac{ux}{M}\right)}$$

- การใช้งาน Fourier ในรูปแบบของ pattern ที่ใช้ในการเข้ารหัสข้อมูล
- 1D Fourier pattern
  - Complex number pattern

$$F(u) = \left(\vec{f} \cdot \vec{T}_u^*\right)$$

- Ex. การเข้ารหัสข้อมูล 1D vector (N=4)
  - ข้อมูล -> f = [2 5 5 8];
  - Fourier Encoding Pattern

T(u=0) = [1 1 1 1]

T(u=1) = [1 -i -1 i]

T(u=2) = [1 -1 1 -1]

T(u=3) = [1 i -1 -i]

Forward Transform Masks

การเข้ารหัส (Encoding Process)

F(u=0) = [1 1 1 1] · [2 5 5 8] = (2+5+5+8) = 20

F(u=1) = [1 -i -1 i]·[2 5 5 8] = (2-5i-5+8i) = -3+3i

F(u=2) = [1 -1 1 -1]·[2 5 5 8] = (2-5+5-8) = -6

F(u=3) = [1 i -1 -i] ·[2 5 5 8] = (2+5i-5-8i) = -3-3i

ผลลัพธ์การเข้ารหัส F= [20 (-3+3i) -6 (-3-3i)]

1D Fourier Decoding

$$\vec{T}_u = \frac{1}{M} e^{j2\pi\left(\frac{ux}{M}\right)}$$

- 1D Encoded vector
  - F = [20 (-3-3i) -6 (-3+3i)]
- Decoding Process
  - Fourier Decoding Pattern ((1/N) F')

T'(u=0) = (1/4) [1 1 1 1]

T'(u=1) = (1/4) [1 i -1 -i]

T'(u=2) = (1/4) [1 -1 1 -1]

T'(u=3) = (1/4) [1 -i -1 +i]

Inverse Transform Masks

การถอดรหัส (Decoding Process)

X(u=0) = (1/4) [1 1 1 1]·[20 (-3+3i) -6 (-3-3i)] = 2

X(u=1) = (1/4) [1 i -1 -i] ·[20 (-3+3i) -6 (-3-3i)] = 5

X(u=2) = (1/4) [1 -1 1 -1] ·[20 (-3+3i) -6 (-3-3i)] = 5

X(u=3) = (1/4) [1 -i -1 i] ·[20 (-3+3i) -6 (-3-3i)] = 8

ผลลัพธ์การถอดรหัส X = [2 5 5 8]

Inverse Transform Respons -> recover input

Exercise#1:

- 1D Fourier Encoding (Fourier Transform)
  - Last 4 digits of student ID -> Fourier Coefficients

T(u=0) = [1 1 1 1]

T(u=1) = [1 -i -1 i]

T(u=2) = [1 -1 1 -1]

T(u=3) = [1 i -1 -i]

- 1D Fourier Decoding (Inverse Fourier Transform
  - Fourier Coefficients -> Last 4 digits of student ID

T '(u=0) = (1/4) [1 1 1 1]

T '(u=1) = (1/4) [1 i -1 -i]

T '(u=2) = (1/4) [1 -1 1 -1]

T '(u=3) = (1/4) [1 -i -1 +i]

FOURIER TRANSFORM

2D pattern

Extend Frequency transform to

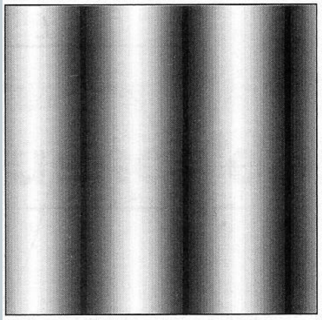
2D input

What would be the changes?

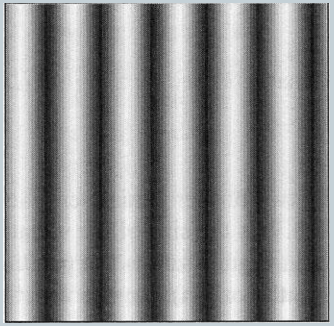
Changing in column (v) as Image of Sine wave

1D sine wave

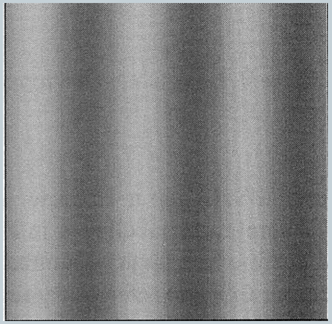
$$f(x,y) = 128 + A\sin\left(\frac{2\pi vy}{N-1} + \phi\right)$$



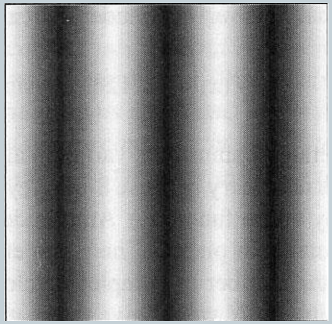
$v = 3;\ A = 127; \phi = 0$



$v = 6;\ A = 127; \phi = 0$



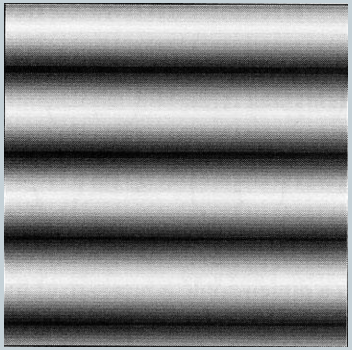
$v = 3;\ A = 51;\ \phi = 0$



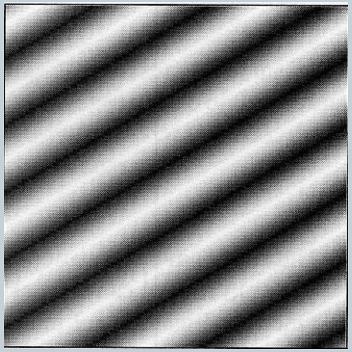
$v = 3;\ A = 127;\ \phi = 90^\circ$

Changing in column (v) and both directions (u and v) as Image of Sine wave

2D sine wave

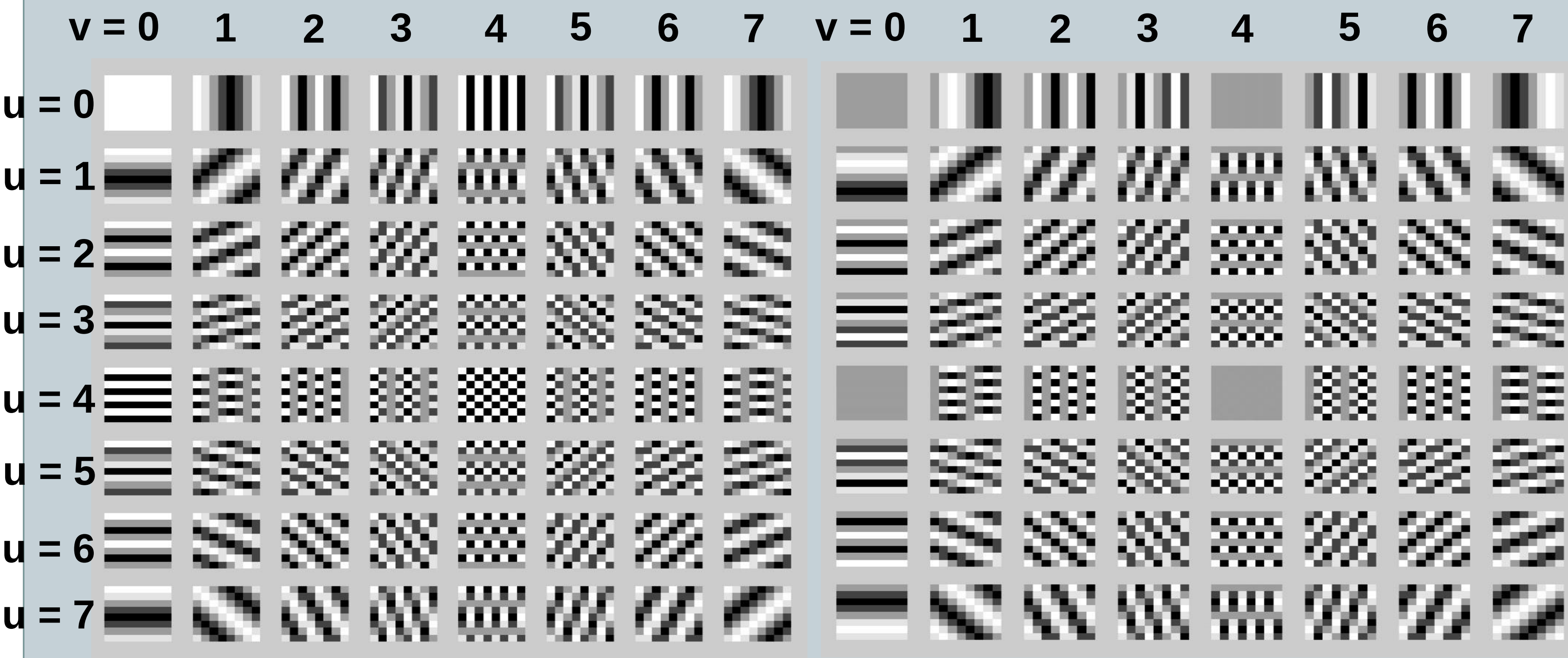


$u = 4; v = 0; A = 127; \phi = 0$



$u = 5; v = 3; A = 127; \phi = 0$

2D DFT Basic Functions
(N=8)



cosine basis functions

(Cosine Domain Matrix)

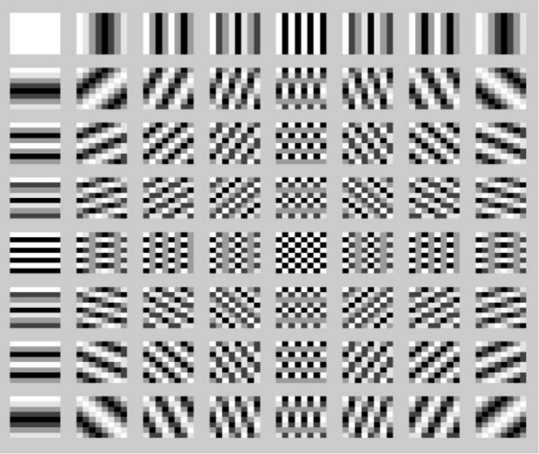
sine basis functions

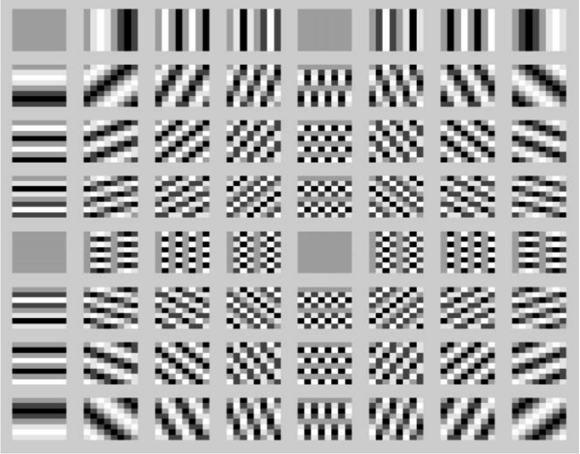
(Sine Domain Matrix)

