UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FAKULTAS ILMU KOMPUTER

# Basic Image Processing 2 – Frequency Domain

CSCE604133 Computer Vision

Fakultas Ilmu Komputer

Universitas Indonesia

# Acknowledgements


- These slides are created with reference to:
  - Computer Vision: Algorithms and Applications, 2nd ed., Richard Szeliski https://szeliski.org/Book/
  - Digital Image Processing, Gonzales and Woods, 3rd ed, 2008.
  - Course slides for CSCE604133 Image Processing – Faculty of Computer Science, Universitas Indonesia
  - Introduction to Computer Vision, Cornell Tech https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html
  - Computer Vision, University of Washington https://courses.cs.washington.edu/courses/cse576/08sp/

# Basis for Images

# Review: Basis in a vector space

- If $V$ is a vector space and $S = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_r\}$ is a set of vectors in $V$, then $S$ is a **basis** for $V$ if the following properties hold:
  - $S$ is linearly independent
  - $S$ spans $V$

- Examples of vector spaces:
  - Euclidean space ($R^2, R^3$ etc)
  - Functions in $R^2$
  - Matrices of size $m$ by $n$

# Examples of Basis

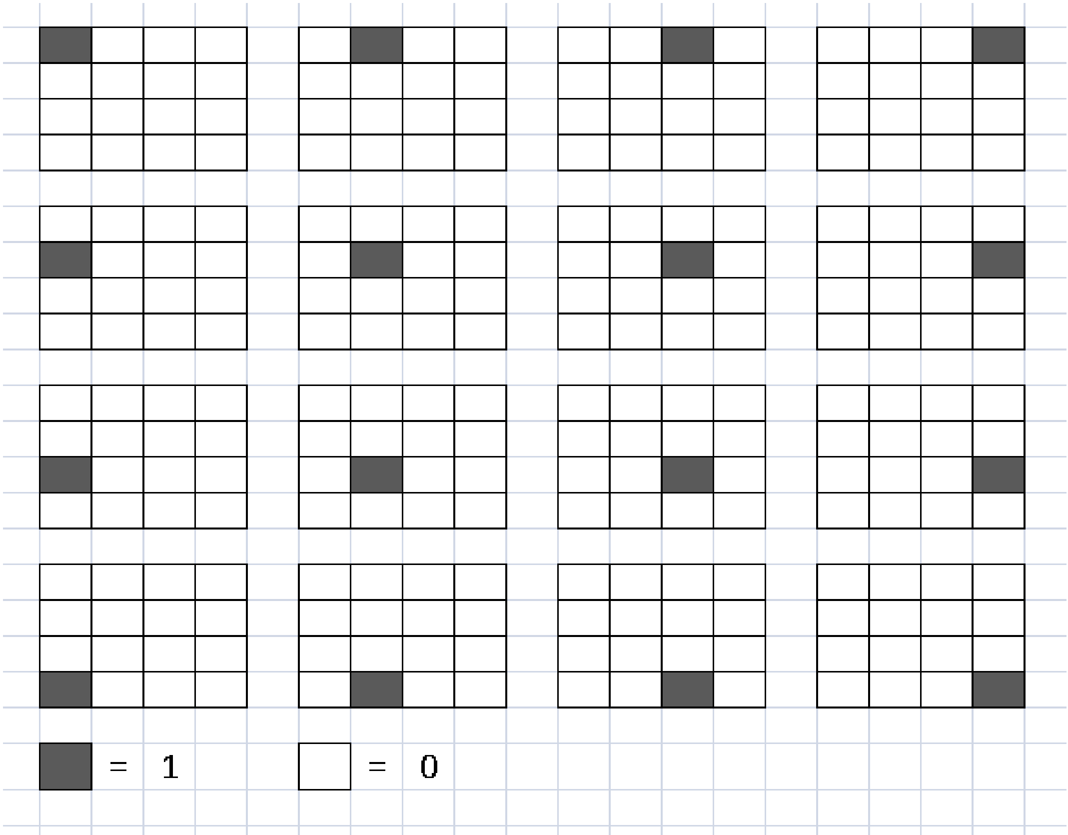
- $\left\{\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix}\right\}$ or $\left\{\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}1\\1\end{bmatrix}\right\}$ may serve as basis for $R^2$

- $\left\{\begin{bmatrix}1 & 0\\0 & 0\end{bmatrix},\begin{bmatrix}0 & 1\\0 & 0\end{bmatrix},\begin{bmatrix}0 & 0\\1 & 0\end{bmatrix},\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}\right\}$ may serve as basis for $M_{2\times 2}$

# Basis for 4x4 matrices?

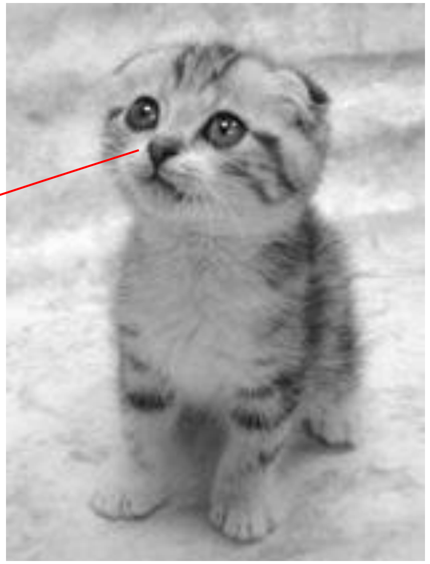
4x4, kalau ditambahin semua jadi 4x4

Simplest one:



# How about images ?

- Remember that we can always consider **images** as **matrices**

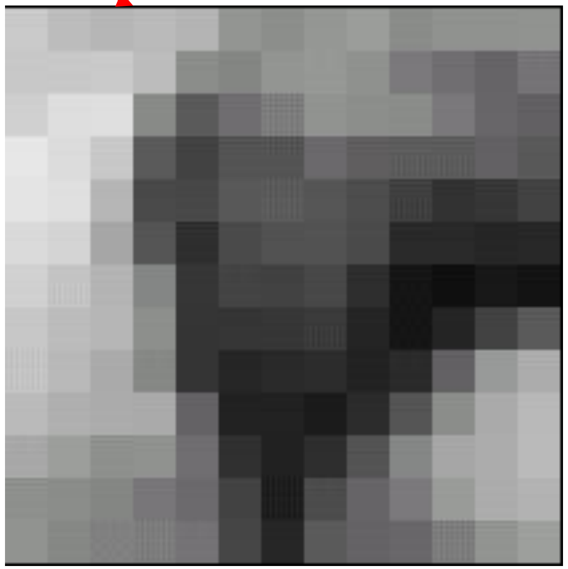


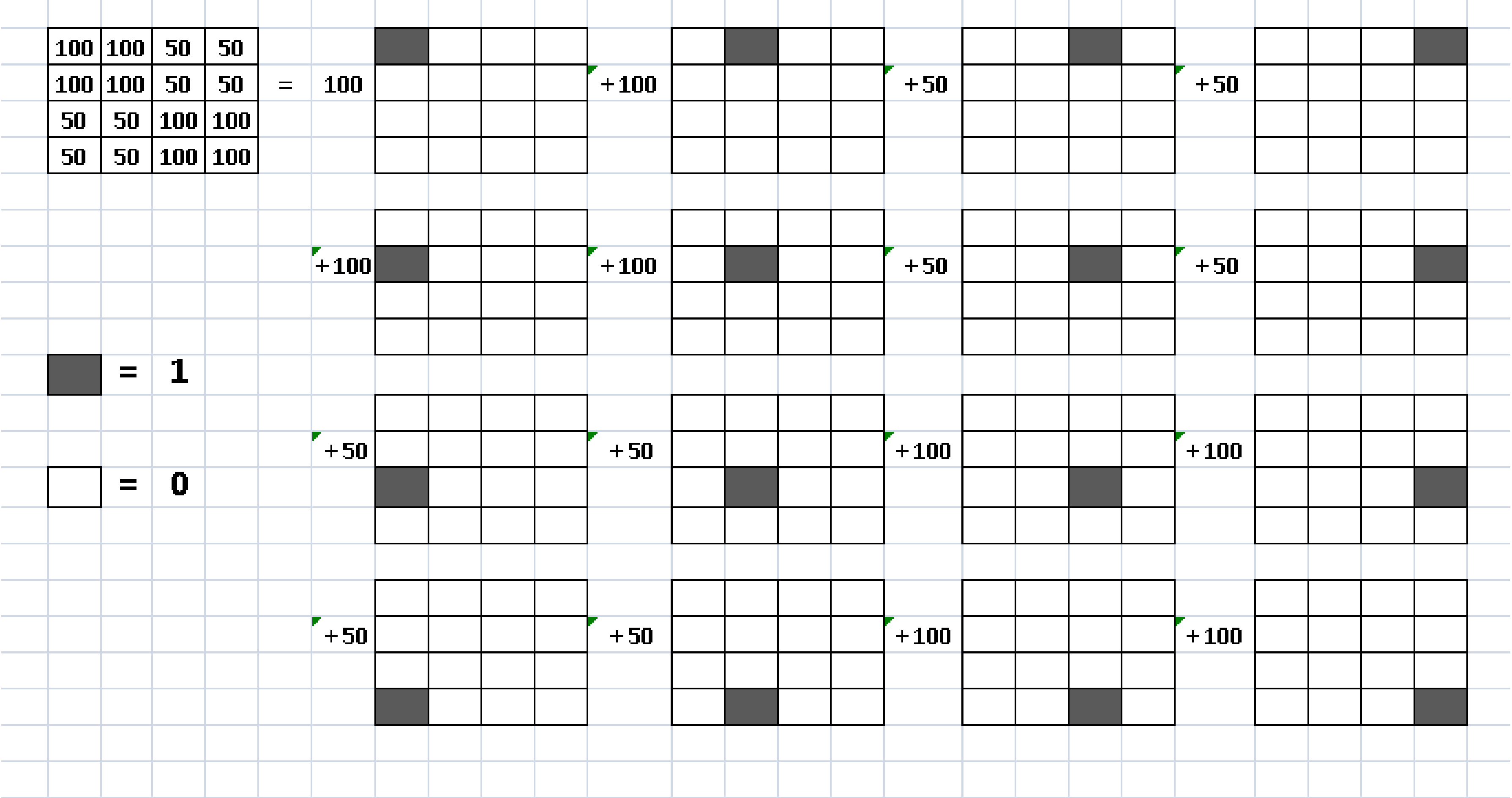
artinya, kita bisa menjabarkan basis dari suatu ruang vektor.

Semua citra 4x4, kita bisa jabarkan sebagai kombinasi linear dari basis-basis 4x4.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 201 | 188 | 181 | 185 | 180 | 147 | 140 | 149 | 155 | 138 | 144 | 144 | 145 |
| 199 | 200 | 201 | 188 | 139 | 132 | 147 | 150 | 143 | 123 | 112 | 102 | 117 |
| 207 | 221 | 222 | 136 | 90 | 111 | 125 | 145 | 140 | 138 | 122 | 104 | 97 |
| 231 | 219 | 200 | 90 | 65 | 84 | 84 | 107 | 95 | 92 | 92 | 99 | 89 |
| 227 | 223 | 181 | 74 | 72 | 89 | 92 | 86 | 77 | 63 | 50 | 55 | 65 |
| 217 | 211 | 166 | 85 | 47 | 75 | 82 | 83 | 75 | 42 | 42 | 39 | 40 |
| 208 | 195 | 179 | 131 | 54 | 68 | 66 | 72 | 46 | 21 | 15 | 24 | 19 |
| 198 | 187 | 181 | 141 | 53 | 54 | 55 | 59 | 37 | 21 | 37 | 66 | 90 |
| 195 | 184 | 170 | 134 | 52 | 38 | 42 | 45 | 35 | 43 | 98 | 152 | 172 |
| 186 | 175 | 171 | 169 | 100 | 34 | 34 | 27 | 44 | 85 | 139 | 170 | 184 |
| 167 | 156 | 142 | 144 | 112 | 48 | 32 | 46 | 84 | 133 | 166 | 172 | 186 |
| 142 | 139 | 131 | 120 | 108 | 67 | 30 | 76 | 102 | 123 | 153 | 171 | 178 |
| 145 | 134 | 128 | 125 | 117 | 70 | 38 | 91 | 101 | 105 | 125 | 146 | 157 |

=



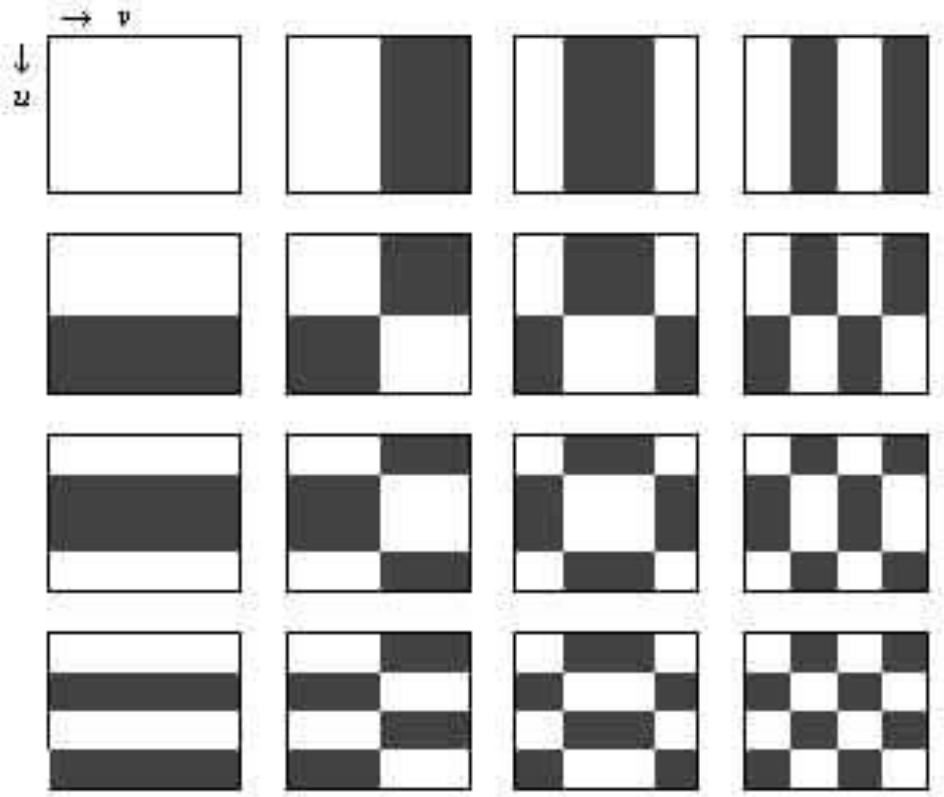



- Consider that we have a 4x4 image, and we want to use the basis shown above. In this case, we have **16** nonzero coefficient.