ตัวกรองของ Fourier (Fourier Domain Pattern)

- 2D Periodic pattern (Domain matrix) ของ Fourier
  - Periodic pattern อยู่ในลักษณะ matrix ที่มีการเปลี่ยนแปลงใน 2 ทิศทาง คือ
    - ทิศทาง row & column
  - ถ้านำค่าของ Fourier Domain matrix มาแสดงในลักษณะของภาพ
    - จะเห็นว่า periodic pattern เป็นตัวแทนของ **ความถี่** และ **ทิศทาง** ของการเปลี่ยนแปลง ซึ่งสามารถมองว่าเป็นรูปแบบและแนวทิศของเส้นในภาพ ได้เช่นกัน

Cosine periodic pattern





Sine periodic pattern

ตัวอย่างของ periodic pattern ของ**ข้อมูลขนาด 8x8 พิกเซล**

ซึ่งจะทำให้มี Fourier Domain Matrix (Pattern) ถึง **64 pattern ของ cosine** และ **64 pattern ของ sine wave**

2D Fourier Encoding

$$\vec{T}_{u,v} = \frac{1}{\sqrt{MN}} e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

ข้อมูล -> $f = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$

$$F(u,v) = \left([f] \cdot \vec{T}^{*}_{u,v} \right)$$

2D Forward Transform Masks

T[0,0]	T[0,1]	T[0,2]	T[0,3]
T[1,0]	T[1,1]	T[1,2]	T[1,3]
T[2,0]	T[2,1]	T[2,2]	T[2,3]
T[3,0]	T[3,1]	T[3,2]	T[3,3]

T[0,0] =

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

T[0,1] =

1	- i	-1	i
1	- i	-1	i
1	- i	-1	i
1	- i	-1	i

T[1,0] =

1	1	1	1
- i	- i	- i	- i
-1	-1	-1	-1
i	i	i	i

T[1,1] =

1	- i	-1	i
- i	-1	i	1
-1	i	1	- i
i	1	- i	-1

2D Fourier Encoding

$$\vec{T}_{u,v} = \frac{1}{\sqrt{MN}} e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

- การใช้งาน 2D Fourier ในรูปแบบของ pattern ที่ใช้ในการเข้ารหัสข้อมูล
- 2D Fourier pattern
  - Complex number pattern

$$F(u,v) = \left([f] \cdot \vec{T}^{*}_{u,v} \right)$$

- Ex. การเข้ารหัสข้อมูล 2D vector (M x N = 4 x 4)
  - ข้อมูล -> $f = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$
  - Fourier Encoding Pattern

$$T(0,0) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Forward Transform Masks

การเข้ารหัส (Encoding Process)

$$F(u=0, v=0) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} = 8$$

Exercise #2: 2D Fourier Encoding

$$\vec{T}_{u,v} = \frac{1}{\sqrt{MN}} e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Forward Transform Masks

ข้อมูล -> $f = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$

$$F(u,v) = \left([f] \cdot \vec{T}^*_{u,v} \right)$$

	1	- i	-1	i
T[0,1] =	1	- i	-1	i
	1	- i	-1	i
	1	- i	-1	i

	1	- i	-1	i
T[0,1] =	1	- i	-1	i
	1	- i	-1	i
	1	- i	-1	i

	1	- i	-1	i
T[1,1] =	- i	-1	i	1
	-1	i	1	- i
	i	1	- i	-1

Exercise #2: 2D Fourier Decoding

$$\vec{T'}_{u,v} = \frac{1}{\sqrt{MN}} e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Forward Transform Masks

ข้อมูล -> $f = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$

$$F(u,v) = \left([f] \cdot \vec{T}^{*}_{u,v} \right)$$

	1	1	1	1			1	i	-1	- i
T' [0,0] =	1	1	1	1		T' [0,1] =	1	i	-1	- i
	1	1	1	1			1	i	-1	- i
	1	1	1	1			1	i	-1	- i
	1	1	1	1			1	i	-1	- i
T' [1,0] =	i	i	i	i		T' [1,1] =	i	-1	- i	1
	-1	-1	-1	-1			-1	- i	1	i
	- i	- i	- i	- i			- i	1	i	-1

How would arrangement of **input data** or **transform result**

(การจัดกลุ่มข้อมูลอินพุท หรือ ผลลัพธ์จากการแปลงข้อมูล)

effect Calculation Speed?

(ส่งผลต่อความเร็วในการคำนวณอย่างไร)

2D DFT Separable Property

$$\boxed{F(u,v)=\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}}$$

$$O(n)=(MN)_{x,y}\,x(MN)_{u,v}$$

$$F(u,v)=\frac{1}{M}\sum_{x=0}^{M-1}\underbrace{\left[\frac{1}{N}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{vy}{N}\right)}\right]}_{\text{1D DFT (row) -> F(x,v)}}e^{-j2\pi\left(\frac{ux}{M}\right)}$$

1D DFT (row) -> F(x,v)

$$\underbrace{F(u,v)=\frac{1}{M}\sum_{x=0}^{M-1} F(x,v)e^{-j2\pi\left(\frac{ux}{M}\right)}}_{\text{1D DFT (column)}}$$

1D DFT (column)

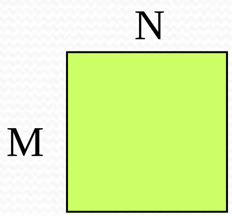
$$O(n)_{row}=MN^2$$

$$+$$

$$O(n)_{Column}=M^2N$$

$$O(n)_{Total}=MN^2+M^2N$$

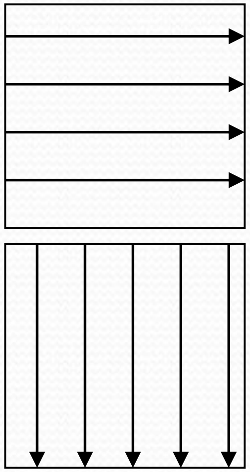
Fast Fourier Transform (FFT)



2D image

1D DFT (y-direction)

1D DFT (x-direction)



$$O(n)_{row} = MN^2$$

+

$$O(n)_{Column} = M^2N$$

$$O(n) = (MN)_{x,y}\, x(MN)_{u,v}$$

$$O(n)_{Total} = MN^2 + M^2N$$

1D DFT $\longrightarrow$ 1D Fast Fourier Transform (FFT)

⇩

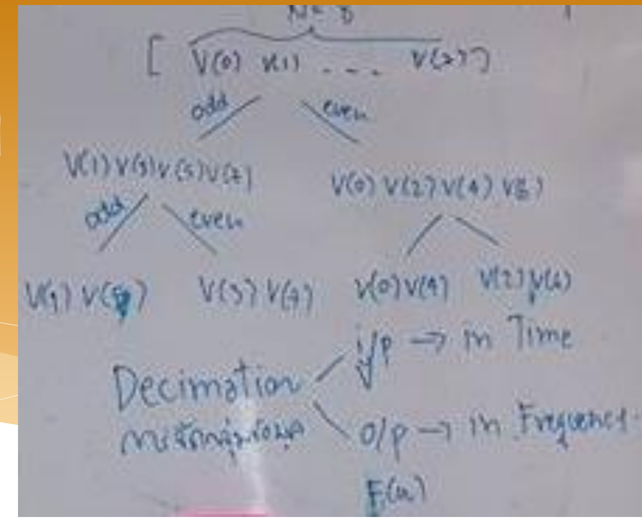
Separate discrete data points into several groups

(decimation in Time or Frequency with Radix I / II / III / IV)

⇩

Calculate DFT in hierarchical manner

Fast Fourier Transform



* จัดกลุ่มข้อมูลในรูปแบบ **tree**
* เพื่อให้การคำนวณเป็นแบบ **hierarchy** ตาม tree
  * โดย**เริ่มคำนวณจาก leaf** ของ tree ซึ่งเป็น**กลุ่มข้อมูลขนาดเล็ก**ก่อน
  * แล้วจึง**รวมผลของการคำนวณกลุ่มข้อมูลขนาดเล็ก**เพื่อให้ได้**ผลลัพธ์ของกลุ่มข้อมูลขนาดใหญ่ขึ้นในชั้นขึ้นไปจนถึง root**
* ผลลัพธ์ที่ต้องการจริงๆคือ
  * เพื่อ **optimize** **จำนวนครั้งของการคูณ (O(n))** ให้เหลือน้อยที่สุด
* ข้อจำกัดของ FFT
  * จำนวนข้อมูลที่ต้องการจัดกลุ่มจะต้องเป็นไปตามเงื่อนไขที่กำหนด
    * เช่น Radix II -> N = power of 2
    * ถ้าจำนวนข้อมูลไม่เป็น N = power of 2
      * ต้องทำ zero padding เพื่อให้ได้ N = power of 2

Fast Fourier Transform

(decimation in Time with Radix II)

If we let

$$W_N = e^{-i2\pi/N}$$

the Discrete Fourier Transform can be written

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

If N is a multiple of 2, $N = 2M$ for some positive integer M, substituting $2M$ for N gives

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

Fast Fourier Transform

Separating out the *M* even and *M* odd terms,

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

Notice that

$$W_{2M}^{\mathrm{u}(2x)} = e^{-i2\pi\upsilon(2\mathrm{x})/(2M)} = e^{-i2\pi\upsilon\mathrm{x}/M} = W_M^{ux}$$

and

$$W_{2M}^{u(2x+1)} = e^{-i2\pi u(2\mathrm{x}+1)/(2M)} = e^{-i2\pi u\,\mathrm{x}/M}\, e^{-i2\pi u/2M} = W_M^{ux} W_{2M}^{u}$$

So,

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

Fast Fourier Transform

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

Can be written as

$$F(u) = \frac{1}{2}\left\{F_{even}(u) + F_{odd}(u) W_{2M}^{u}\right\}$$

Simplifying further, the first M terms of the Fourier transform of 2M items can be computed by

$$F(u) = \frac{1}{2}\left\{F_{even}(u) + F_{odd}(u) W_{2M}^{u}\right\} \qquad 0 \le u \le M-1$$

and the last M terms can be computed by

$$F(u) = \frac{1}{2}\left\{F_{even}(u) - F_{odd}(u) W_{2M}^{u}\right\} \qquad M \le u \le 2M-1$$

Fast Fourier Transform

If *M* is itself a multiple of 2, do it again!