# Why Transformation?

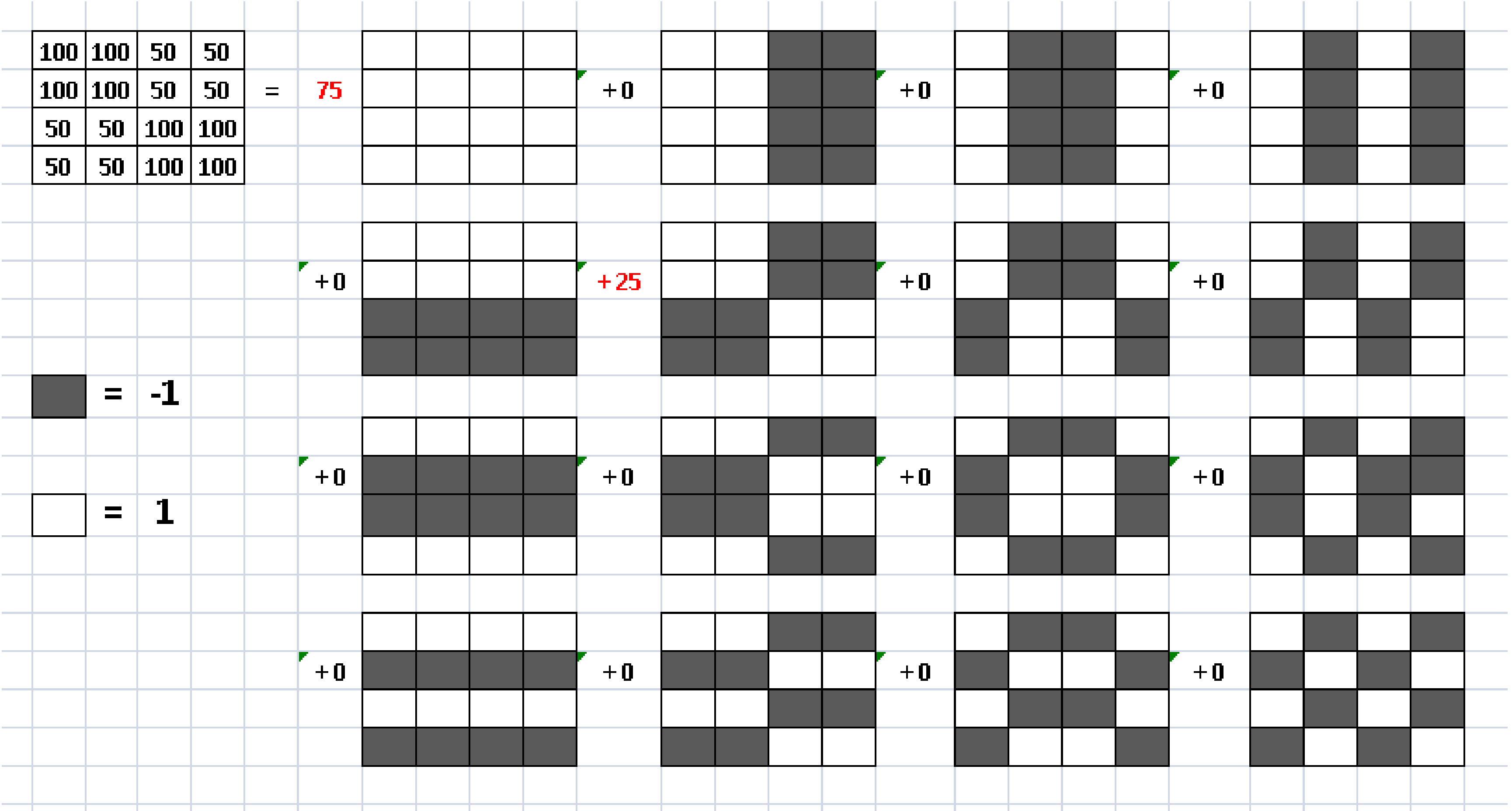
- Transformation can be used to simplify the solution of a problem [Brigham,1974]
  - $y = e^x$ can be solved as $\ln y = x \ln e = x$
- Finding a specific information to solve a problem
  - To get smaller amount of data to be stored
  - To get object features
  - To remove noise
  - Etc.

# Other basis for 4x4 matrices (Hadamard)




Ini adalah basis untuk 4x4

White =  1, Black = - 1




- How many nonzero coefficients?
- Only 2 !
  → minimum storage
  → basics of image compression (discards zeroes or near-zero coefficients)

Menggunakan matriks hadamard sebagai basis

Banyak koefisien yang 0

- In this example, we have **transformed** the image from

| | | | |
|---|---|---|---|
| 100 | 100 | 50 | 50 |
| 100 | 100 | 50 | 50 |
| 50 | 50 | 100 | 100 |
| 50 | 50 | 100 | 100 |

to

| | | | |
|---|---|---|---|
| 75 | 0 | 0 | 0 |
| 0 | 25 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

→ Easier to see the dominant **pattern** (or frequency) relative to the basis

# Hadamard Transform (in general)

- 2-D Hadamard Transform:

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

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

- $b_{\,n}(z)$ is the $n^{th}$ bit of binary $z$ representation.
- Example : $n = 3, z = 6\ (110)$ maka $b_0(z) = 0, b_1(z) = 1, b_2(z) = 1$

# Example of Hadamard Transform

- The basis function of the Hadamard transform (also of the Walsh transform) is the orthogonal column and row
- To get the transformation, multiply the basis function with the input image (white equal to positive and black equal to negative). Each position in H(u,v) using only 1 block of basis function.
  - The coefficient we get (300 and 100) is somewhat different from our previous example (75 and 25) because in the original formula, the divider is 4 (not 16)

| | | | |
|---|---|---|---|
| 100 | 100 | 50 | 50 |
| 100 | 100 | 50 | 50 |
| 50 | 50 | 100 | 100 |
| 50 | 50 | 100 | 100 |

| | | | |
|---|---|---|---|
| 300 | 0 | 0 | 0 |
| 0 | 100 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

$$\mathrm{H}(0,0) = (100+100+50+50+100+100+50+50+50+50+100+100+50+50+100+100)/4$$

$$= 1200/4 = 300$$

$$\mathrm{H}(0,1) = (100+100-50-50+100+100-50-50+50+50-100-100+50+50-100-100)/4 = 0$$

$$\mathrm{H}(0,2) = (100-100-50+50+100-100-50+50+50-50-100+100+50-50-100+100)/4 = 0$$

$$\mathrm{H}(0,3) = (100-100+50-50+100-100+50-50+50-50+100-100+50-50+100-100)/4 = 0$$

# Example of Hadamard Transform (2)

$H(1,0) = (\dots\dots)/4 = 0$
$H(1,1) = (\dots\dots)/4 = 400/4 = 100$
$H(1,2) = (\dots\dots)/4 = 0$
$H(1,3) = (\dots\dots)/4 = 0$

$H(2,0) = (\dots\dots)/4 = 0$
$H(2,1) = (\dots\dots)/4 = 0$
$H(2,2) = (\dots\dots)/4 = 0$
$H(2,3) = (\dots\dots)/4 = 0$

$H(3,0) = (\dots\dots)/4 = 0$
$H(3,1) = (\dots\dots)/4 = 0$
$H(3,2) = (\dots\dots)/4 = 0$
$H(3,3) = (\dots\dots)/4 = 0$

- Only a small part of the transformed image have large values, the other parts are zeroes.
- We only need to store the values which are not zeroes, it means that the size of our image representation becomes very small, so we can compress the image

| | | | |
|---|---|---|---|
| 300 | 0 | 0 | 0 |
| 0 | 100 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

# Example of Inverse Hadamard Transform

- From the transformed image we could get the original image by using the basis, for each position f(x,y) using all blocks at related position of (x,y).
- The result of the image reconstruction is the original image.

$$f(0,0) = (300 + 0 + 0 + 0 + 0 + 100 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0)/4 = 400/4 = 100$$
$$f(0,1) = (300 \dots\dots + 100 \dots\dots.)/4 = 400/4 = 100$$
$$f(0,2) = (300 \dots\dots - 100 \dots\dots.)/4 = 200/4 = 50$$
$$f(0,3) = (300 \dots\dots - 100 \dots\dots.)/4 = 200/4 = 50$$
$$f(1,0) = (300 \dots\dots + 100 \dots\dots.)/4 = 400/4 = 100$$
$$f(1,1) = (300 \dots\dots + 100 \dots\dots.)/4 = 400/4 = 100$$
$$f(1,2) = (300 \dots\dots - 100 \dots\dots.)/4 = 200/4 = 50$$
$$f(1,3) = (300 \dots\dots - 100 \dots\dots.)/4 = 200/4 = 50$$
$$f(2,0) = (300 \dots\dots - 100 \dots\dots.)/4 = 200/4 = 50$$
$$f(2,1) = (300 \dots\dots - 100 \dots\dots.)/4 = 200/4 = 50$$
$$f(2,2) = (300 \dots\dots + 100 \dots\dots.)/4 = 400/4 = 100$$
$$f(2,3) = (300 \dots\dots + 100 \dots\dots.)/4 = 400/4 = 100$$
$$f(3,0) = (300 \dots\dots - 100 \dots\dots.)/4 = 200/4 = 50$$
$$f(3,1) = (300 \dots\dots - 100 \dots\dots.)/4 = 200/4 = 50$$
$$f(3,2) = (300 \dots\dots + 100 \dots\dots.)/4 = 400/4 = 100$$
$$f(3,3) = (300 \dots\dots + 100 \dots\dots.)/4 = 400/4 = 100$$

# Example: DCT Basis

- DCT Basis used for jpg compression
- For JPEG 2000, the basis used is Wavelet (see next slides)

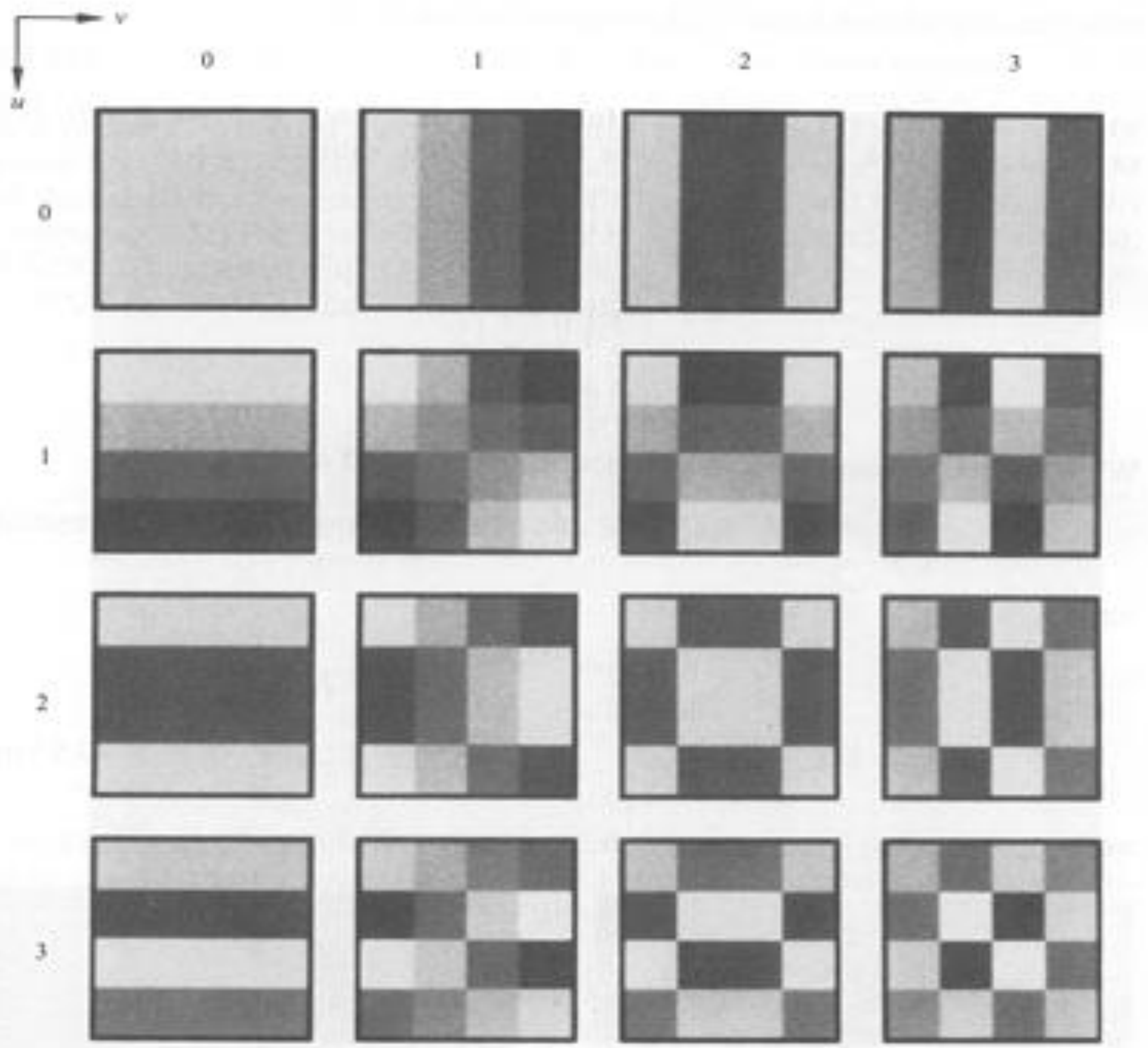



*Figure 3.28* Discrete cosine transform basis functions for $N = 4$. Each block consists of $4 \times 4$ elements, corresponding to $x$ and $y$ varying from 0 to 3. The origin of each block is at its top left. The highest value is shown in white. Other values are shown in grays, with darker

# Example: DCT Basis (2)

- Format bmp: 158 KB
- Format jpg: 132 KB

Ada sekian byte yang koefinsien 0.

Ada basis yang gak perlu diperhitungkan jadi untuk mengutuhkan gambar.

# Walsh Transform

- 2-D Walsh transformation:

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

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

- $b_n(z)$ is the $n^{th}$ bit of binary $z$ representation.
- Example : $n = 3, z = 6, b_n(z) = 110$ then $b_0(z) = 0, b_1(z) = 1, b_2(z) = 1$

# Walsh Transform (2)

Prinsip sama:

Akan mengubah intensitas citra untuk menjadi basis linear yang dapat merepresentasikan citra tersebut.

Hammad DCT, Walsh, Intinya semua basis merepresentasikan basis berbeda.

- In visually representation, for N = 4, The Walsh basis function can be illlustrated as follows.
- The formula for forward and inverse transform is the same, so that the basis function can be used for both forward and inverse transform.

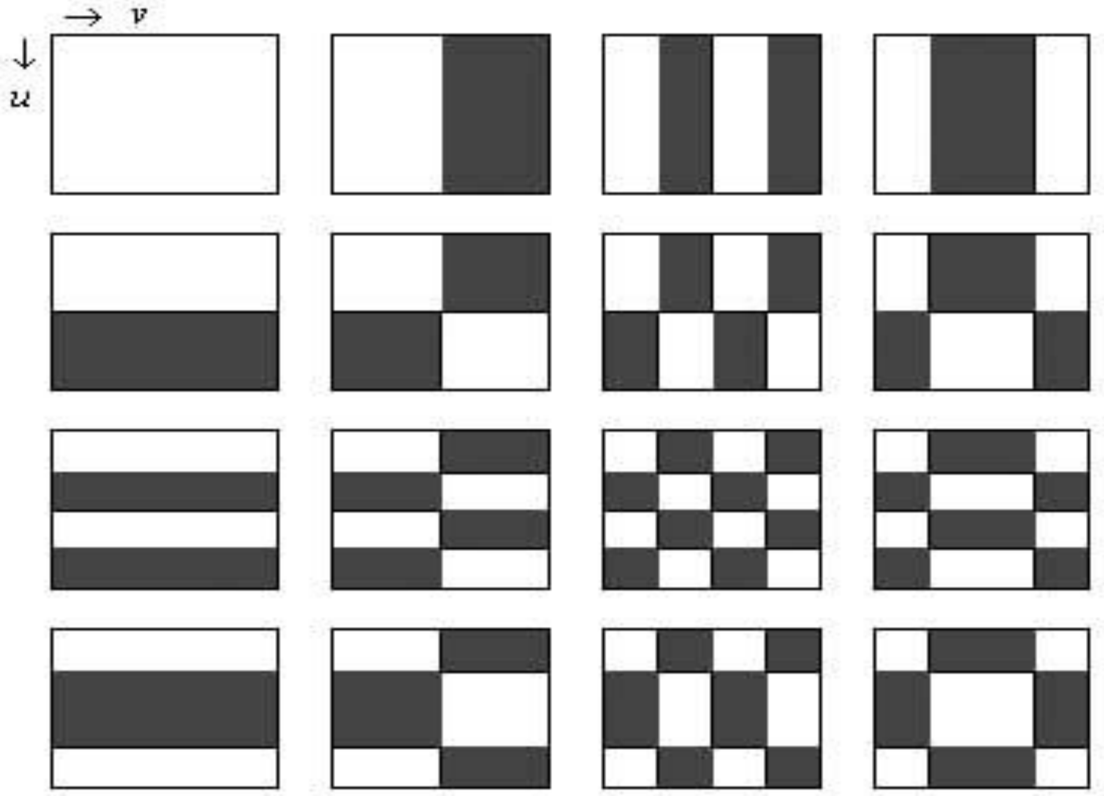



Figure 3.25 Walsh basis function for N = 4. Each block consist of 4x4 elements, corresponding to x and y varying from 0 to 3. The origin of each block is at its top left. White and black denote +1 and -1 respectively. (Gonzalez, 1993)

# Wavelet Transform

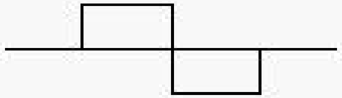
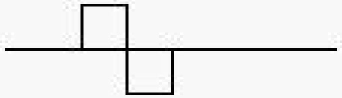
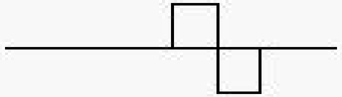
- Wavelet is originally come from a scaling function. From a scaling function we can make a mother wavelet.  Other wavelets can be found using scale operation, dilation operation, dilation and shifting of the mother wavelet.
- Scaling function → mother wavelet → dilated mother wavelet, dilated and shifted mother wavelet.

Mau banyak citra dengan 0 coefficient yang banyak, tapi ada beberapa koefisien yang kompleks

Oleh karena itu kita bisa menggunakan wavelet transform

# Basis Wavelet Haar

- Scaling function and wavelets form a new basis.

| | | |
|---|---|---|
| Scaling Function |  | $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ |
| Mother wavelet |  | $\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$ |
| Mother wavelet yang didilasikan |  | $\begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}$ |
| Mother wavelet yang didilasikan dan digeser |  | $\begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix}$ |

# 2-D Wavelet Transform

- 2-D wavelet transform is done to the row and the column which consists of the following division.
  - LL representation, also known as approximation
  - LH representation, horizontal detail
  - HL representation, vertical detail
  - HH representation, diagonal detail

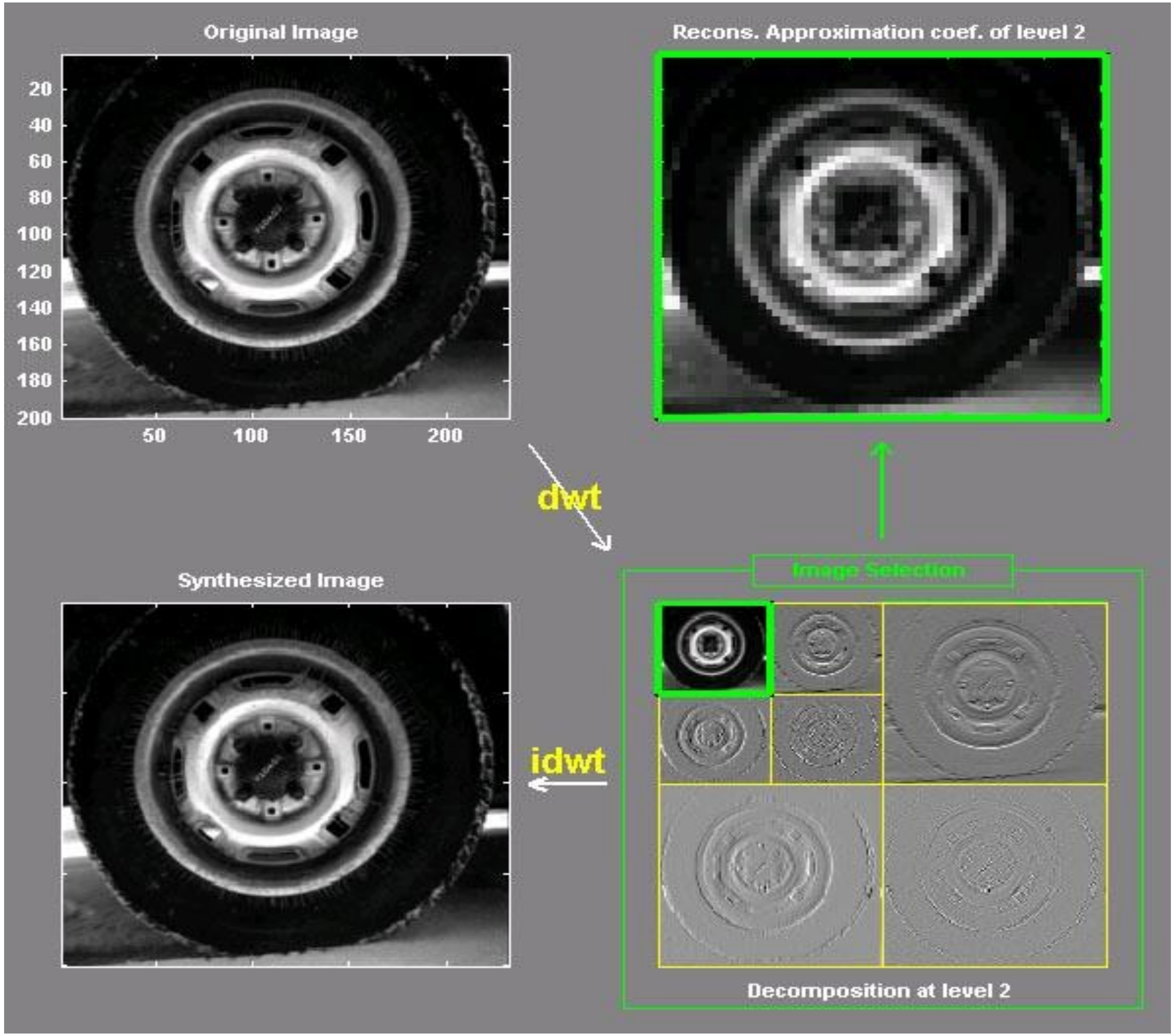
| | |
|---|---|
| LL | LH |
| HL | HH |

# 2-D Wavelet Transform

- 2 level decomposition of Haar wavelet transform using Wavelet Toolbox in Matlab

  Membagi semua informasi di dalam representasi citra, menjadi suatu wavelength

  Sehingga bisa merpresentasikan gambar  untuk melihat perubahan pola

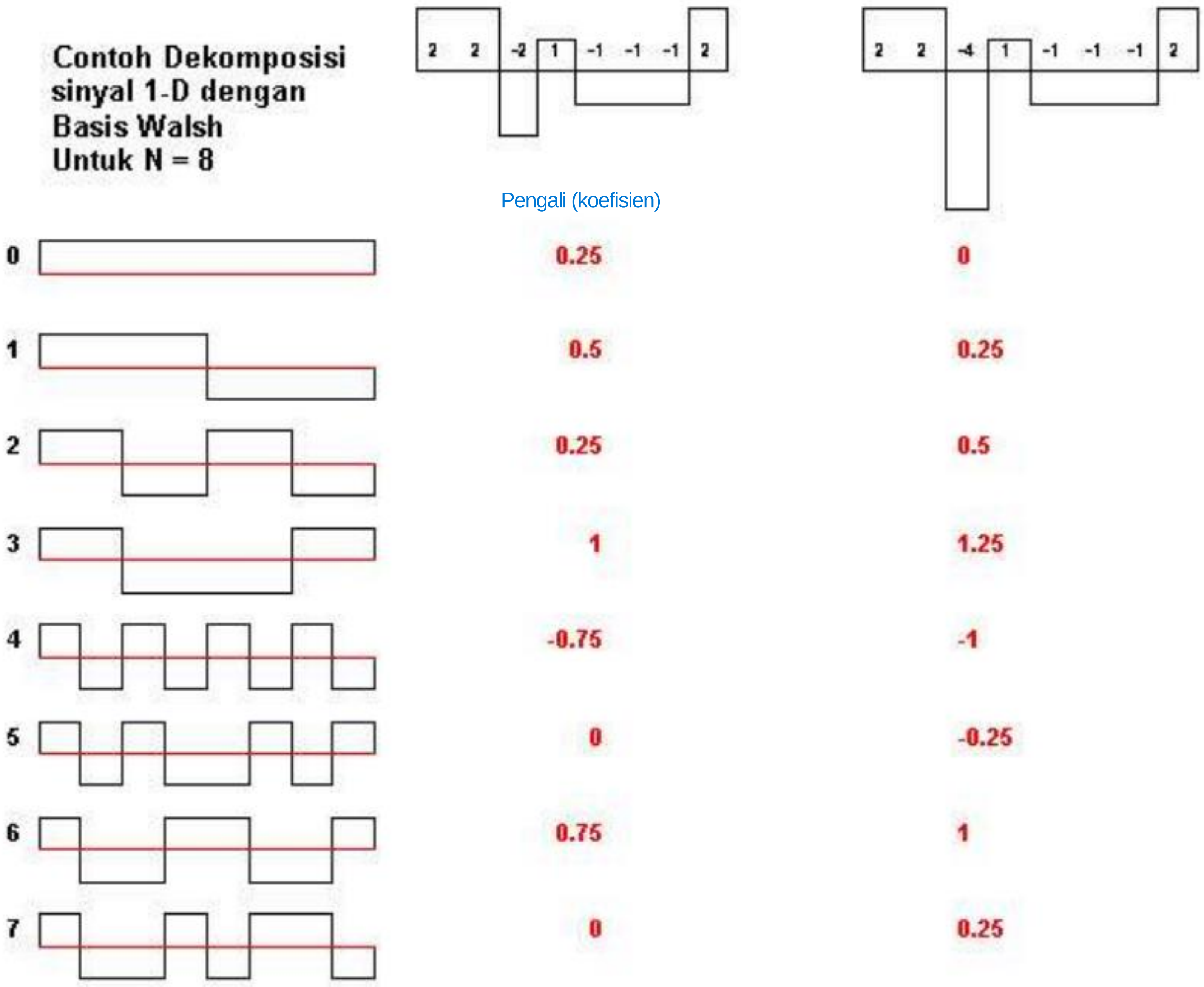


# Wavelet vs Walsh (1)

ekstrapolasi, kalau misal kita punya gambar wajah orang, dan 1 lagi gak punya wajah orang yang senyum

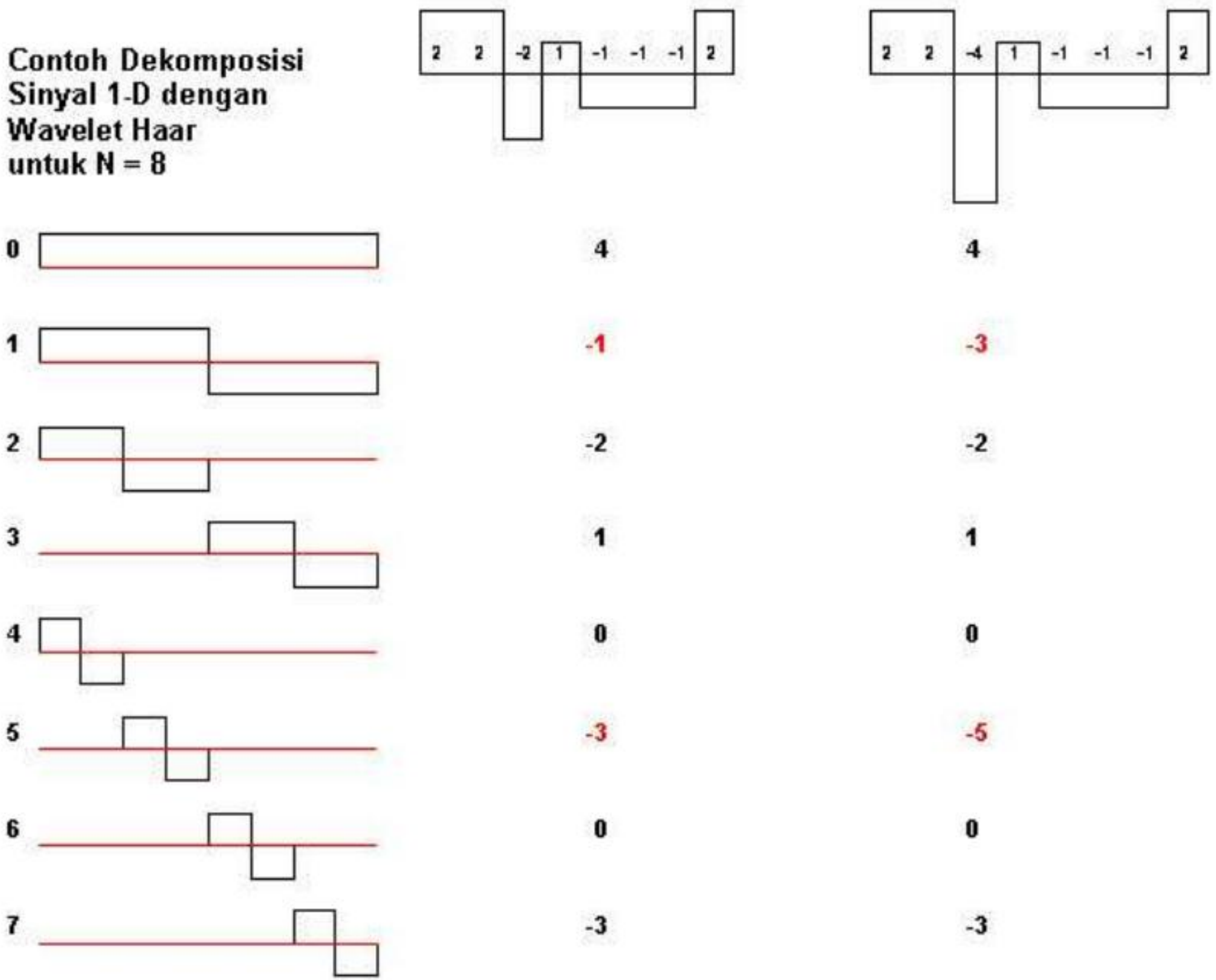
Representasi gambar wajah orang vs gambar wajah senyum perbedaannya sangat signifikan.