If *N* is a power of 2, keep recursively subdividing until you have one element, which is its own Fourier Transform.

```
FourierTransform FFT(Signal f)
{
   if (length(f) == 1) return f;
   evenpart = FFT(EvenTerms(f));
   oddpart  = FFT( OddTerms(f));
   for (s = 0; s < length(f) / 2; s++) {
      result[s  ] = evenpart[s] + W_2M ^ s * oddpart[s];
      result[s+M] = evenpart[s] - W_2M ^ s * oddpart[s];
   }
}
```

From theory

to

Fast Fourier Transform

Hierarchical Calculation of FFT

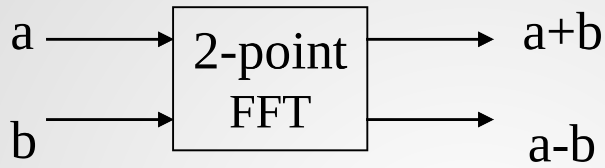
(Butterfly Structure/Algorithm)

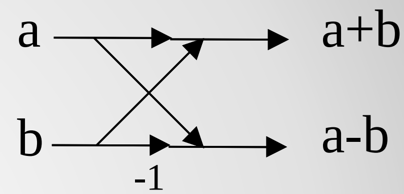
DIT-FFT (Scalable) (1)

2-point FFT

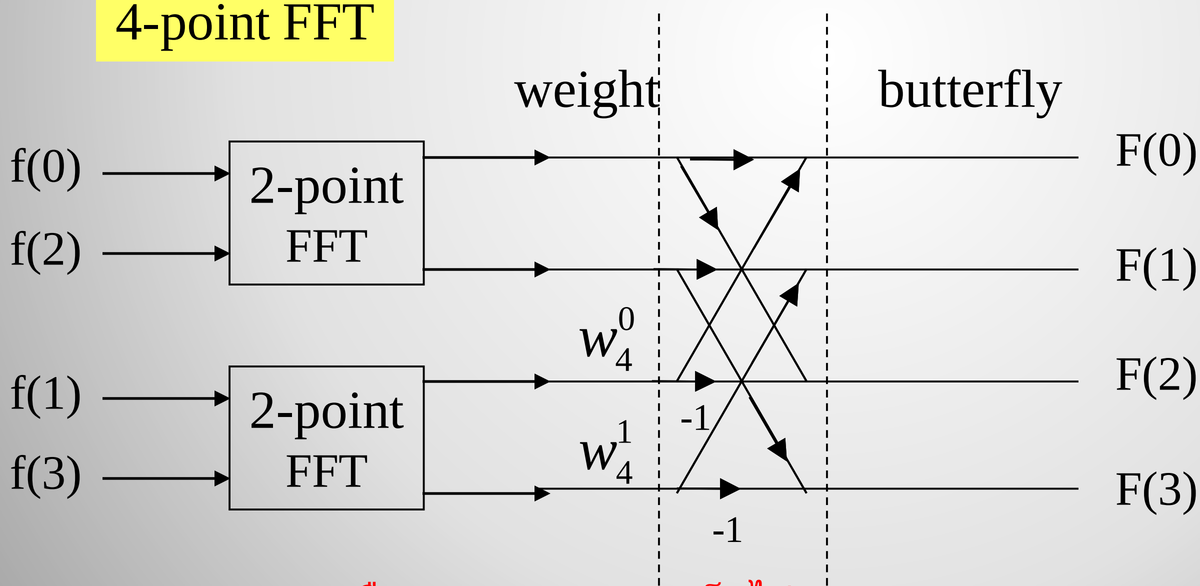
butterfly

เขียนการคำนวณในรูปของการบวก
ซึ่งแทนด้วยตำแหน่งที่ลูกศรตัดกัน





4-point FFT



$F(u=0,1) =$
$\{F_{even}(u) + F_{odd}(u)W_{2M}\}$

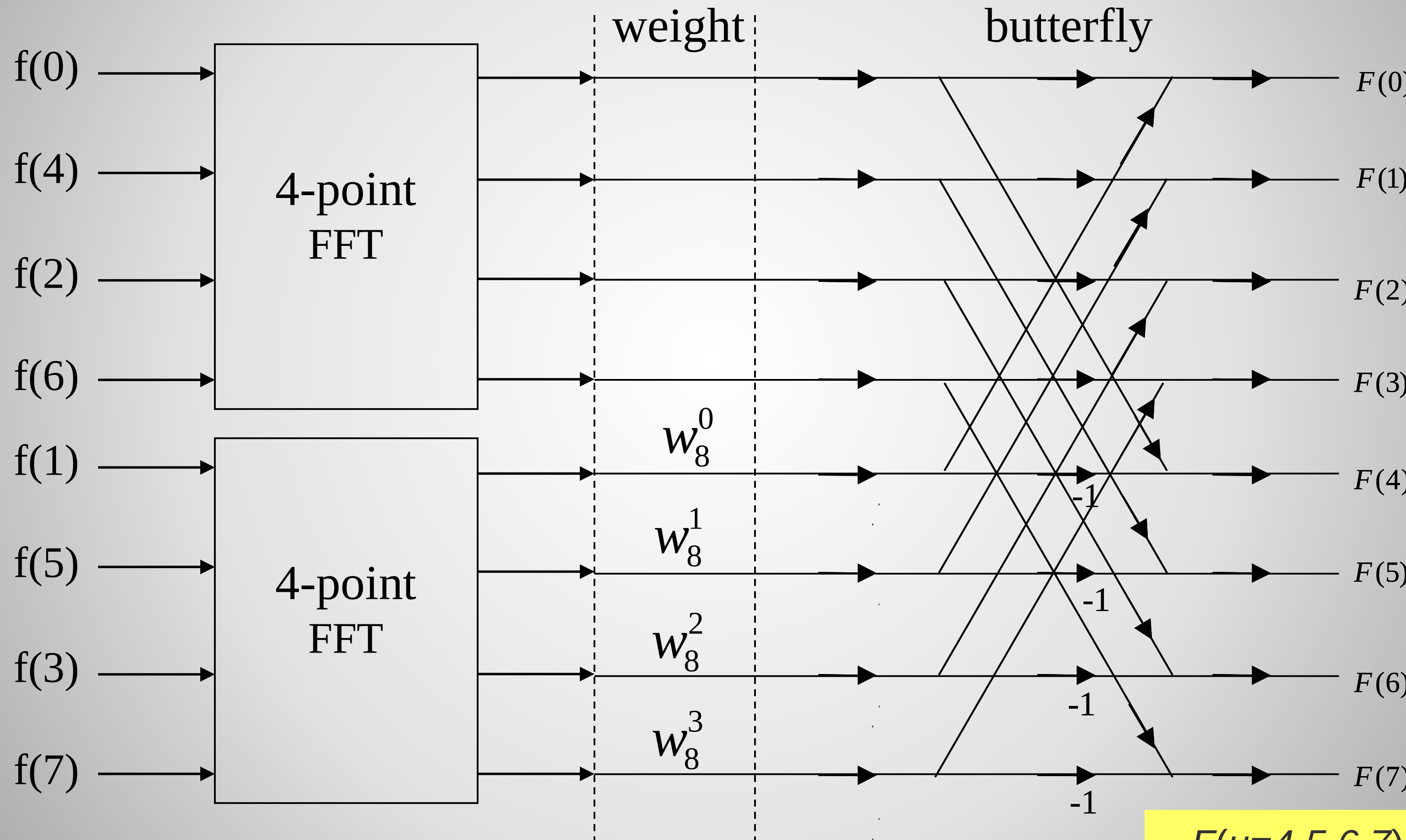
$F(u=2,3) =$
$\{F_{even}(u) - F_{odd}(u)W_{2M}\}$