Wajah orang senyum dan diam, encode nya berbeda



# Wavelet vs Walsh (2

Small changes in the input signal only change the output slightly (compared to Walsh)



UNIVERSITAS INDONESIA
FAKULTAS ILMU KOMPUTER

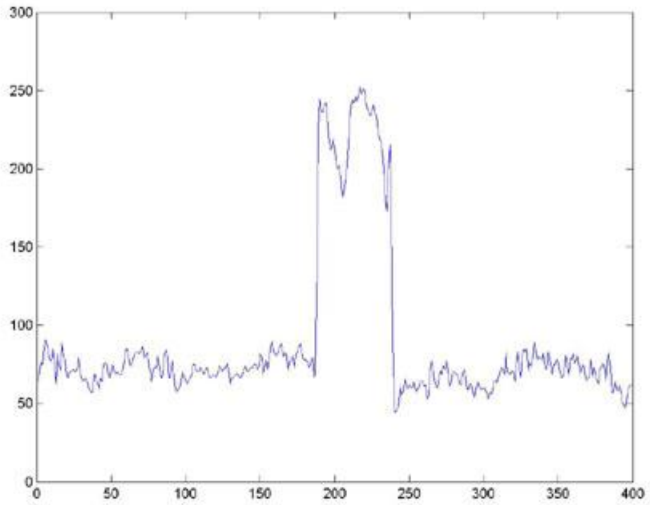
# Fourier Transform

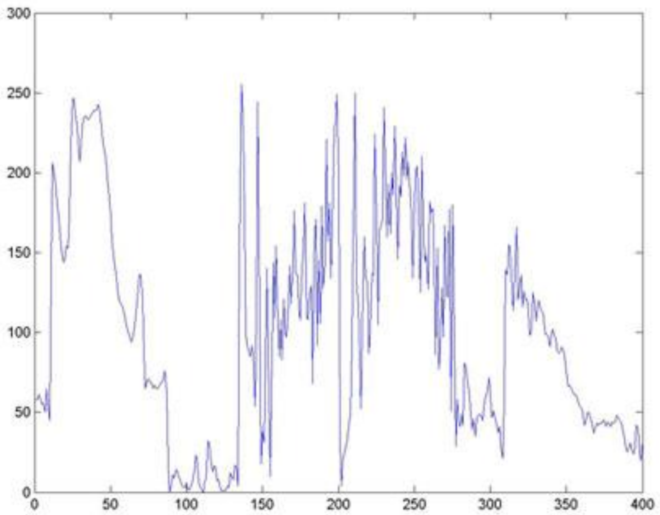
# How do we see "frequency" in images?

The plotted grayscale can be seen as a "function".

Semua pola di gambar bisa di buat menjadi periodic function

Setiap periodic function tersebut dapat dikalian lagi dengan cosine dan sin.



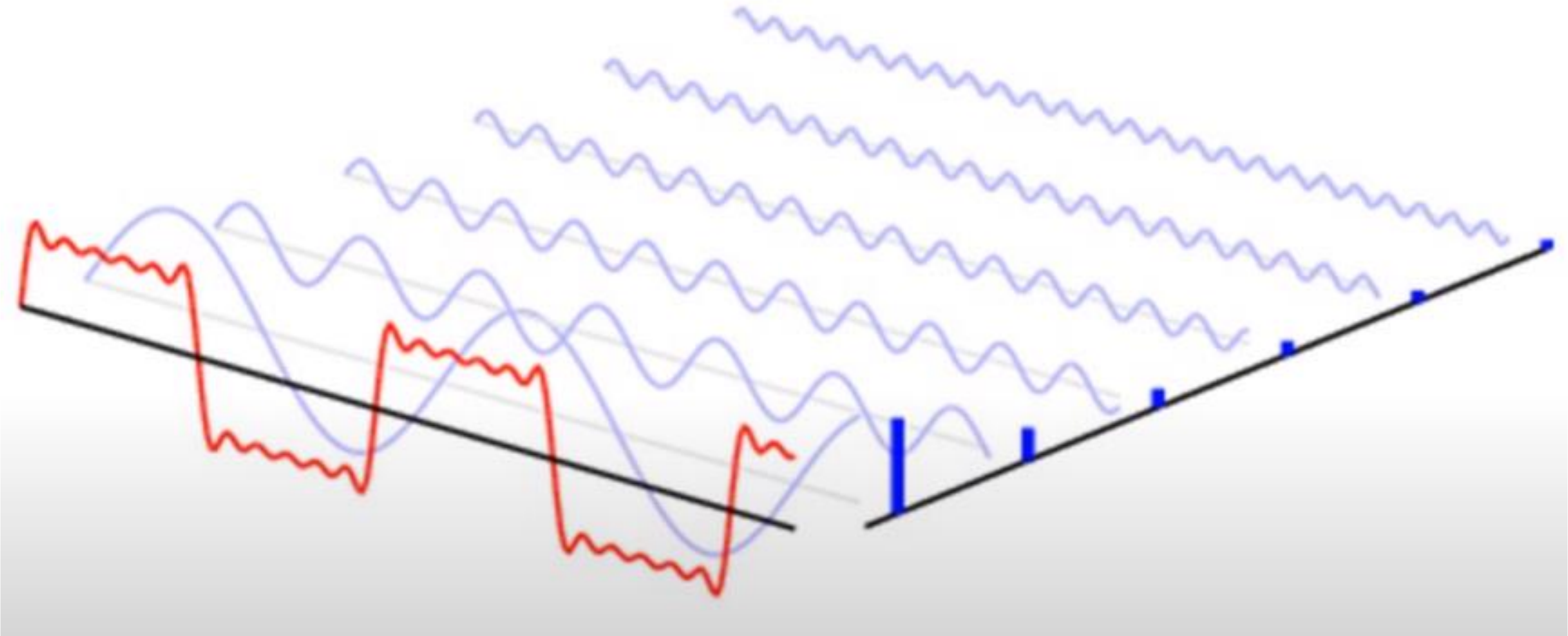



# Fourier Transform

- Jean Baptiste Joseph Fourier (1768-1830)
  - French mathematician
- Research on the *Fourier series*
  - Developed into the *Fourier transform*
- “*Théorie analytique de la chaleur* (The Analytical Theory of Heat)”
  - Fourier series for heat transfer and vibrations
  - Greenhouse effect

# Fourier Series

*"Any periodic function can be denoted as a combination of many sine and/or cosine functions"*

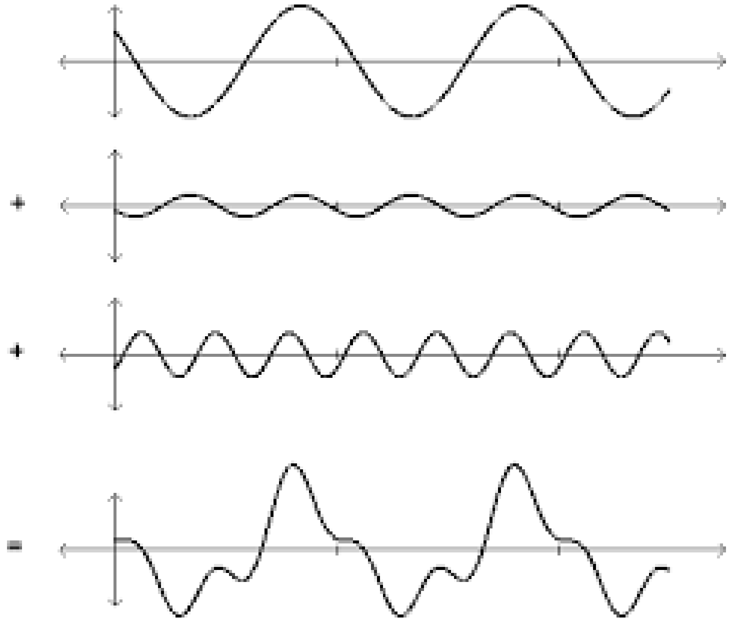


Basis:
sine/cosine
functions

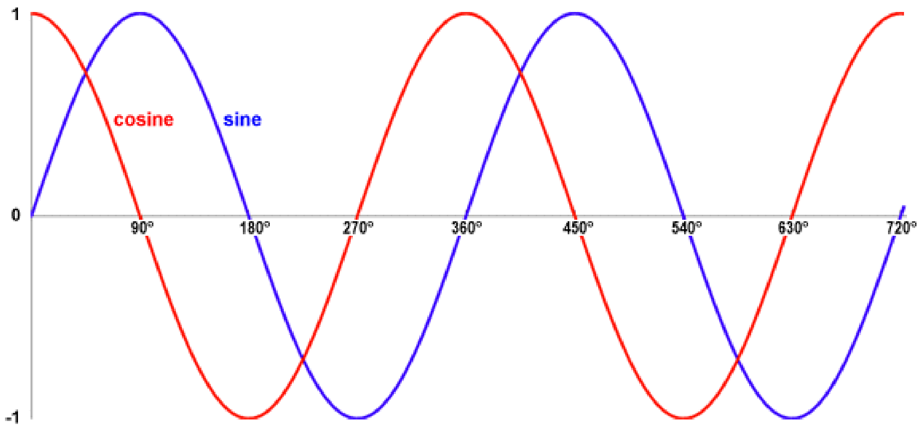
# Fourier Series (2)

*"Any periodic function can be denoted as a combination of many sine and/or cosine functions"*

- Periodic functions are functions that repeat on certain intervals.




- Sine / cosine functions



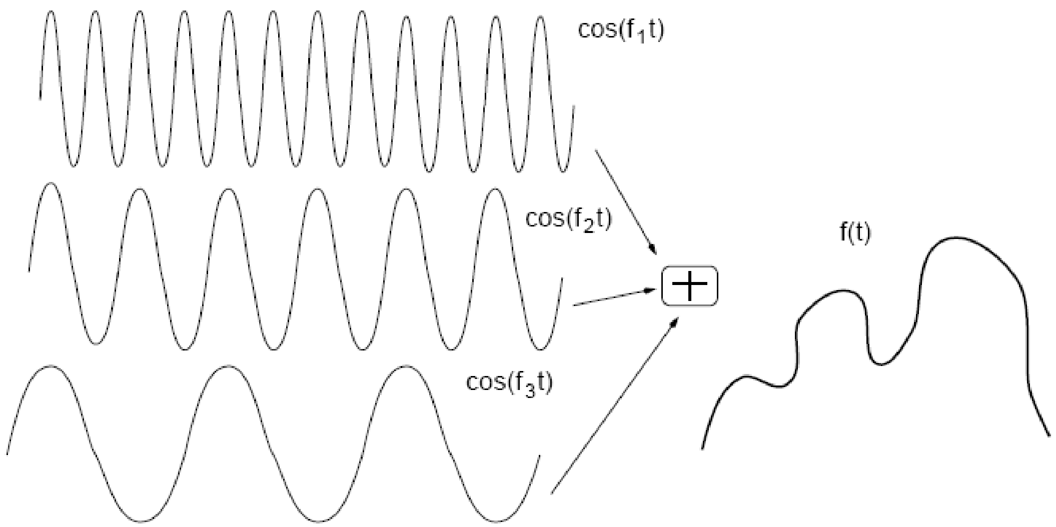
# Fourier Series (3)

- For any function $f(t)$ on a continuous variable $t$ that is periodic with a period $T$.

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{T}t}$$

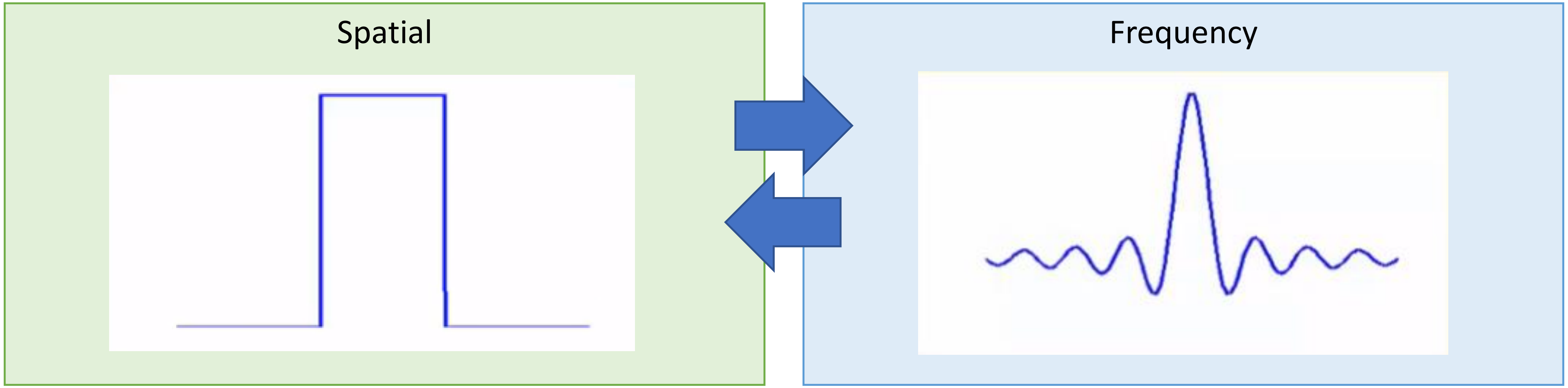
- $c_n$ is the weighting coefficient for every sine/cosine function, where:

$$c_n = \frac{1}{\mathrm{T}} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\, e^{-j\frac{2\pi n}{T}t} dt, \qquad for\ n = 0, \pm 1, \pm 2, \ldots$$



# Fourier Transform

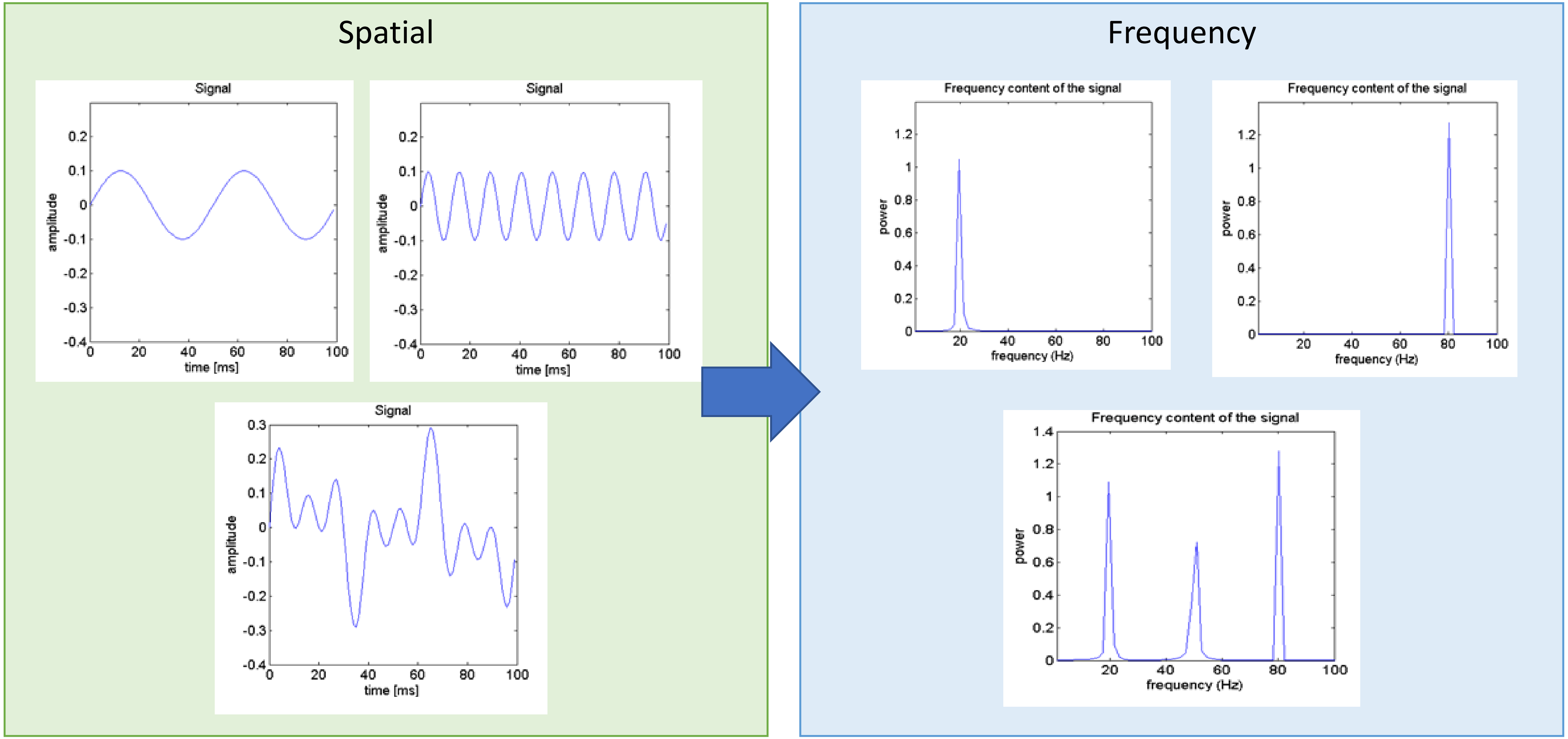
- Transforms images from the **time** domain to the **frequency** domain
  - The information is still there, but in a different form



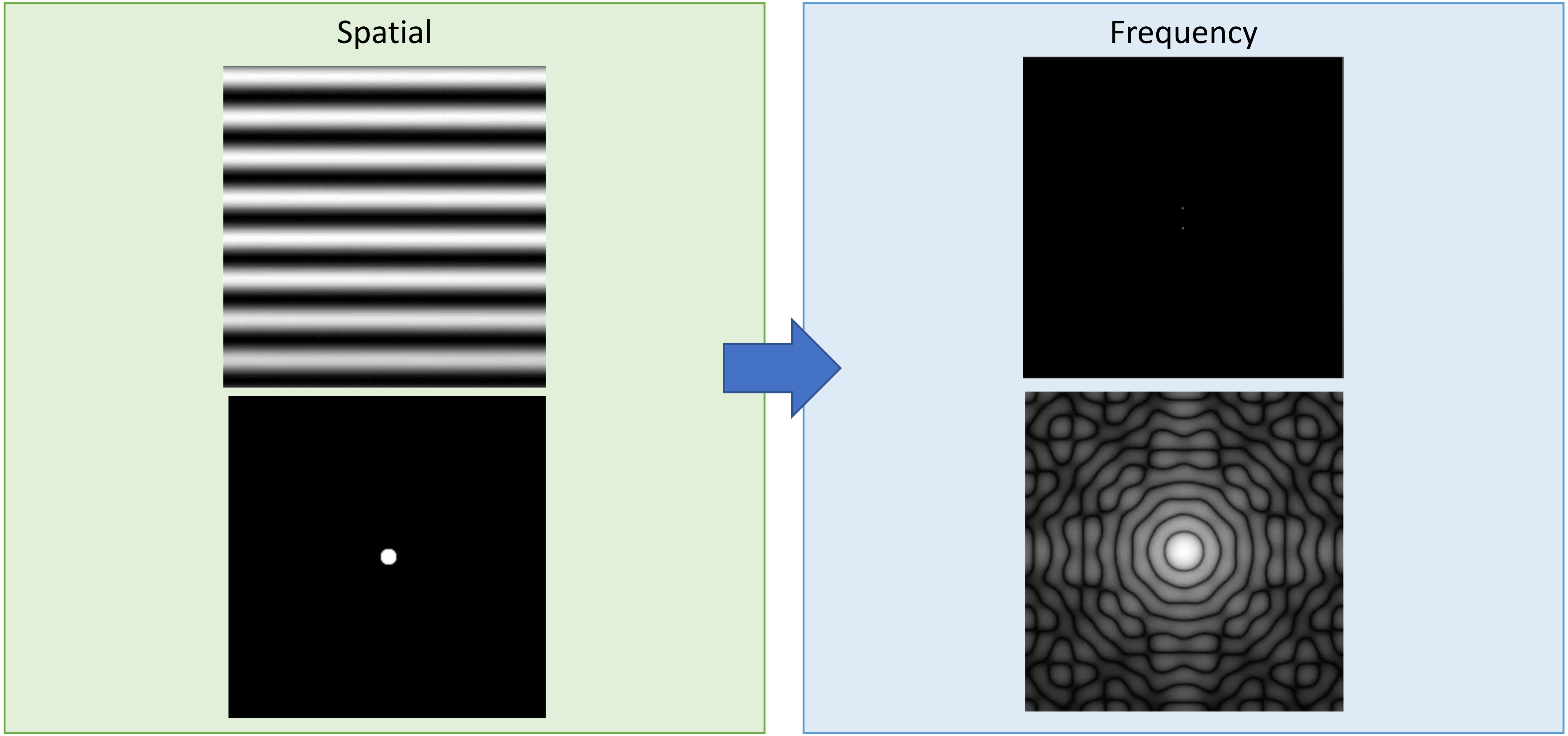

- This transform is invertible: the Fourier transformation can be returned to its original form using an inverse function

# Time to Frequency Domain

Semua perubahan sekompleks di spatial apa pun bisa direpresentasikan di frequency

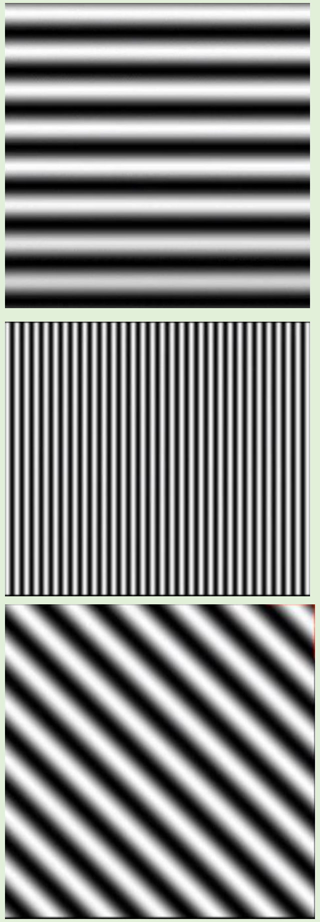


# Image to Frequency



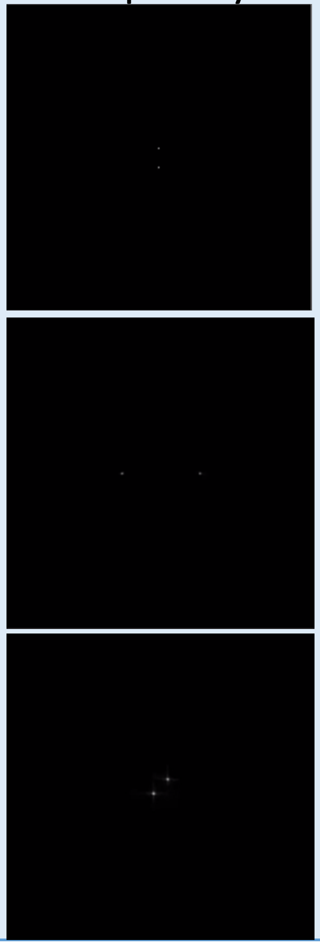
# Image to Frequency (2)

Spatial



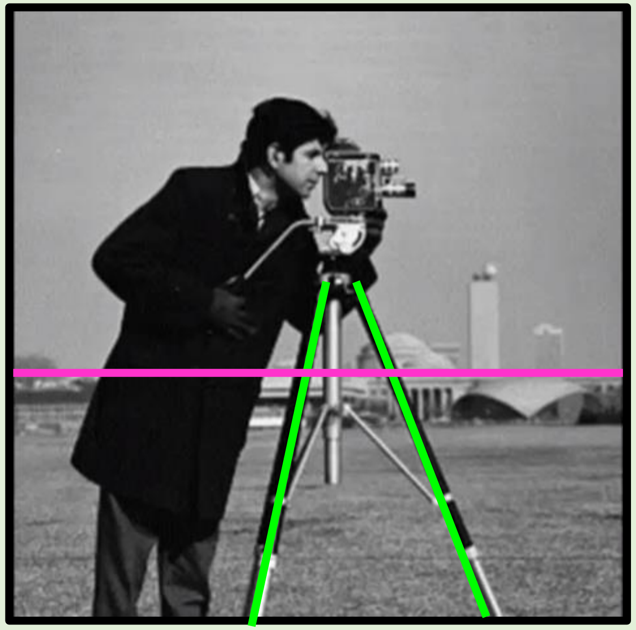
Frequency

frequency kalo diarahkan kita bisa mengetahui seberapa sering pola muncul

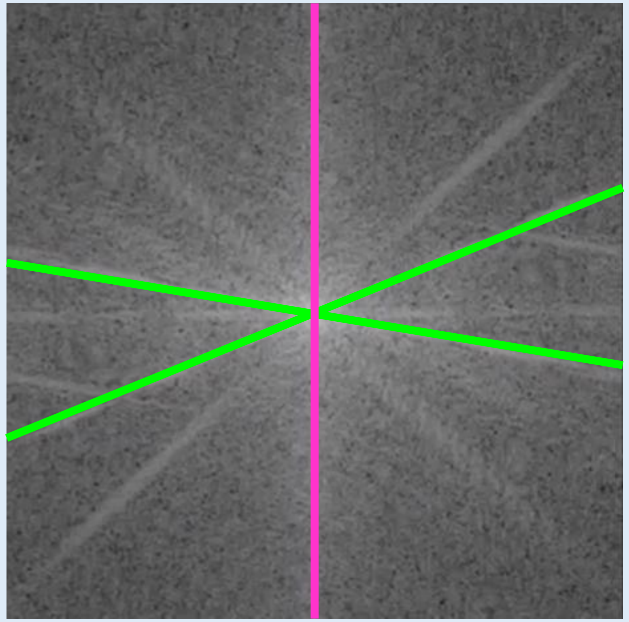


# Image to Frequency (3)

Spatial



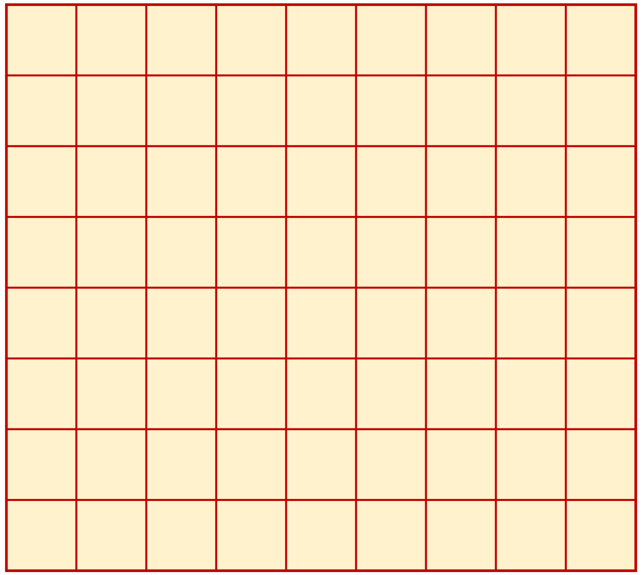
Frequency



We can exploit the corresponding patterns in the frequency domain to understand the shapes in the image.

# 1D **Discrete** Fourier Transform (DFT)

- Digital images have **discrete** values.
- We need **discrete** Fourier transform.



- Forward DFT

$$F(u) = \sum_{x=0}^{N-1} f(x) e^{-j\frac{2\pi u x}{N}}, \qquad u = 0,1,\ldots.N-1$$

- Inverse DFT

$$f(x) = \frac{1}{N} \sum_{u=0}^{N-1} F(u) e^{j\frac{2\pi u x}{N}}, \qquad x = 0,1,\ldots.N-1$$

# 1D DFT – The Math

- Forward DFT

$$F(u) = \sum_{x=0}^{N-1} f(x)\boxed{e^{-j\frac{2\pi ux}{N}}}, \quad u = 0,1,\ldots.N-1$$

$$\boxed{\cos\left(\frac{2\pi ux}{N}\right) - jsin(\frac{2\pi ux}{N})}$$

| $x$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $f(x)$ | 0 | 31 | 64 | 127 | 255 |

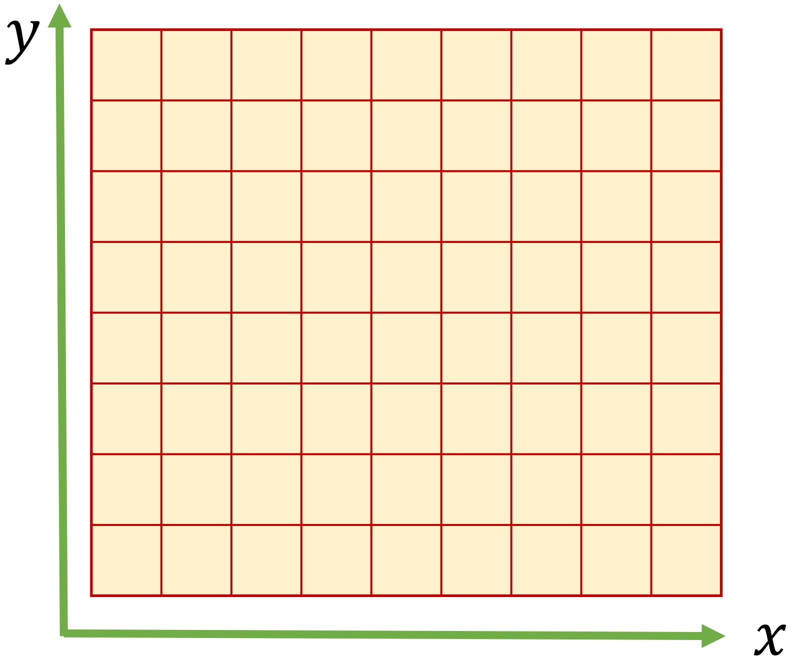
| $u$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $F(u)$ | | | | | |

- $F(0) = \sum_{x=0}^{4} f(x)e^{-j\frac{0}{5}} = \sum_{x=0}^{4} f(x) = 476$
- $F(1) = \sum_{x=0}^{4} f(x)e^{-j\frac{2\pi ux}{N}} = \sum_{x=0}^{4} f(x)\left[\underbrace{\cos\left(\frac{2\pi x}{5}\right)}_{\text{Real}} \underbrace{- j\sin\left(\frac{2\pi x}{5}\right)}_{\text{Imaginary}}\right]$ **Complex Number!** $\boxed{C = +jI}$

$$\begin{aligned} &= 0 + 31(\cos(2\pi/5) - j\sin(2\pi/5)) \\ &\quad + 63(\cos(4\pi/5) - j\sin(4\pi/5)) \\ &\quad +127(\cos(6\pi/5) - j\sin(6\pi/5)) \\ &\quad + 255(\cos(8\pi/5) - j\sin(8\pi/5)) = -65.33 + 250.65j \end{aligned}$$