Scaling (1/N) หรือ (1/2M) สามารถเก็บไปคำนวณตอนทำ inverse Fourier Transform

8-point FFT

$F(u=0,1,2,3) =$

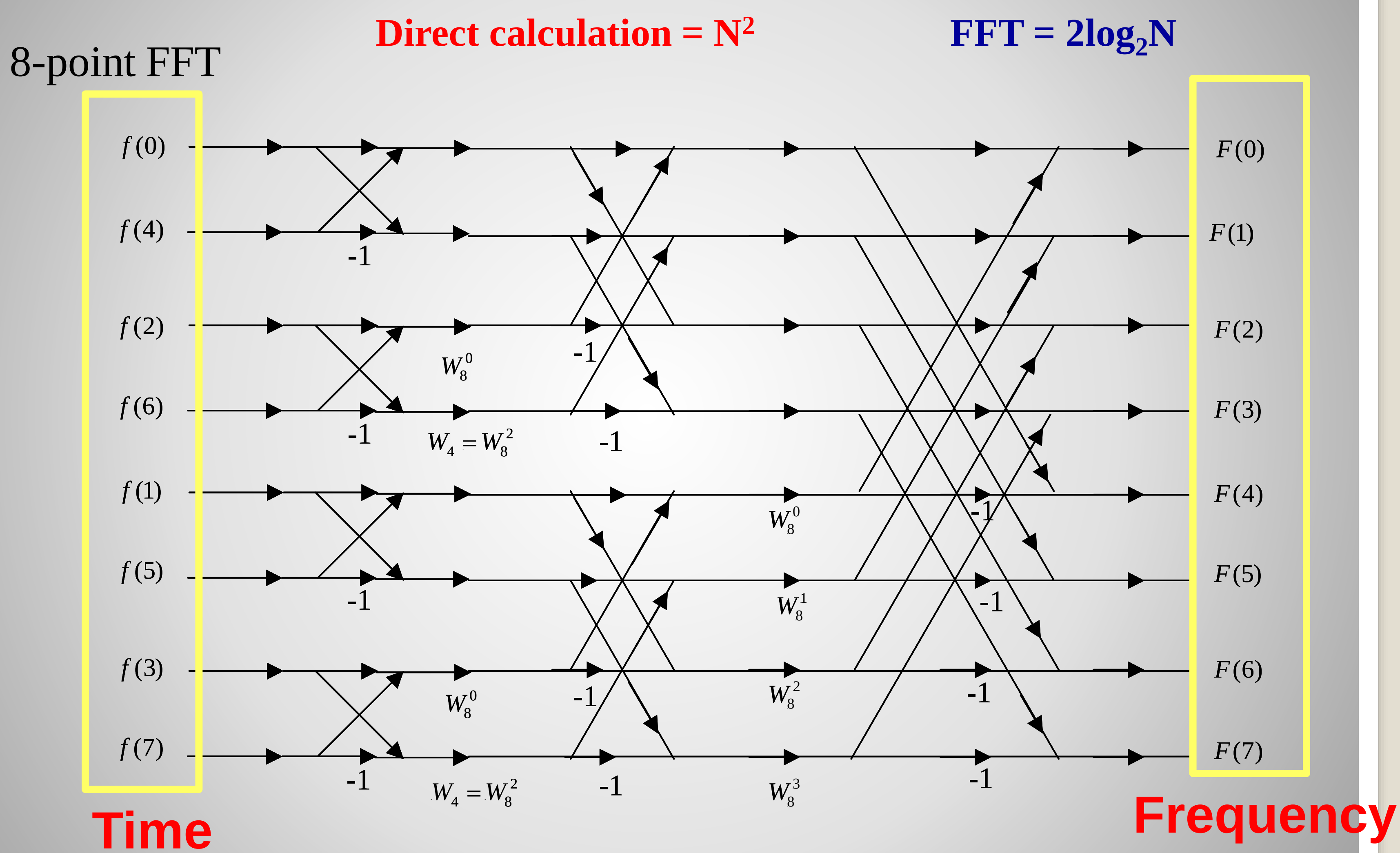
$\{F_{even}(u) + F_{odd}(u)W_{2M}\}$



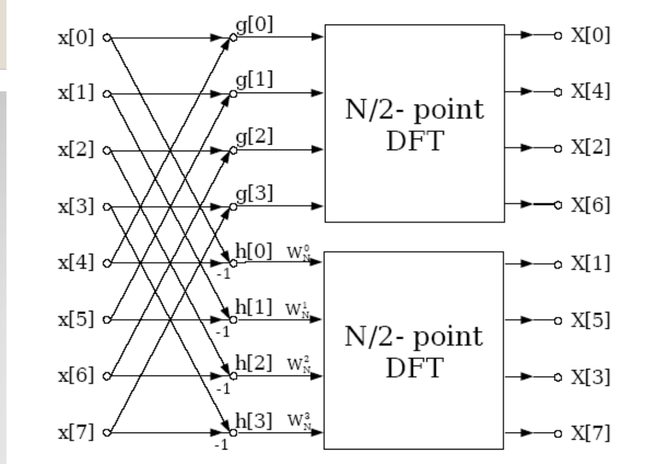
$F(u=4,5,6,7) =$

$\{F_{even}(u) - F_{odd}(u)W_{2M}\}$

DIT-FFT (Scalable) (2)



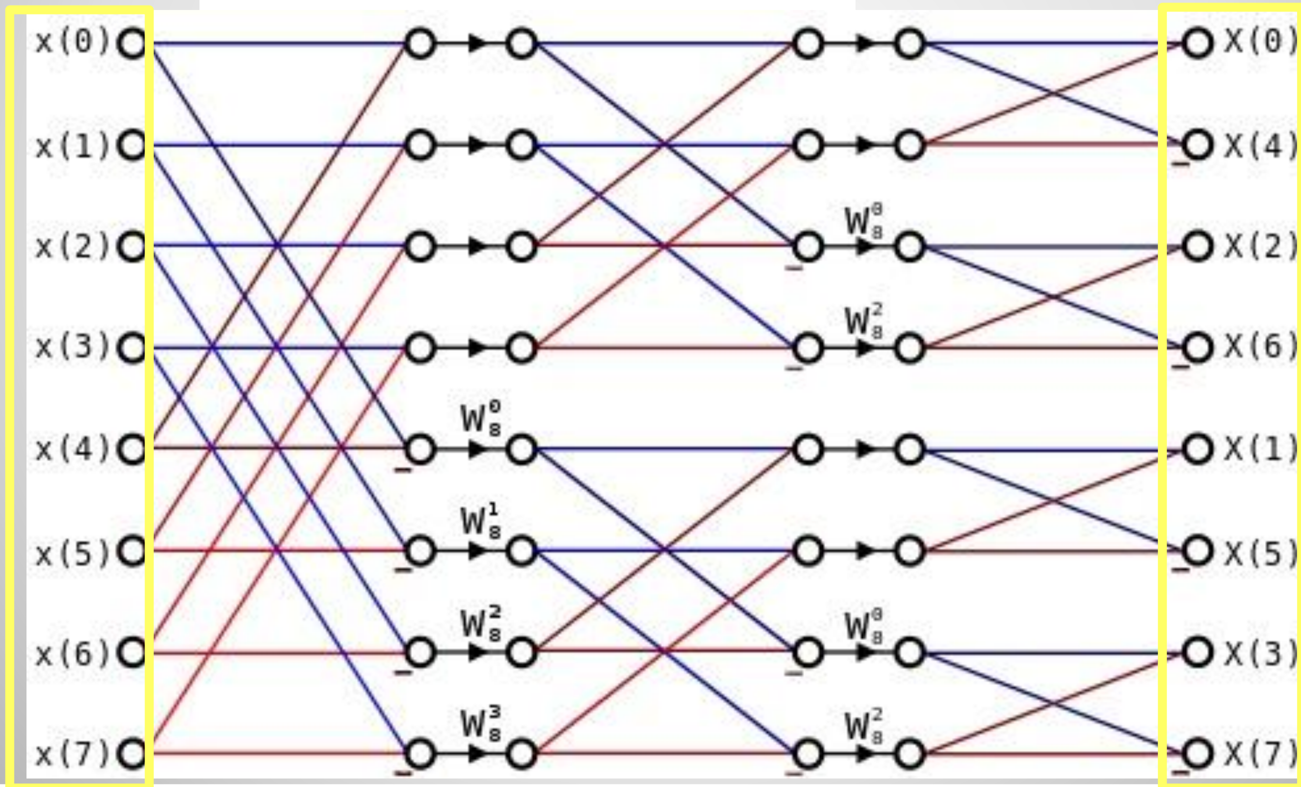
Fast Fourier Transform (DIT-FFT)



กลับโครงสร้างผีเสื้อระหว่าง DIT กับ DIF

Time

Frequency



Fast Fourier Transform (DIF-FFT)

Fast Fourier Transform

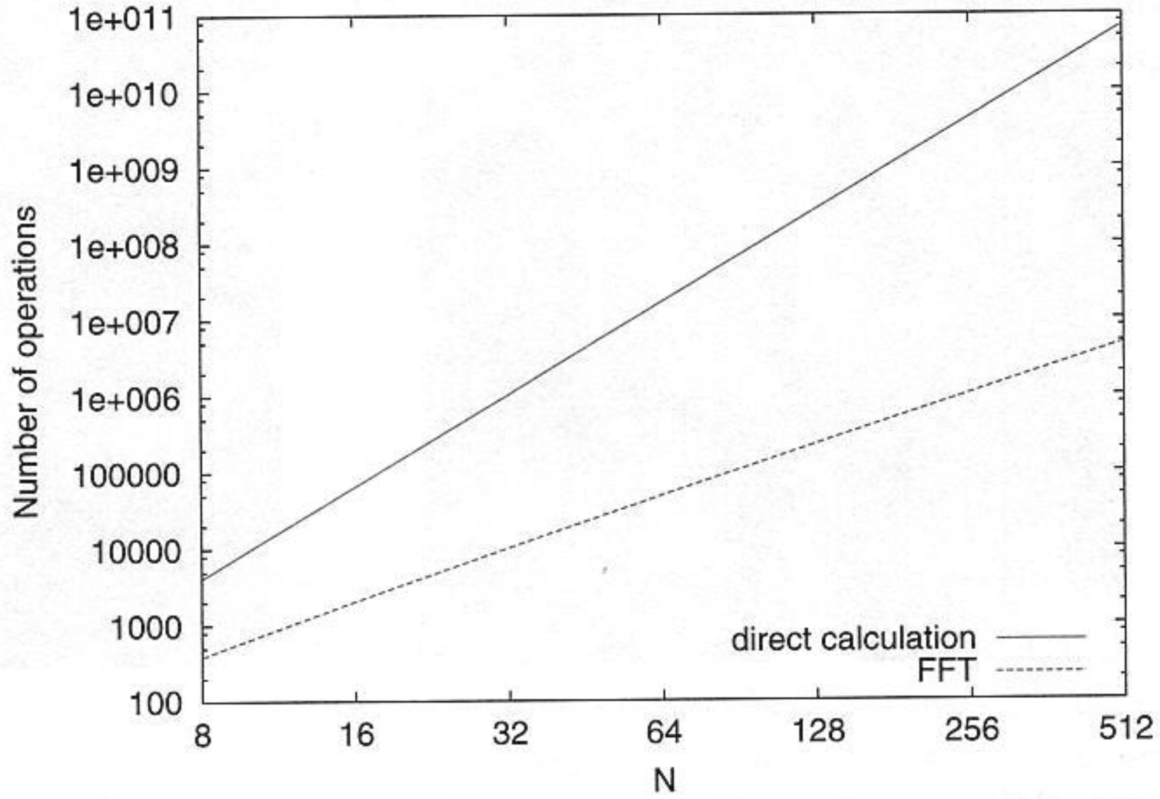
Computational Complexity:

1D-Discrete Fourier Transform → $O(N^2)$

1D-Fast Fourier Transform → $O(2 \log_2 N)$

Remember: The Fast Fourier Transform is just a faster **algorithm** for computing the Discrete Fourier Transform — it does *not* produce a different result.

Processing Time (DFT vs FFT)



Relationship between Convolution (in pixel) and DFT (in frequency)

$$\boxed{f(x,y) * h(x,y) \Leftrightarrow F(u,v)H(u,v)}$$

$$g(x,y) = f(x,y) * h(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n)$$

$$DFT[g(x,y)] = G(u,v) = \frac{1}{MN}\sum_{y=0}^{M-1}\sum_{x=0}^{N-1} g(x,y)W_M^{ux}W_N^{vy}; \qquad W_M^{ux} = e^{-j\frac{2\pi ux}{M}};\ W_M^{vy} = e^{-j\frac{2\pi vy}{N}}$$

$$= \sum_{y=0}^{M-1}\sum_{x=0}^{N-1}\left[\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n)\right]W_M^{ux}W_N^{vy}$$

$$= \frac{1}{MN}\sum_{y=0}^{M-1}\sum_{x=0}^{N-1}\left[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m,y-n)\right]W_M^{mu}W_N^{nv}W_M^{(x-m)u}W_N^{(y-n)v}$$

$$= \frac{1}{MN}\left[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)W_M^{mu}W_N^{nv}\right]\sum_{y=0}^{M-1}\sum_{x=0}^{N-1} h(x-m,y-n)W_M^{(x-m)u}W_N^{(y-n)v}$$

$$= \frac{1}{MN}F(u,v)H(u,v)$$

$$\boxed{IDFT\left[\frac{1}{MN}F(u,v)H(u,v)\right] = g(x,y) = f(x,y) * h(x,y)}$$

Comparison of Convolution and FFT

$$g(x,y) = f(x,y) * h(x,y) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m, y-n)$$

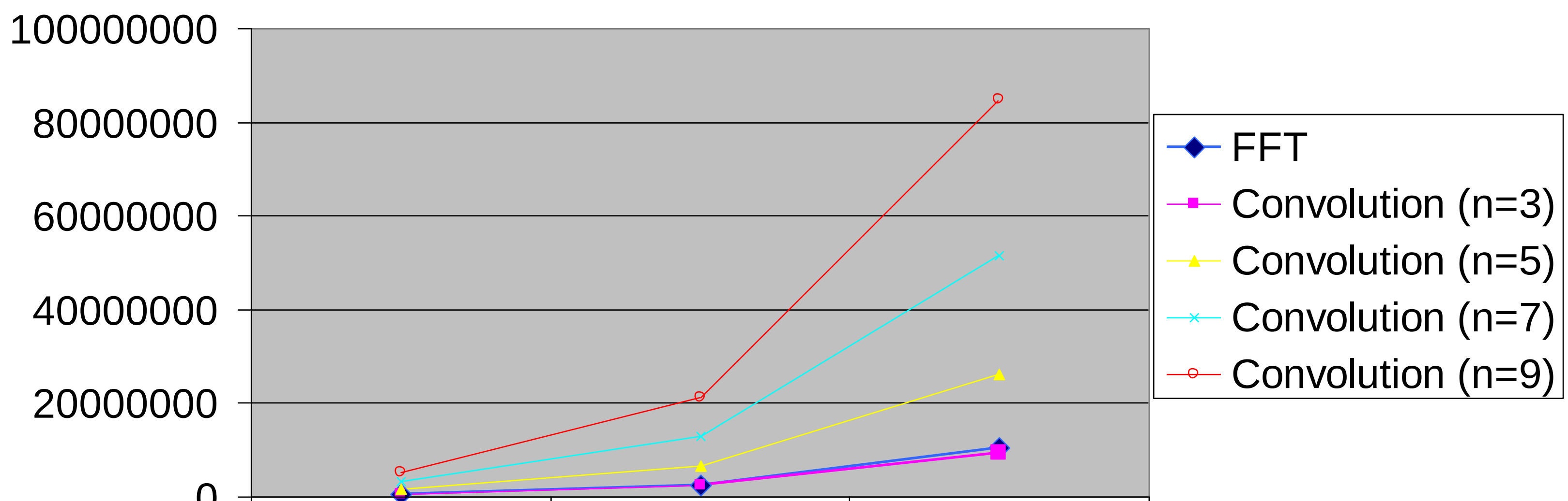
$$IDFT\left[\frac{1}{MN}F(u,v)H(u,v)\right]$$

	Convolution	FFT
Computational Process	Simple	Complex with many steps
Number of Multiplications	$N^2 n^2$	$< N(\log_2 N)$

N x N -> image size

n x n -> filter window size

Number of Multiplication (FFT vs Convolution)



	256	512	1024
FFT	524288	2359296	10485760
Convolution (n=3)	589824	2359296	9437184
Convolution (n=5)	1638400	6553600	26214400
Convolution (n=7)	3211264	12845056	51380224
Convolution (n=9)	5308416	21233664	84934656

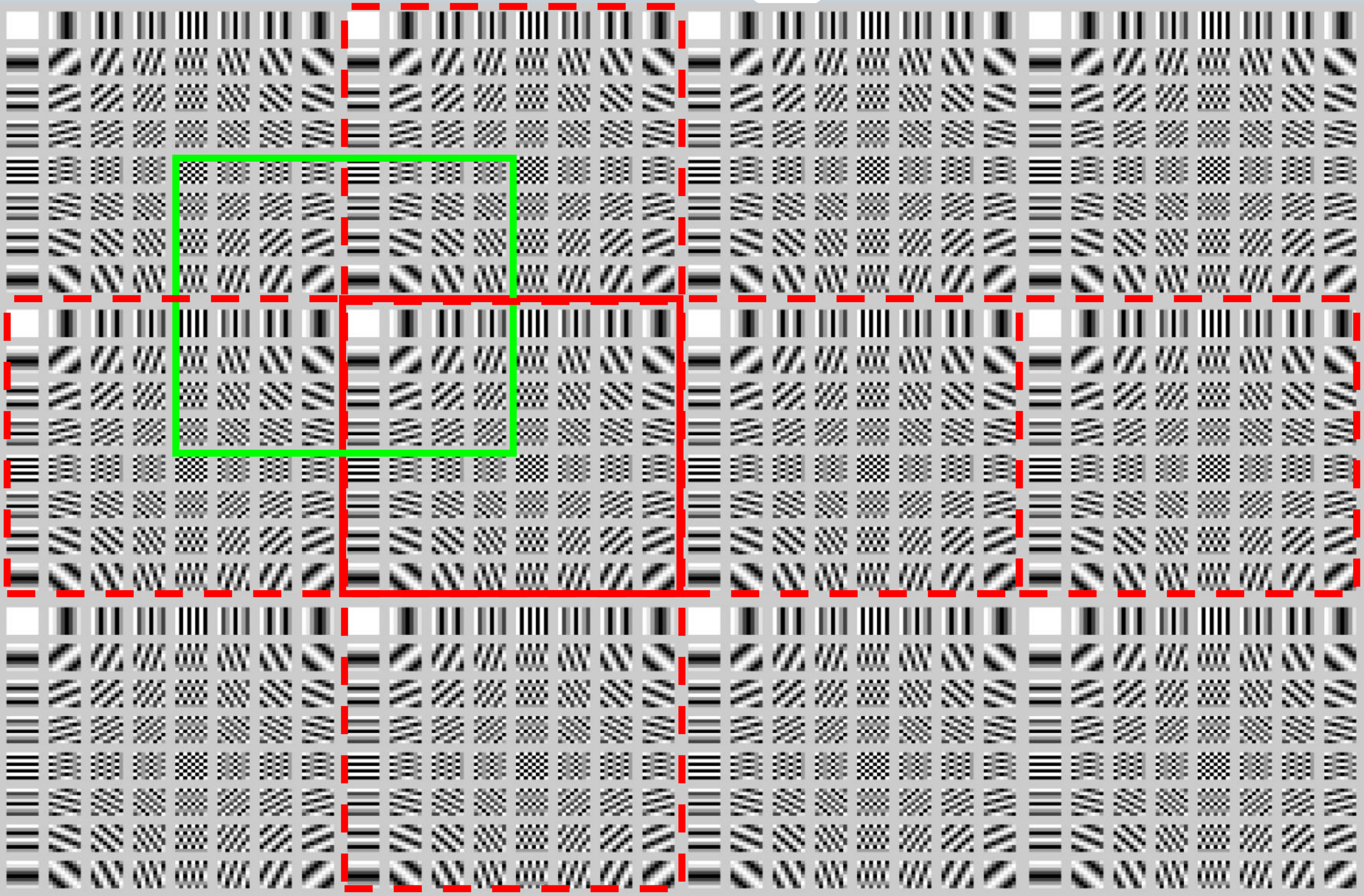
Fast Computation Through FFT

- Fast Filtering
  - เบื้องหลัง convolution ด้วย template ขนาดใหญ่กว่า 3x3
- Fast Transform
  - JPEG2000 / MPEG-4
    - Fast wavelet transform using FFT
- etc.