# 2D Discrete Fourier Transform (DFT)

- A digital image is **2 dimensional**
- Apply the 1D DFT in **2 directions**




- Assume that $f(x, y)$ is $M \times N$ image
- Forward DFT

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

$$u = 0,1, \dots . M - 1, v = 0,1,2, \dots . , N - 1$$

- Inverse DFT

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

$$x = 0,1, \dots . M - 1, y = 0,1,2, \dots . , N - 1$$

# Real and Imaginary Results of 2D DFT

Real | Imaginary

$$\left[\cos\left(2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right) - j\sin\left(2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right)\right]$$

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

$M = N = 64$ (image size), $u = 6$ AND $v = 6$ (index)

$$\cos\left(2\pi\left(\frac{6x}{64}+\frac{6y}{64}\right)\right)$$

basis real

Real Part

Imaginary Part

$$\sin\left(2\pi\left(\frac{6x}{64}+\frac{6y}{64}\right)\right)$$

basis imaginary

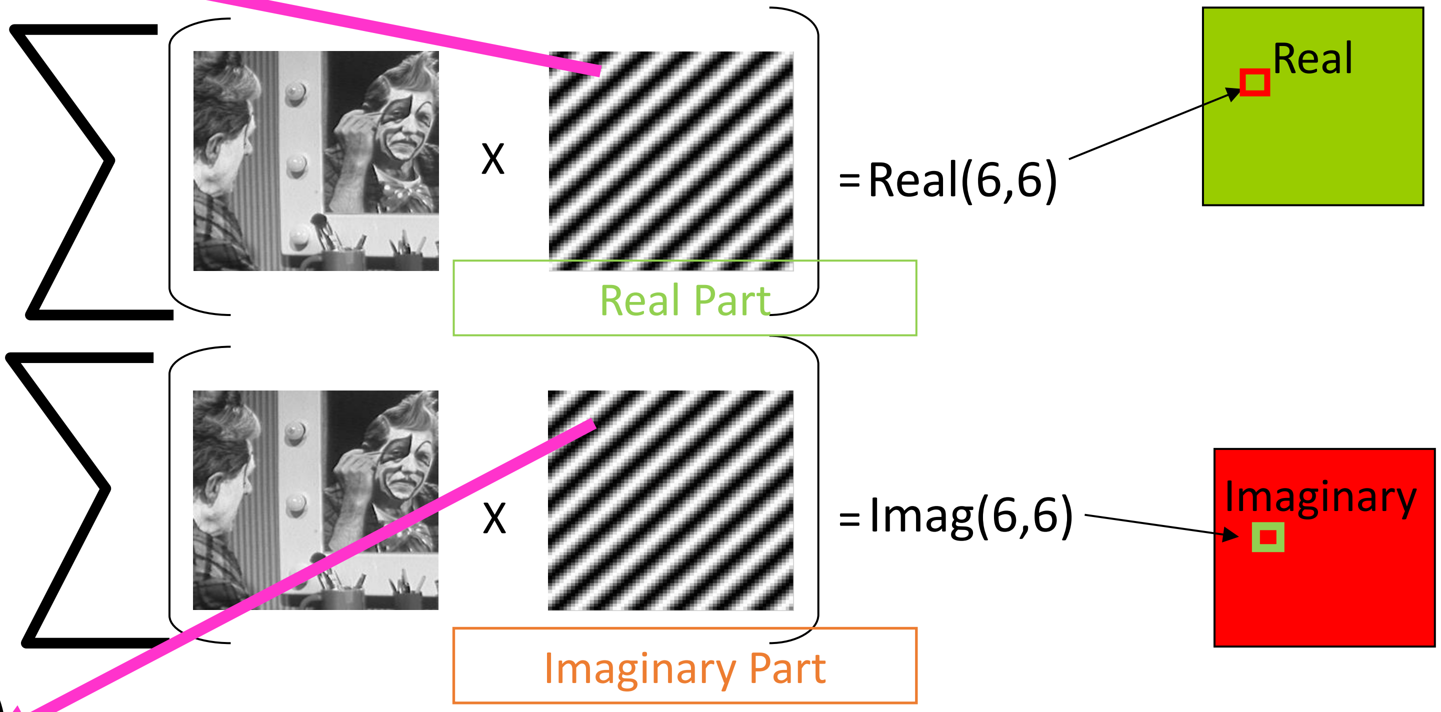
These are the Fourier Basis functions

# Fourier Basis Functions

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\left[\cos\left(2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right) - j\sin\left(2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right)\right]$$

$M = N = 64$ (image size), $u = 6$ AND $v = 6$ (index)

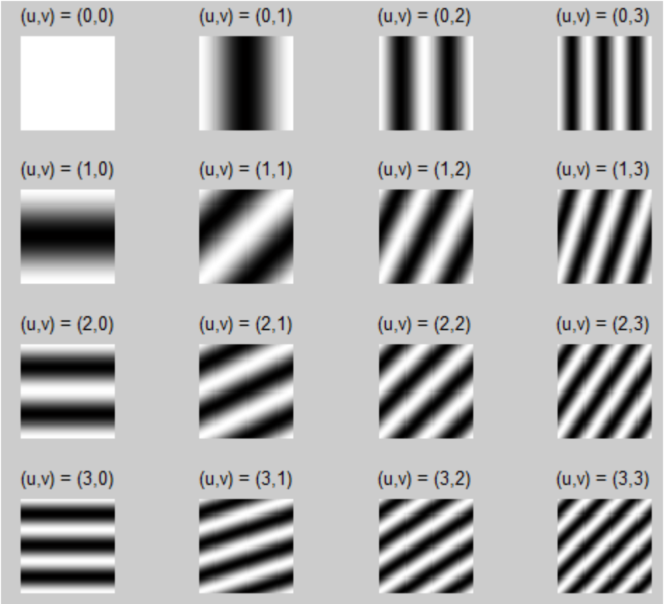
$$\cos\left(2\pi\left(\frac{6x}{64}+\frac{6y}{64}\right)\right)$$



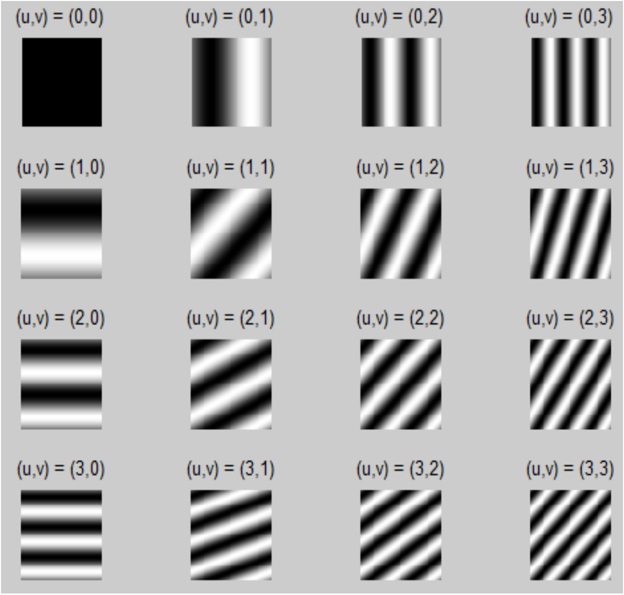

$$\sin\left(2\pi\left(\frac{6x}{64}+\frac{6y}{64}\right)\right)$$

# Fourier Basis Functions (2)

- In this example, we used a $64 \times 64$ image.
- The Fourier transform result has the same size.
- Thus, we have $64 \times 64 = 4096$ complex basis functions for this image.




Real Part



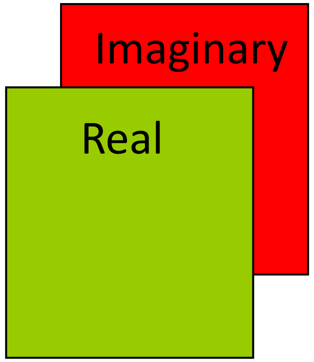

Imaginary Part

dari Basis Function, akan mendaptkan koefisien untuk mepresentasikan citra

# 2D DFT

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

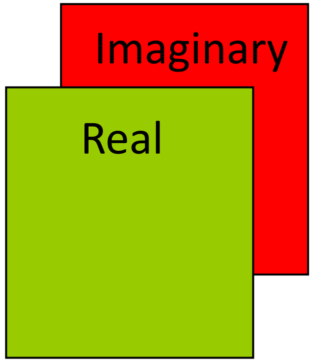
Kalau ada citra 2 Dimensi




complex valued matrix

# Properties of the 2D DFT

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




complex valued matrix

- $\boldsymbol{F}(\boldsymbol{u})$ is a complex function : $F(u) = R(u) + j\, I(u)$
- Magnitude of FT (spectrum) : $|F(u)| = \sqrt{R^2(u) + I^2(u)}$
- Phase of FT : $\phi\left(F(u)\right) = \tan^{-1}\left(\frac{I(u)}{R(u)}\right)$
- Power of $f(x)$: $\boldsymbol{P}(\boldsymbol{u}) = |F(u)|^2$ : $R^2(u) + I^2(u)$

# Visualizing 2D DFT

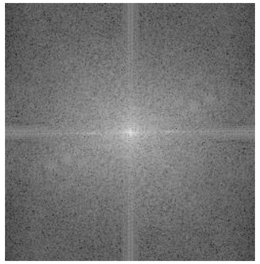
$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Imaginary

Real

complex valued matrix

$$\sqrt{\text{Real}^2 + \text{Imaginary}^2} =$$

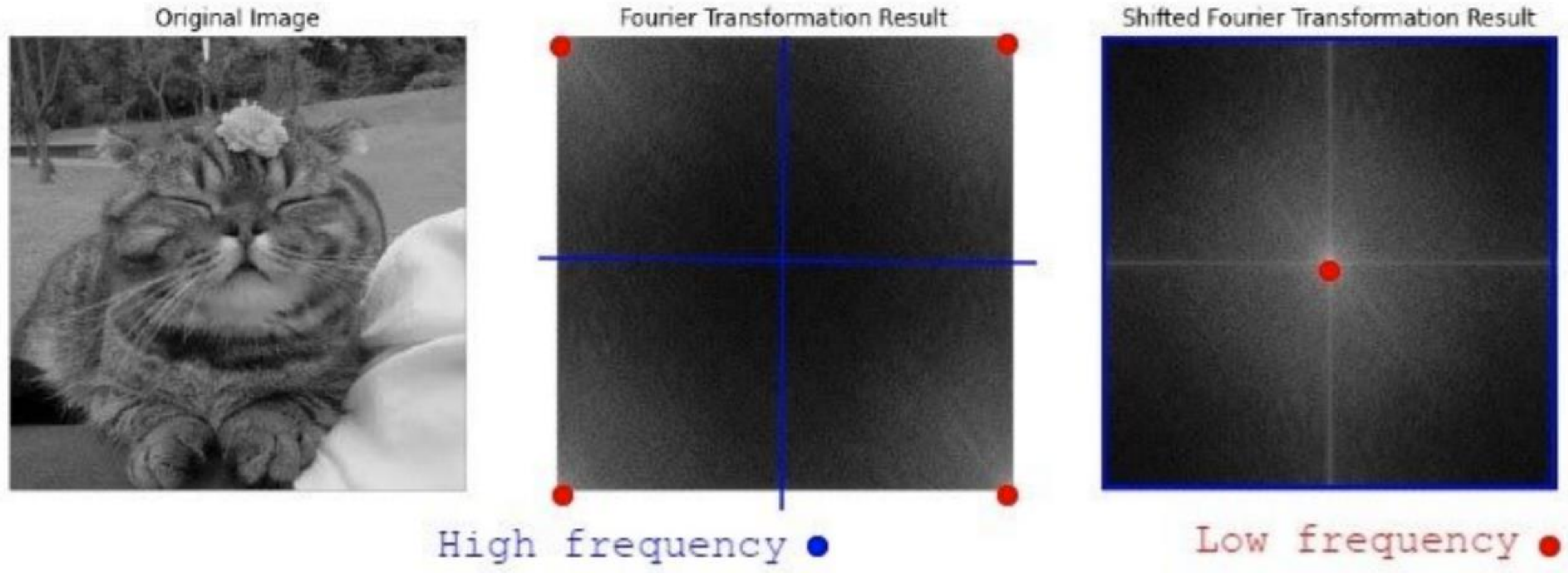


$|F\ (u,v)|$

The 2D DFT on images will be identical in size!

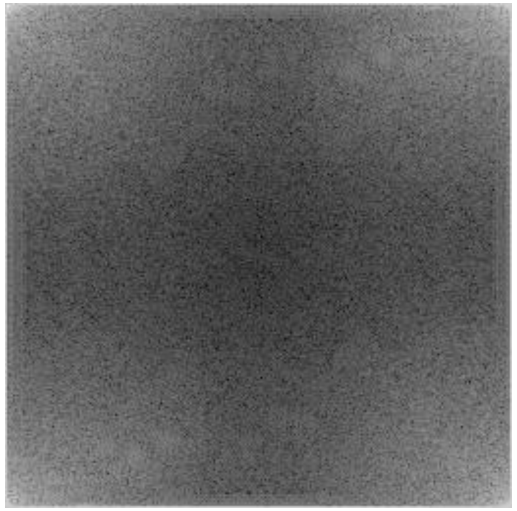
# Visualizing 2D DFT – Shifting

- A Fourier Transform is well visualized with the Zero Frequency component in the center N/2 (N/2,N/2 in case of 2D)

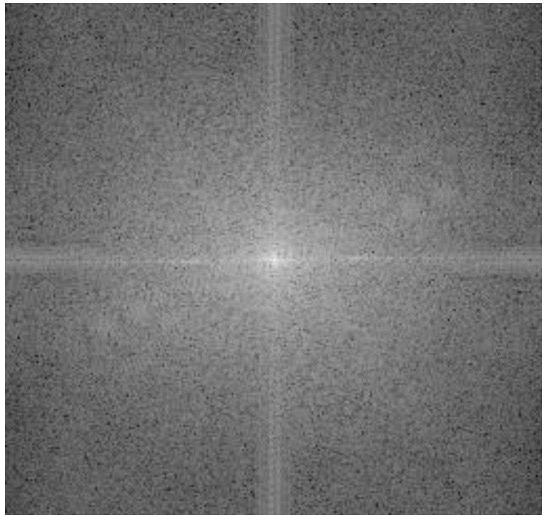


# Visualizing 2D DFT – Shifting (2)

original image



before shifting



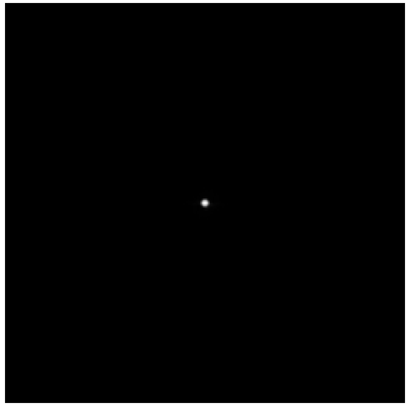
after shifting

# Visualizing 2D DFT - Scaling

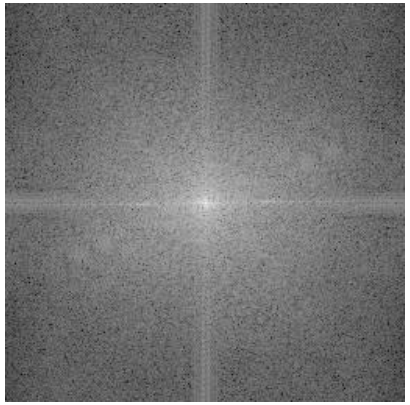
- The Dynamic range of Fourier spectra usually is much higher than the typical display device is able to reproduce faithfully.
- We often use the logarithm function to perform the appropriate compression of the range.