Fourier Transform Properties

SHIFTING PROPERTY

2D DFT Cosine Periodic Property

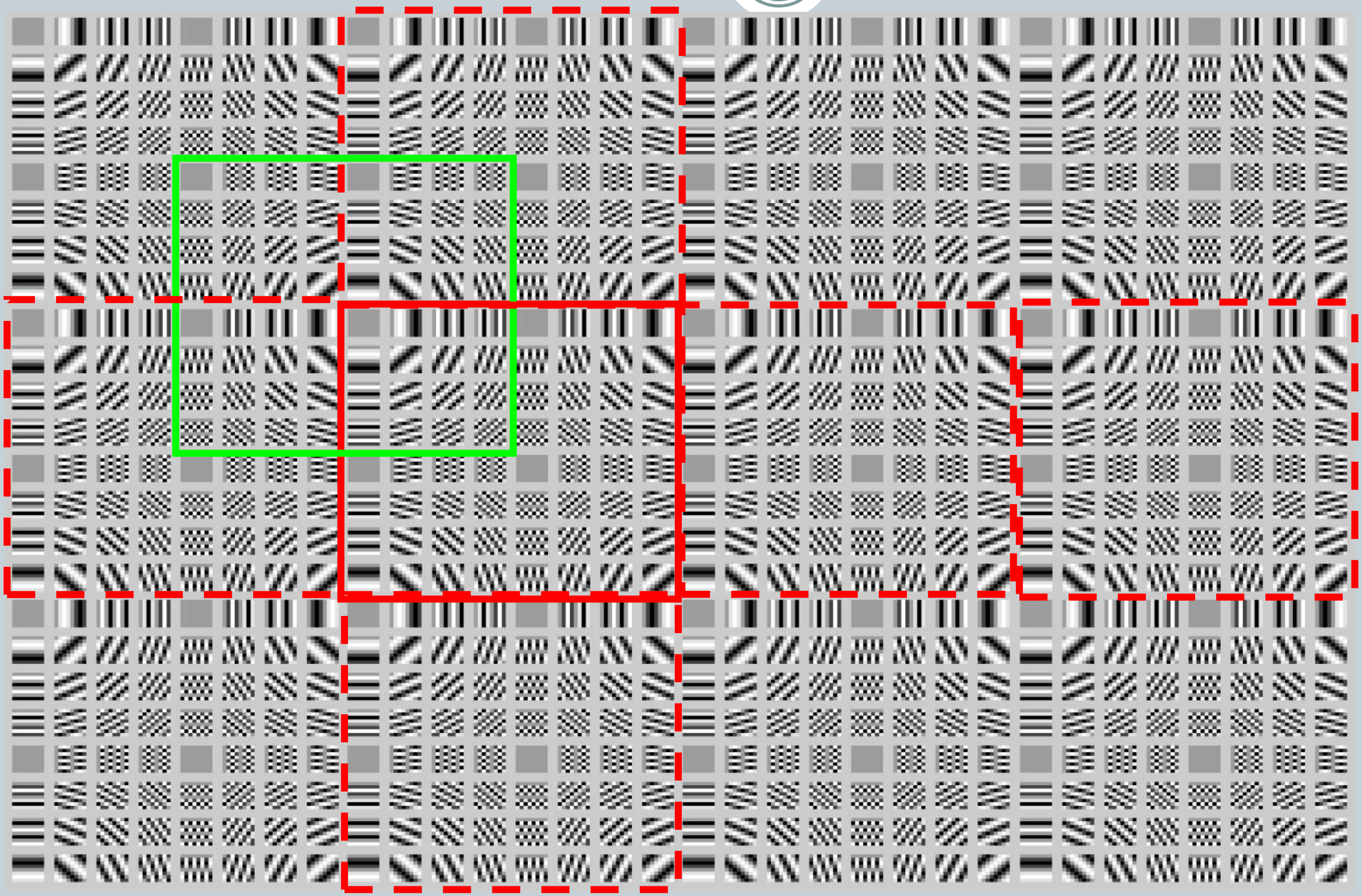


ในกรอบภาพ

ขยายออกนอกกรอบภาพ

เลื่อนตำแหน่งในการพิจารณาข้อมูล

2D DFT Shifted Sine Periodic Property

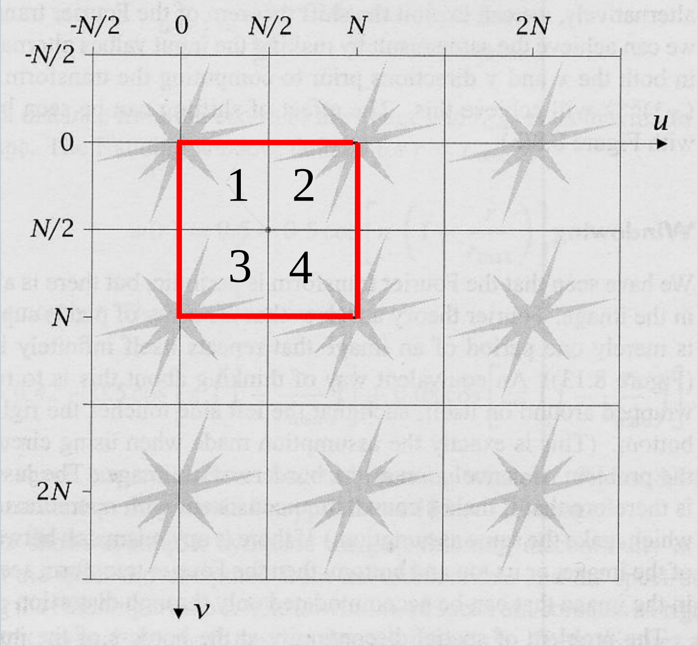


ในกรอบภาพ

ขยายออกนอกกรอบภาพ

เลื่อนตำแหน่งในการพิจารณาข้อมูล

Fourier Transform property (1)



$1 = 4^*$

$2 = 3^*$

$$F(u,v) = \left(F\left(u + \frac{N}{2}\right) \right)^* = F(-u,-v)^*$$

$$F(u,v) = F(u+M,v) = F(u,v+N) = F(u+M,v+N)$$

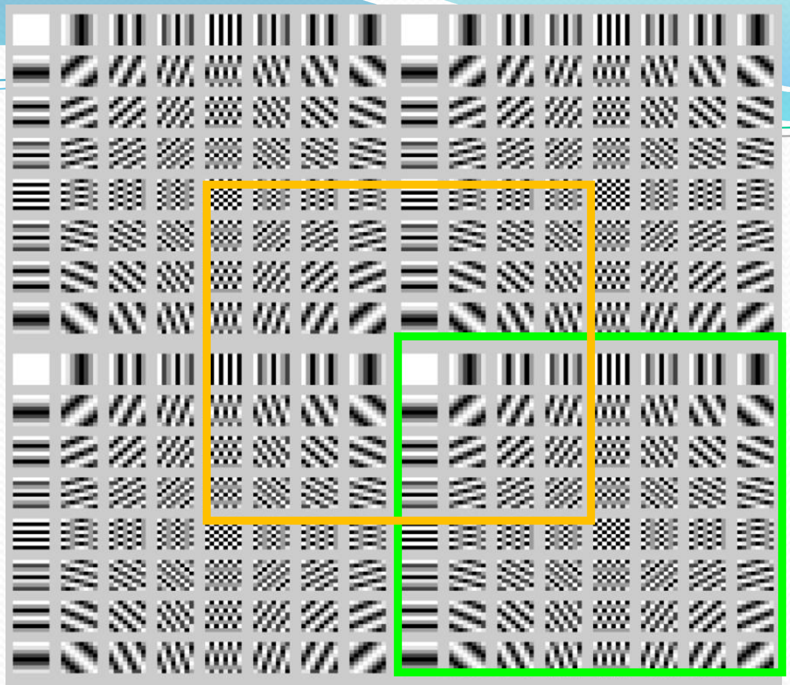
$$f(x,y)e^{-j2\pi\left(\frac{u_0 x}{M}+\frac{v_0 y}{N}\right)} \Leftrightarrow F(u-u_0, v-v_0)$$

$$f(x,y)e^{-j2\pi\left(\frac{Mx}{2M}+\frac{Ny}{2N}\right)} \Leftrightarrow F(u-M/2, v-N/2)$$

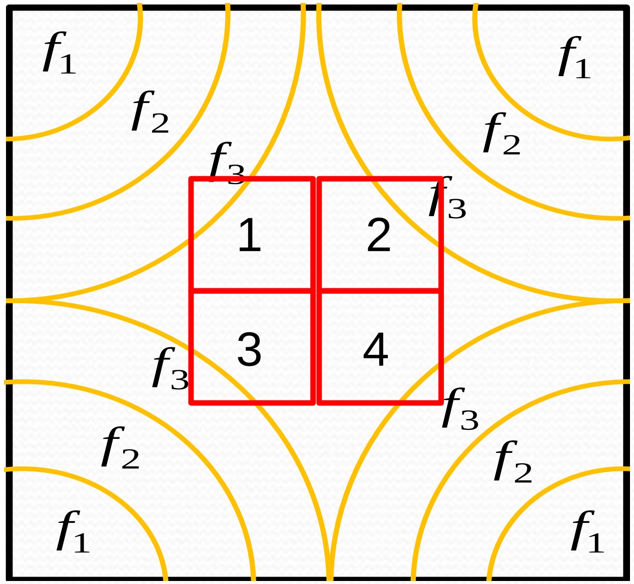
$$f(x,y)e^{-j\pi(x+y)} \Leftrightarrow F(u-M/2, v-N/2)$$

$$f(x,y)(-1)^{(x+y)} \Leftrightarrow F(u-M/2, v-N/2)$$

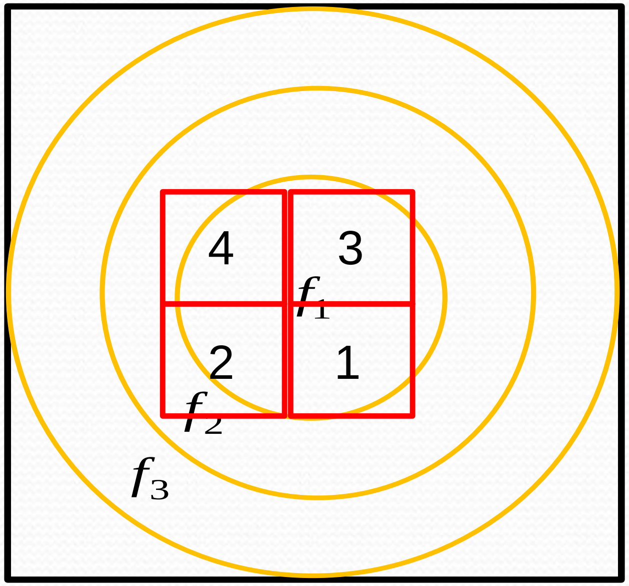
Fourier Shifting



Shifting in frequency for **Fast Programming**



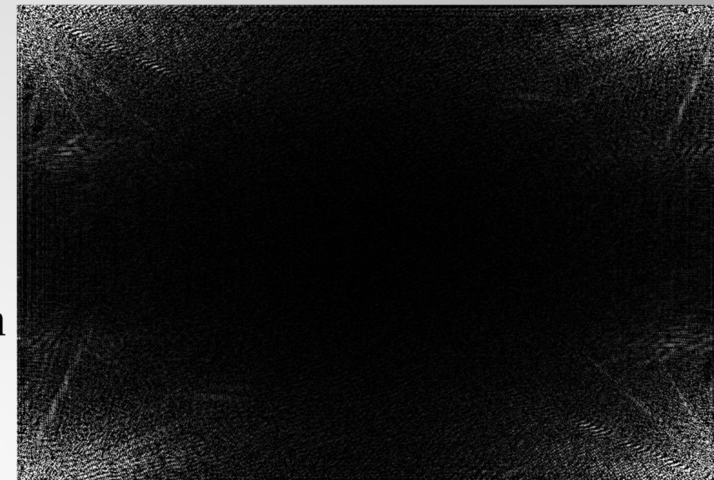
No Frequency Shifting



Frequency Shifting

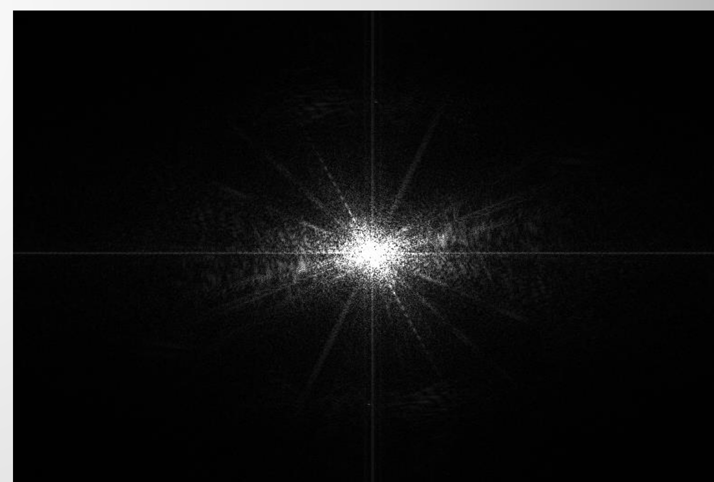
Fourier Transform property (2)

DFT
Unshifted
Magnitude Spectrum



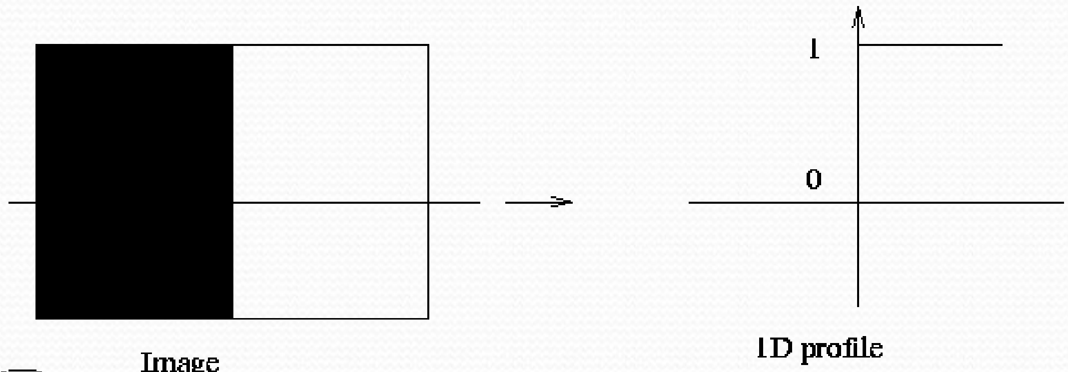
DFT
Shifted
Magnitude Spectrum

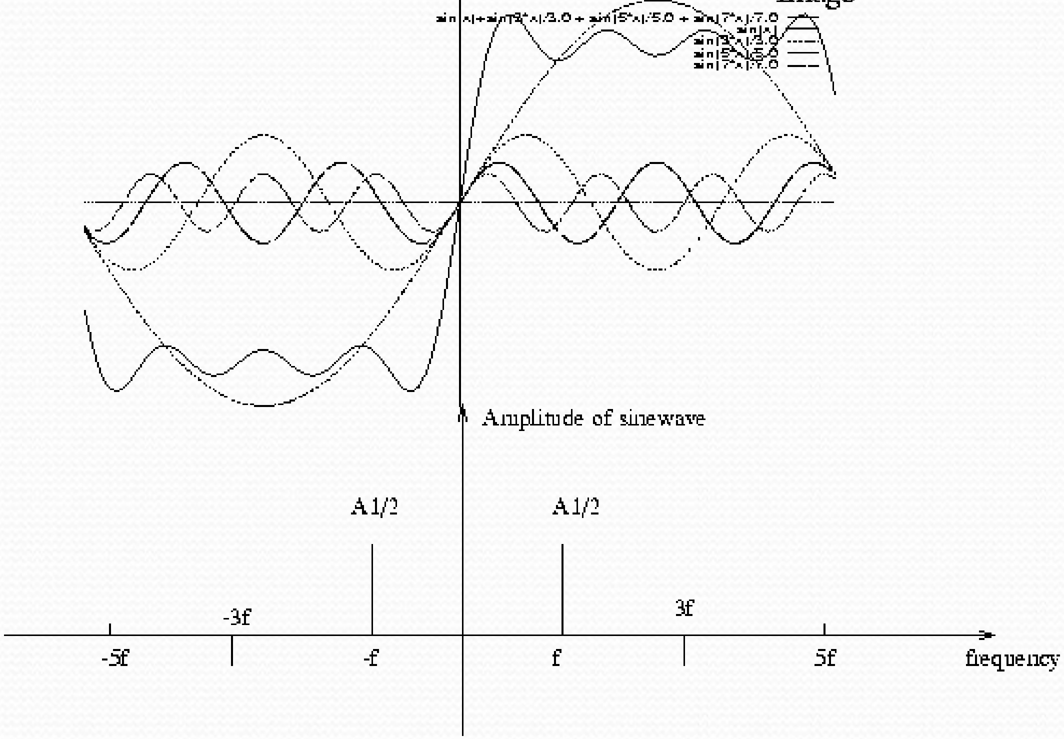
DFT $(f * (-1)^{(x+y)})$

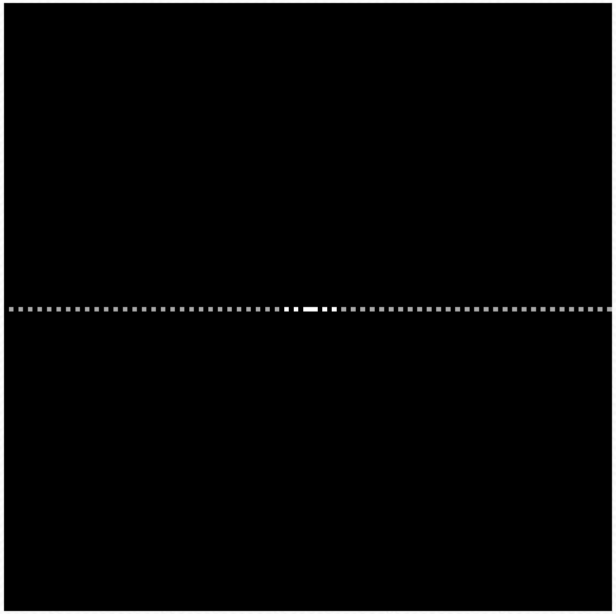


DFT Magnitude response |F(u,v)|

Unit step response







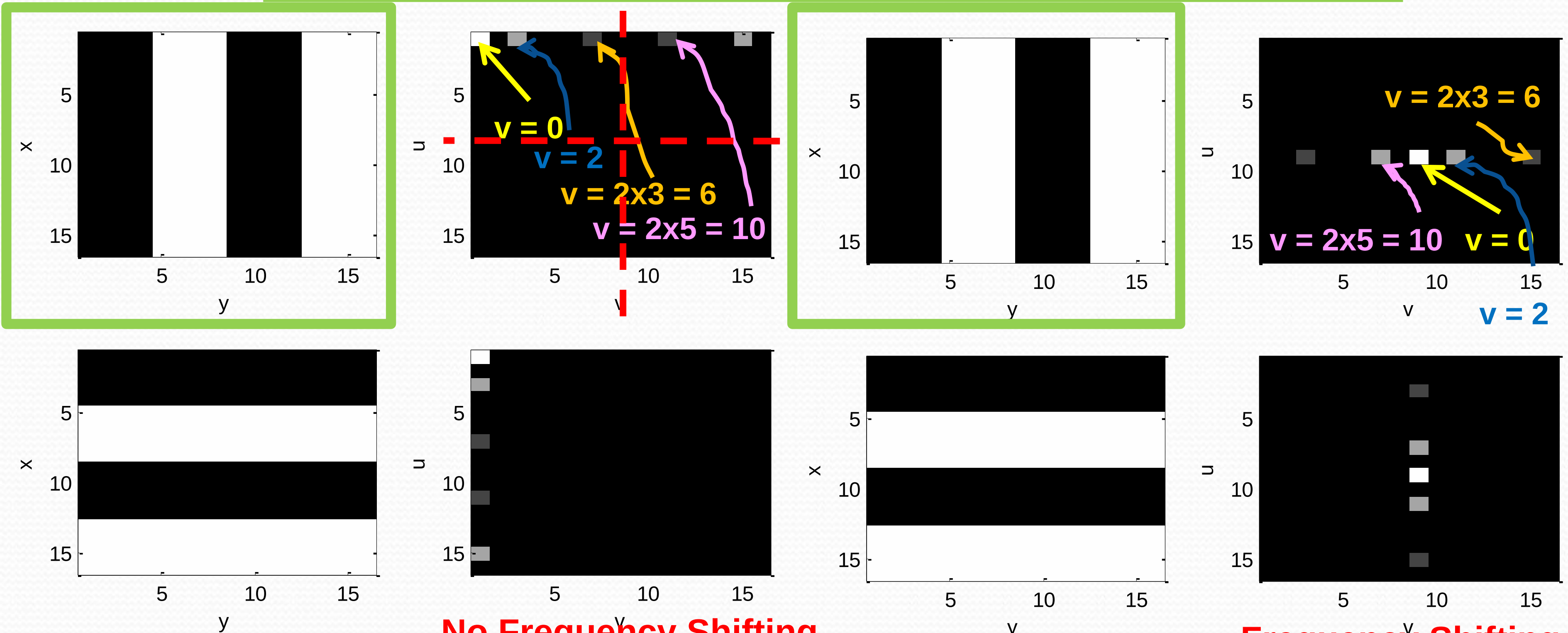
1	2
3	4

M=16; N=16

M/2=8; N/2=8

4	3
2	1

v = changing in column direction (cycle/image size) = 2



No Frequency Shifting

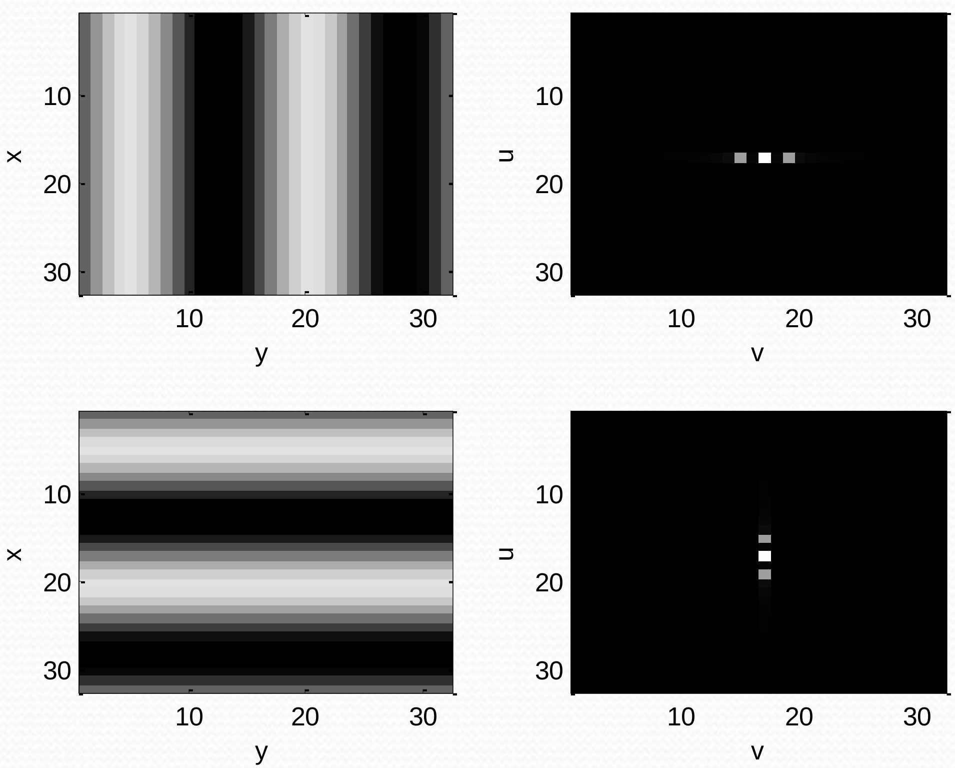
Frequency Shifting

u = changing in column direction (cycle/image size) = 2

Image of sine at v = 2

M=32; N=32

M/2=16; N/2=16



Non-zero coefficients

F(u=0,v=0)
F(u=0,v=2)
F(u=0,v = -2)

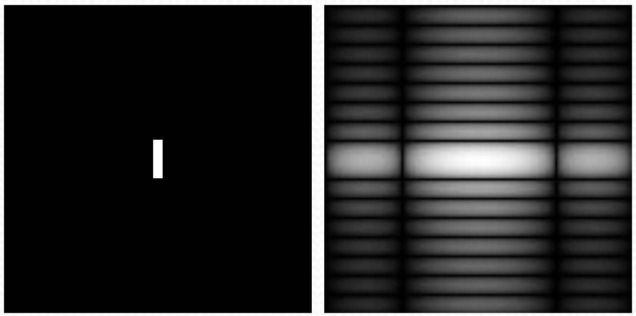
Non-zero coefficients

F(u=2,v=0)
F(u=2,v=0)
F(u = -2,v=0)

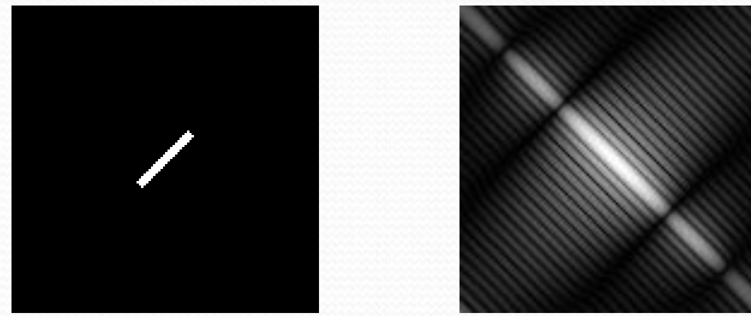
No Odd harmonic frequency

DFT Magnitude response |F(u,v)|

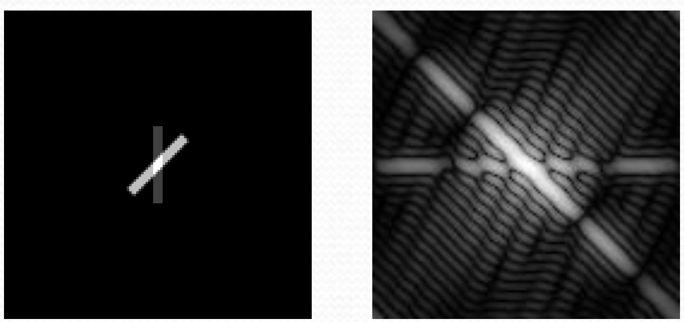
Magnitude response มีความสัมพันธ์กับทิศของการเปลี่ยนแปลงในภาพ



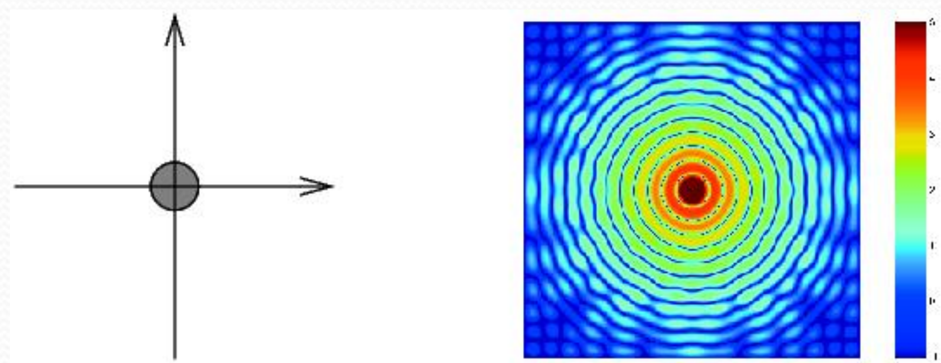
Small vertical line response



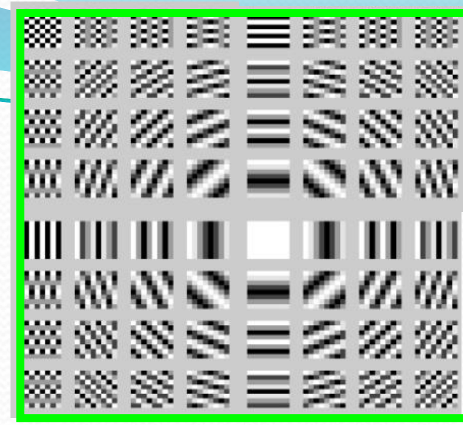
-45 degree rotated line response

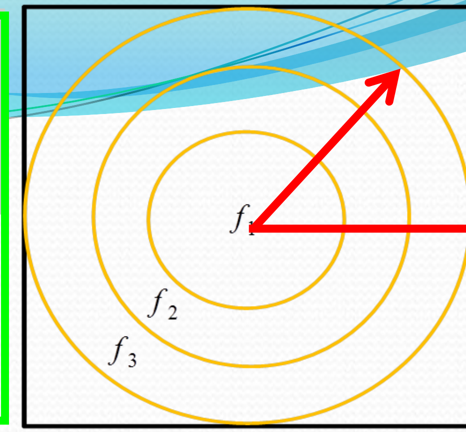


Linear Combination response

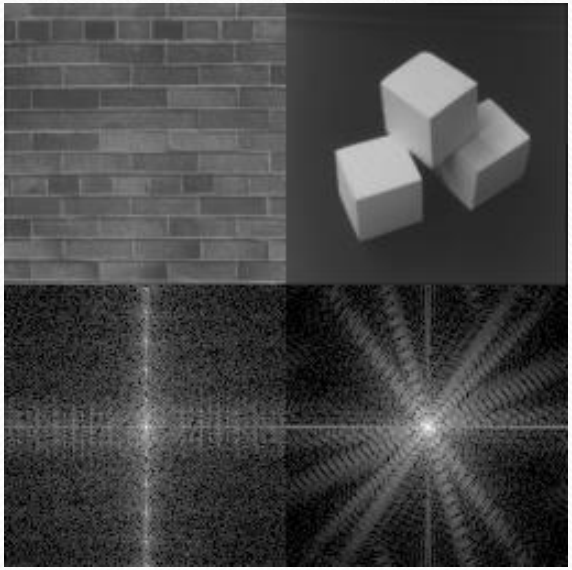


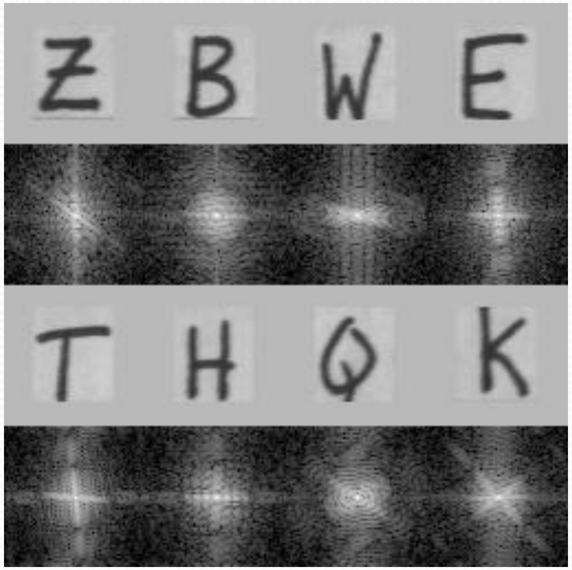
DFT Magnitude response |F(u,v)|

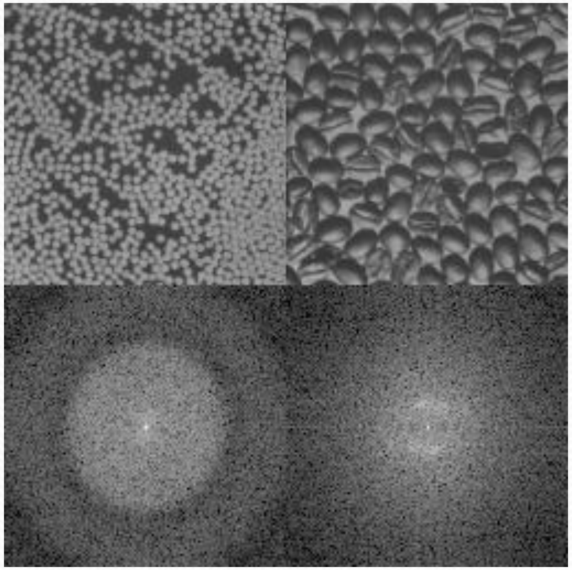




Frequency Shifting







ถ้าต้องการสร้างตัวกรองเส้นขอบและที่มีทิศทางการเปลี่ยนแปลงตามกำหนด
ต้องสร้างตัวกรองอย่างไร

	Basic Idea of Transformation	Fourier Transform	Discrete Cosine Transform	Wavelet Transform
Forward Transform	$T = f \bullet B$ $T(u) = K_u \sum_{x=0}^{N-1} f(x) B_u(x)$	$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x) e^{-j2\pi\left(\frac{ux}{N}\right)}$ $F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\left[\cos\left(\frac{2\pi ux}{N}\right) - j\sin\left(\frac{2\pi ux}{N}\right)\right]$	$DCT^{c2}(u) = \sqrt{\frac{2}{N}} K_u \sum_{x=0}^{N-1} f(x)\cos\left(\frac{(2x+1)u\pi}{2N}\right)$ $K_u = \begin{cases} \frac{1}{\sqrt{2}} & if\ u=0 \\ 1 & otherwise \end{cases}$	$DWT_S(j_0,k) = \frac{1}{\sqrt{N}}\left[2^{j_0/2}\sum_{x=0}^{N-1} f(x) S(2^{j_0}x-k)\right]$ $DWT_W(j,k) = \frac{1}{\sqrt{N}}\left[2^{j/2}\sum_{x=0}^{N-1} f(x) W(2^{j}x-k)\right]$
Inverse Transform	$f(x) = \sum_{u=0}^{N-1} K_u T(u) B_u(x)$	$f(x) = \sum_{u=0}^{N-1} F(u) e^{j2\pi\left(\frac{ux}{N}\right)}$ $f(x) = \sum_{u=0}^{N-1} F(u)\left[\cos\left(\frac{2\pi ux}{N}\right) + j\sin\left(\frac{2\pi ux}{N}\right)\right]$	$f(x) = \sqrt{\frac{2}{N}}\sum_{u=0}^{N-1} K_u DCT^{c2}(u)\cos\left(\frac{(2x+1)u\pi}{2N}\right)$	$f(x) = \frac{1}{\sqrt{N}}\sum_{k} DWT_S(j_0,k) 2^{j_0/2} S_{j_0,k}(2^{j_0}x-k) +$ $\frac{1}{\sqrt{N}}\sum_{j=j_0}^{\infty}\sum_{k} DWT_W(j,k) 2^{j/2} W_{j,k}(2^{j}x-k)$
Transform Constant	K_u	$\frac{1}{N} = \sqrt{\frac{1}{N}}\sqrt{\frac{1}{N}}$	$\sqrt{\frac{2}{N}}$ $\quad K_u = \begin{cases} \frac{1}{\sqrt{2}} & if\ u=0 \\ 1 & otherwise \end{cases}$	$\frac{2^{j/2}}{\sqrt{N}} = \sqrt{\frac{2^{j}}{N}}$
Representing continuity signal		Best	Better	Good
Representing discontinuity signal		Not good	Not good but better	Best
Energy Compaction (Vanishing moment)		Not good	Good	Better depending on chosen basis function