$$D(u, v) = c\log(1 + |F(u, v)|)$$

original image



before scaling

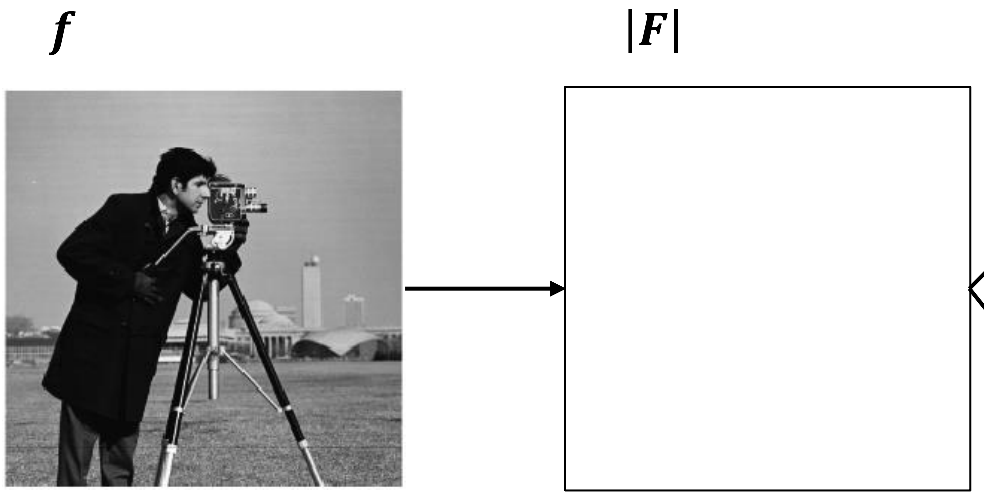


after scaling

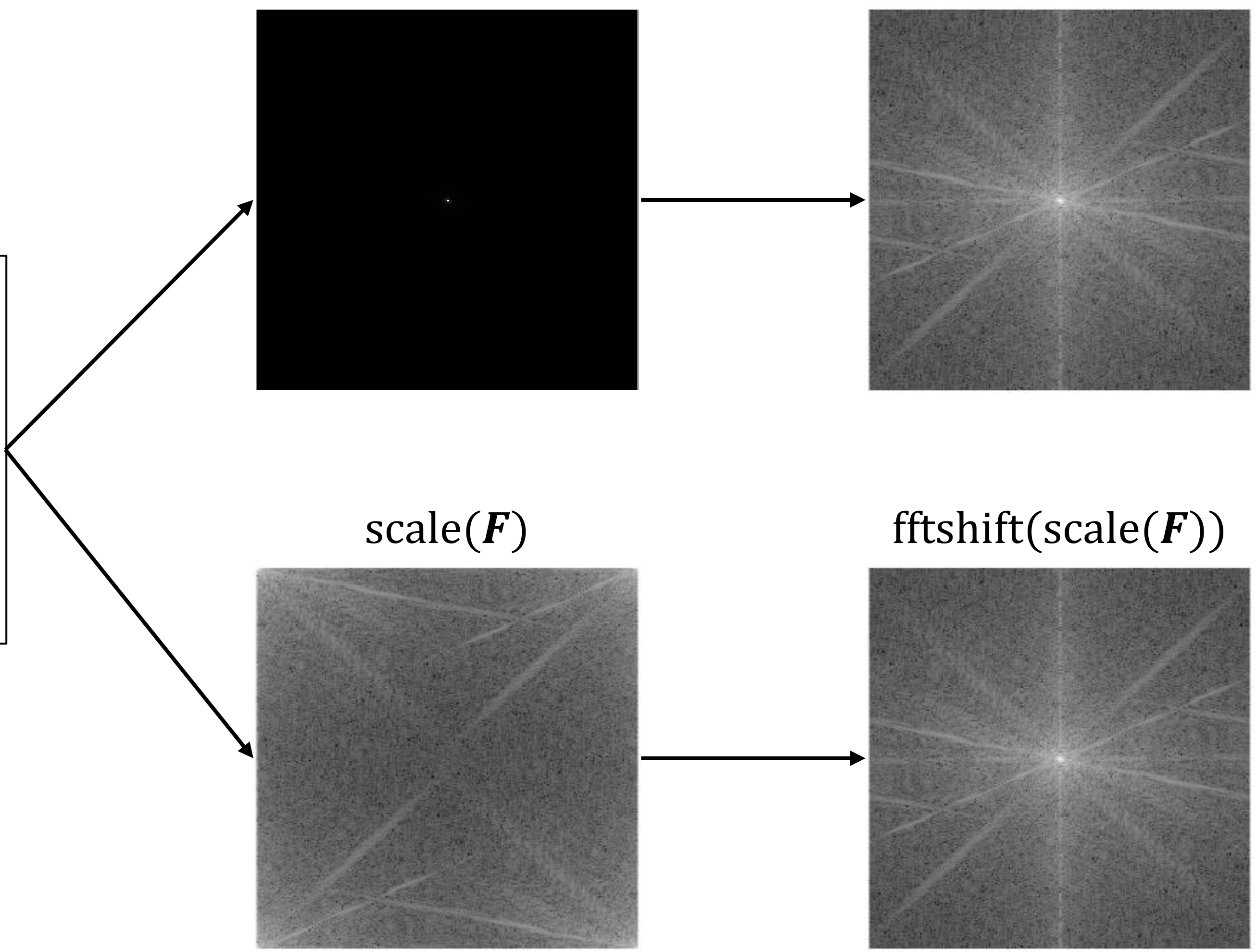
# Shift or scale first?

Urutan bebas, Harus melakukan untuk mendapatkan visualisasi yang optimal

- It doesn't really matter..

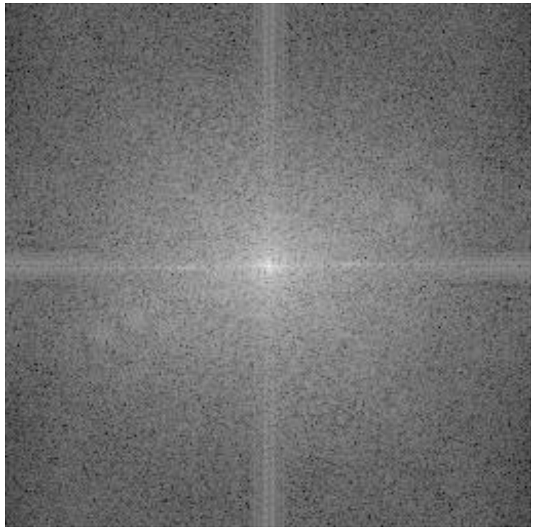



$\text{fftshift}(\boldsymbol{F})$ $\text{scale}(\text{fftshift}(\boldsymbol{F}))$

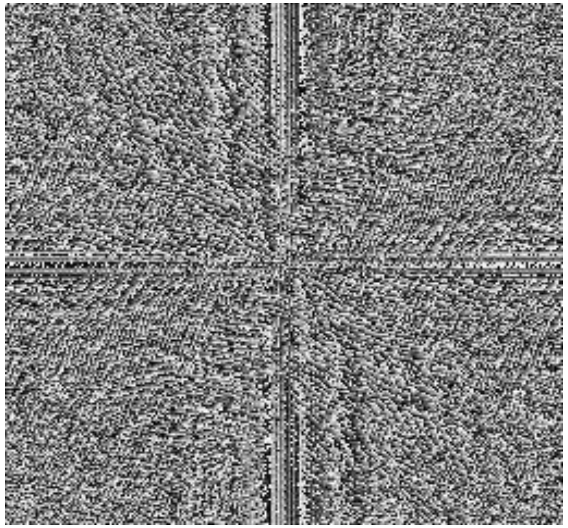


# Visualizing 2D DFT

$f\ (x, y)$



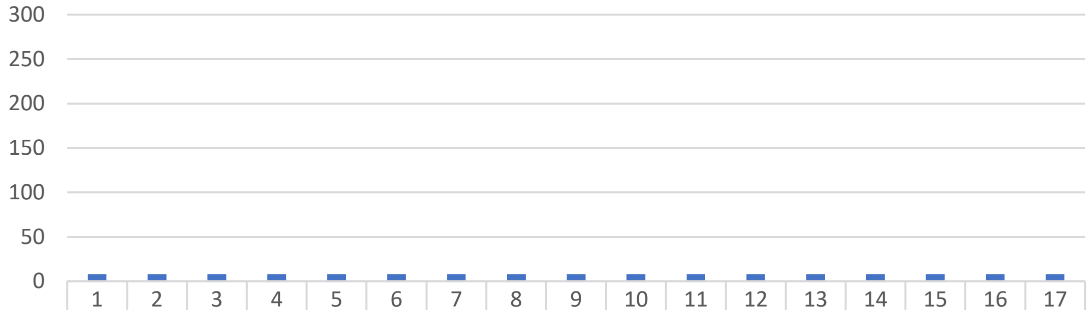
Magnitude
$|F\ (u\ , v)|$



Phase
$\Phi\ (F\ (u, v))$

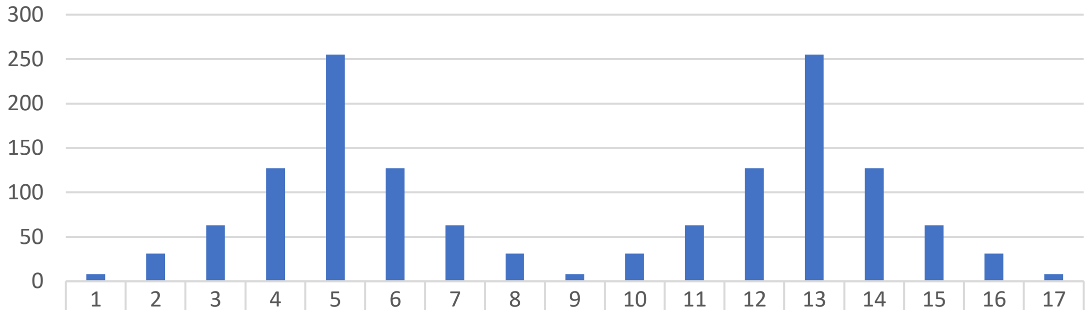
# Low/High Frequency in Discrete Data

## Black



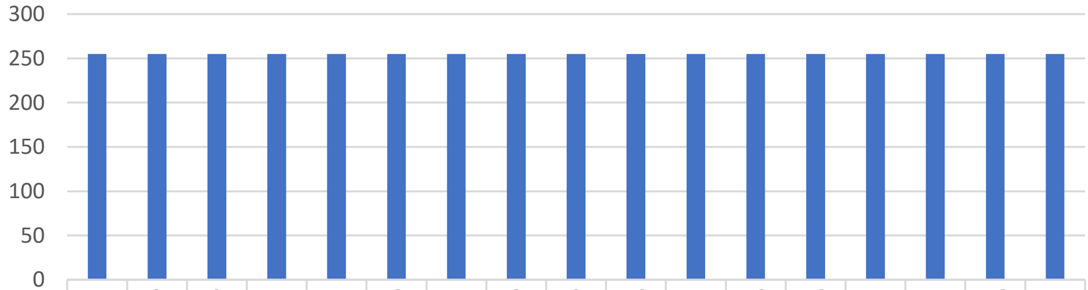

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X1 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |

## Edges/Lines



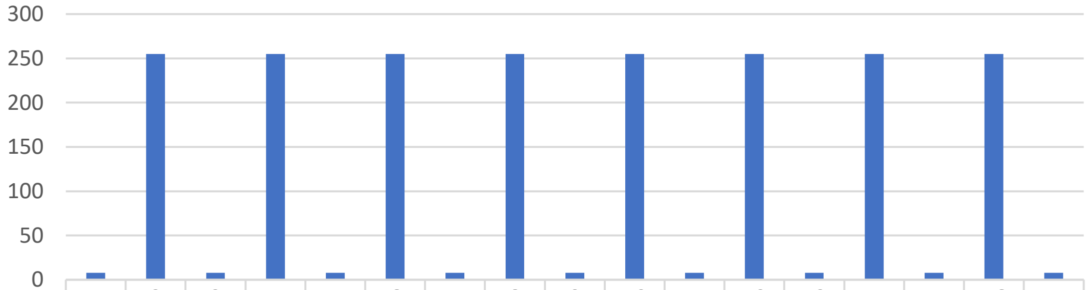

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X2 | 8 | 31 | 63 | 127 | 255 | 127 | 63 | 31 | 8 | 31 | 63 | 127 | 255 | 127 | 63 | 31 | 8 |

## White




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X3 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |

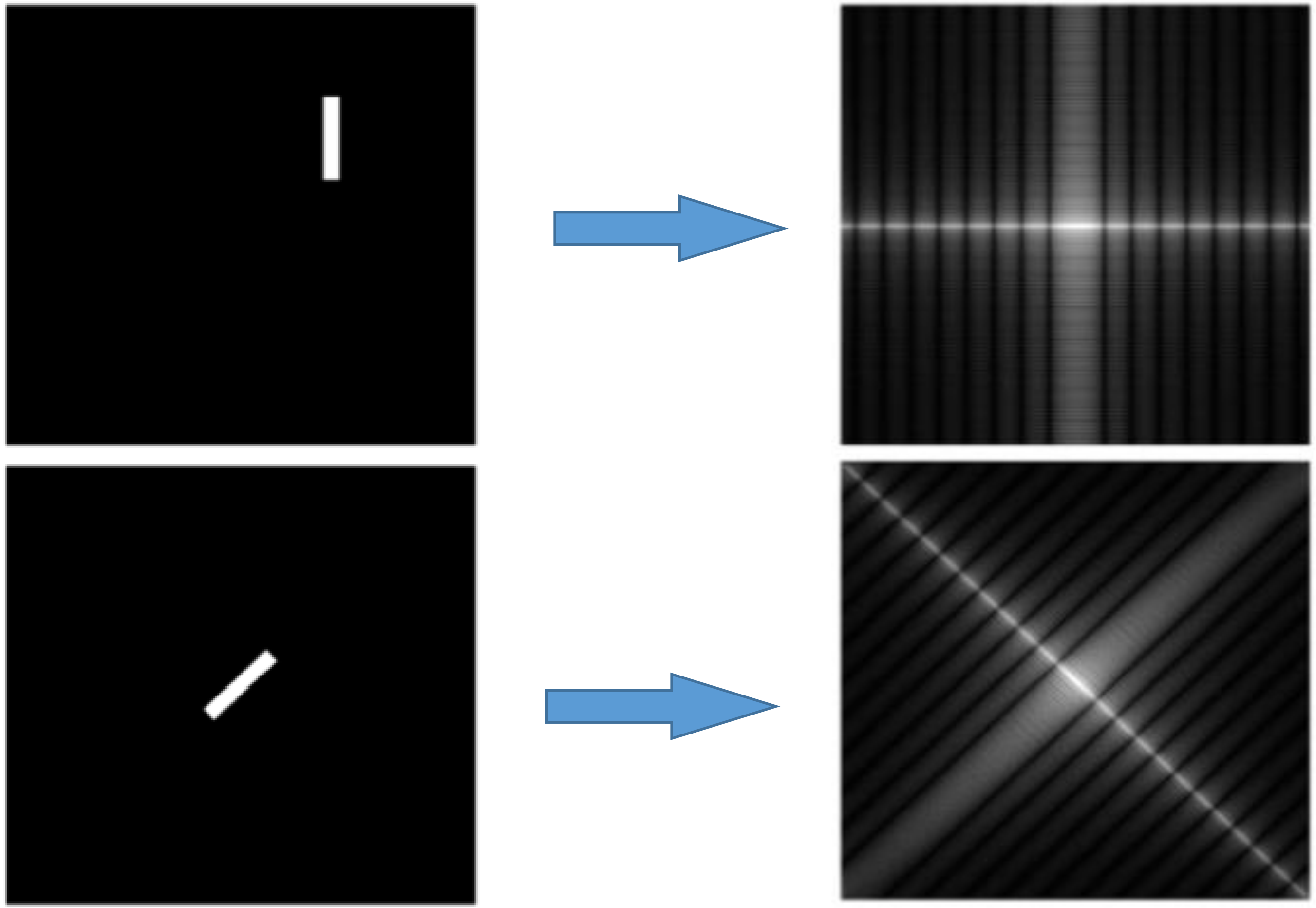
## Points/Noises



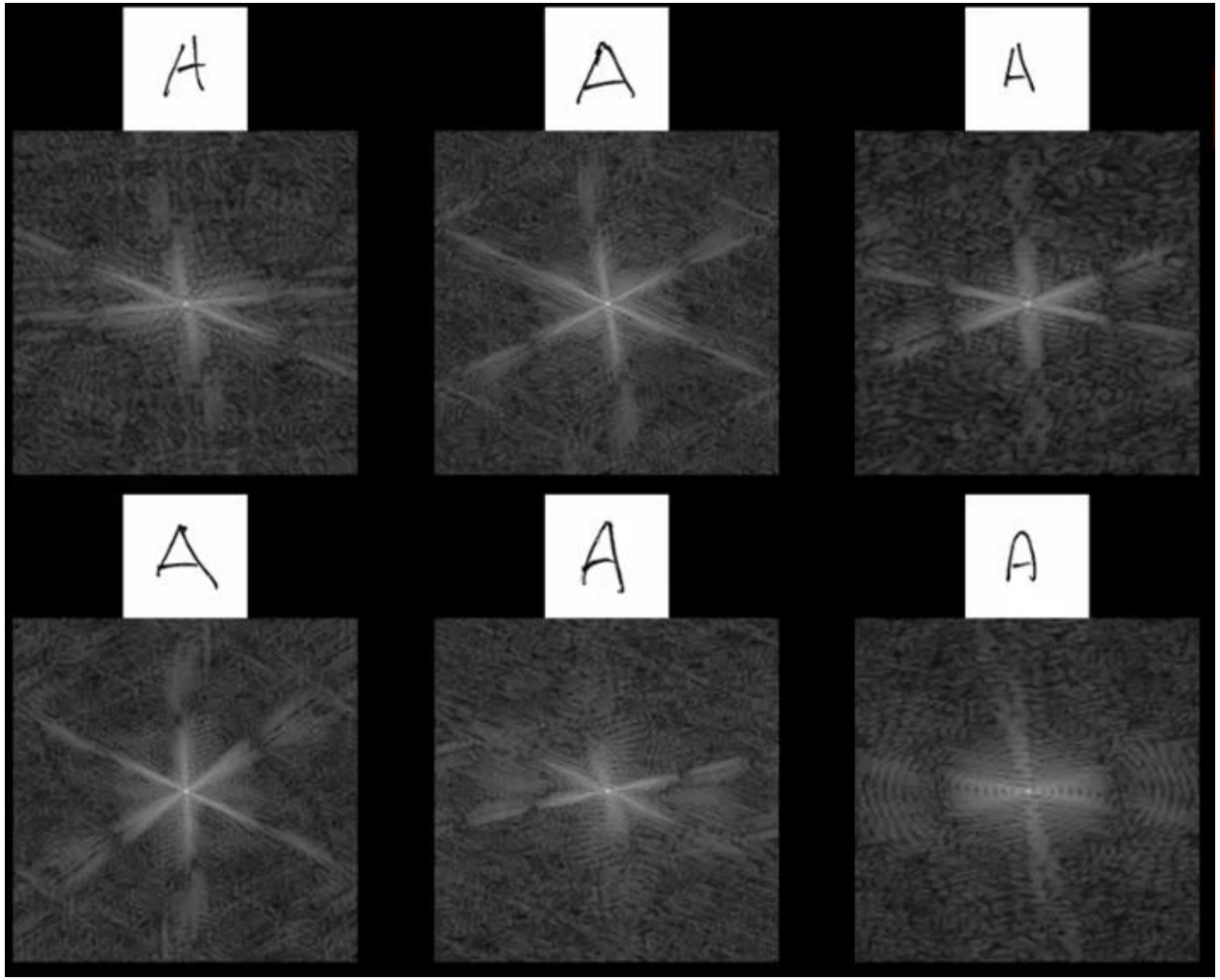

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X4 | 8 | 255 | 8 | 255 | 8 | 255 | 8 | 255 | 8 | 255 | 8 | 255 | 8 | 255 | 8 | 255 | 8 |

# Fourier Transform of Basic Shapes

kalau cuman shape kiri, pasti muncul gambar kanan



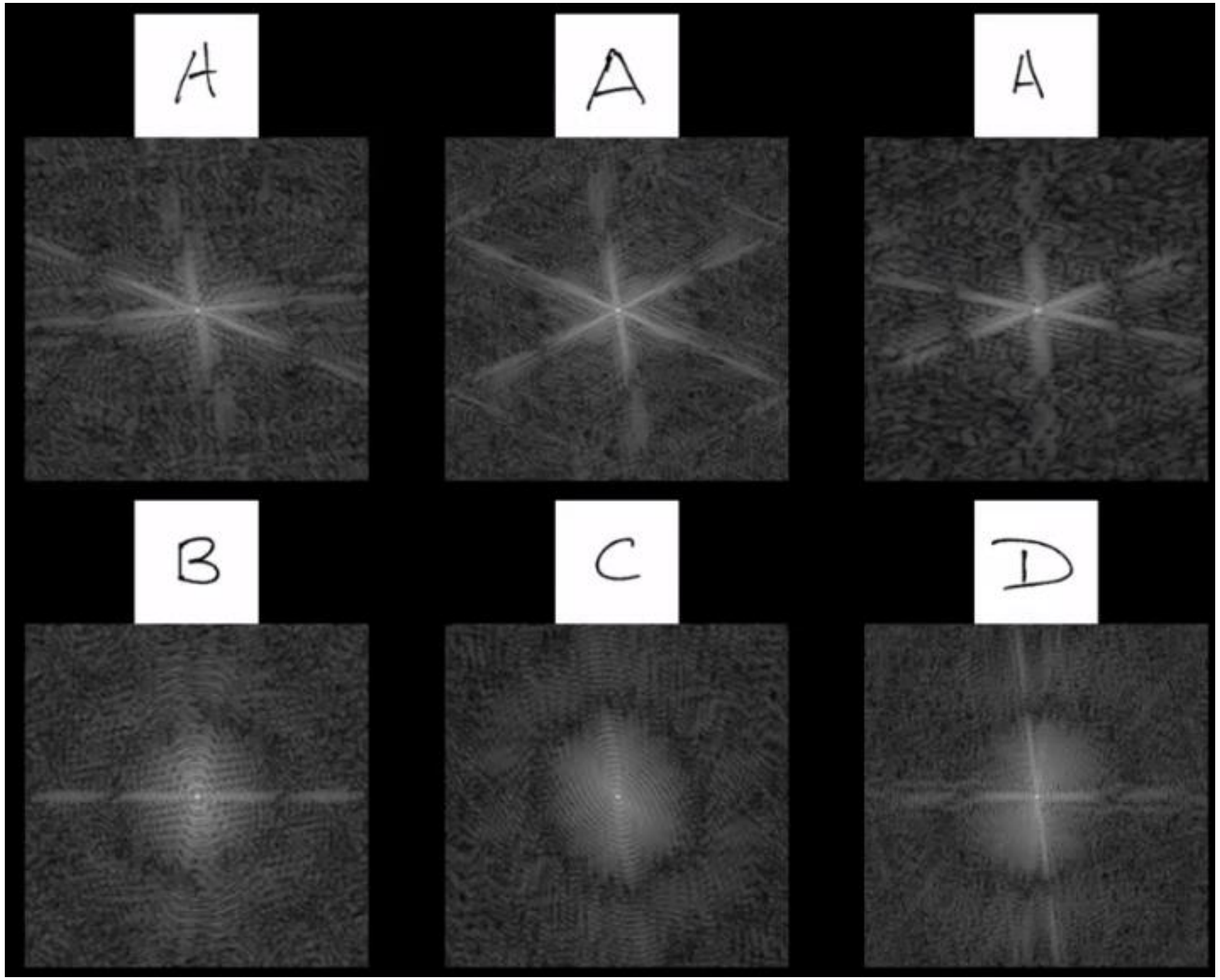
# Fourier Transform for Character Recognition




kalau karakter, gak banyak frequency dalam gambar, maka transformasi fourier sangat mirip.

Jadi fourier transform bisa klasifikasi huruf.

# Fourier Transform for Character Recognition (2)

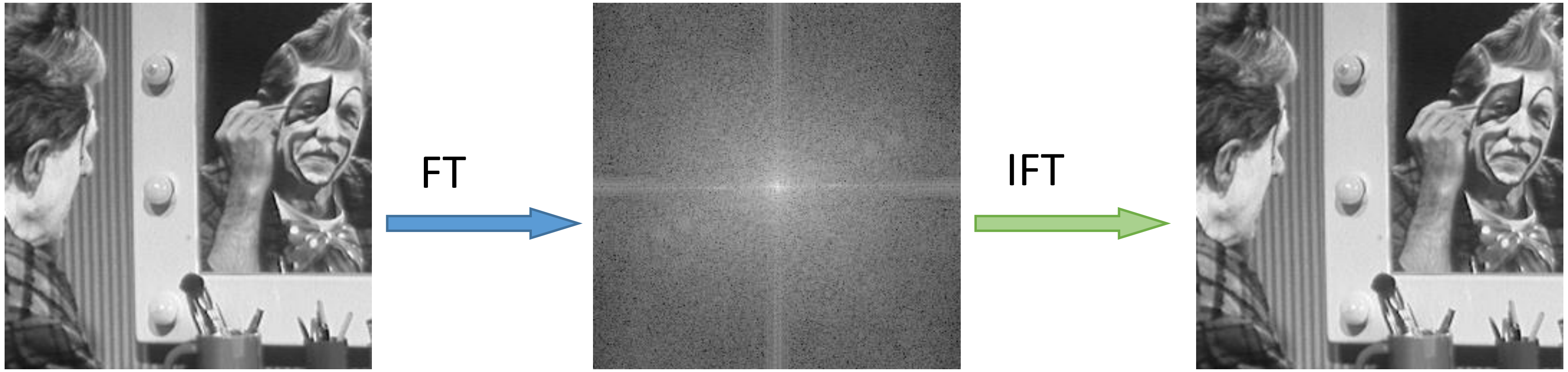


Selain mengubah dari spatial ke frekuensi.

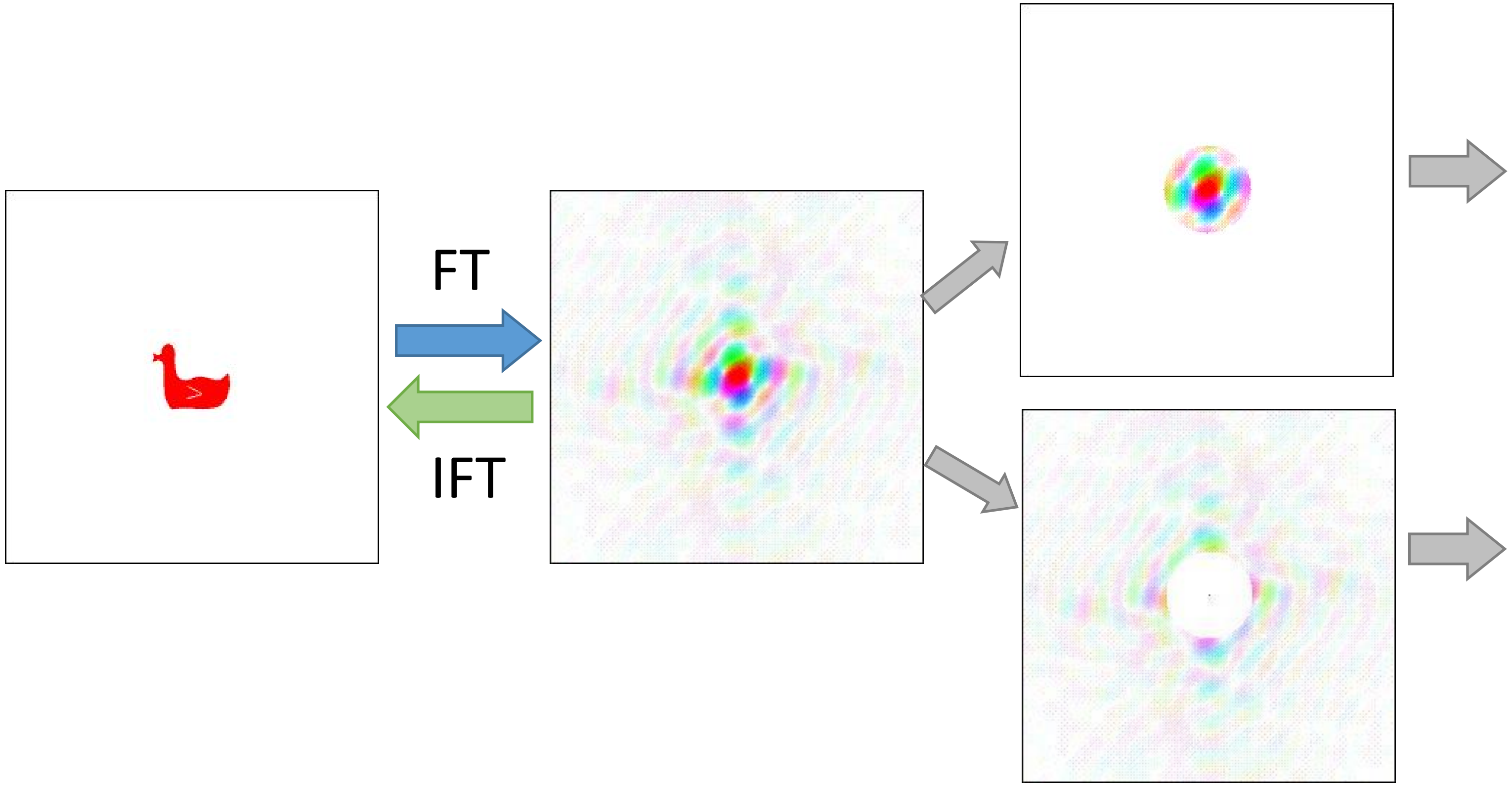
Kita juga bisa mengembalikan dari frekuensi ke gambar

# Inverse Fourier Transform

- Fourier series and transformation can be returned to the original form using an inverse function
- It’s easy to move between domains, process the frequency, then invert back



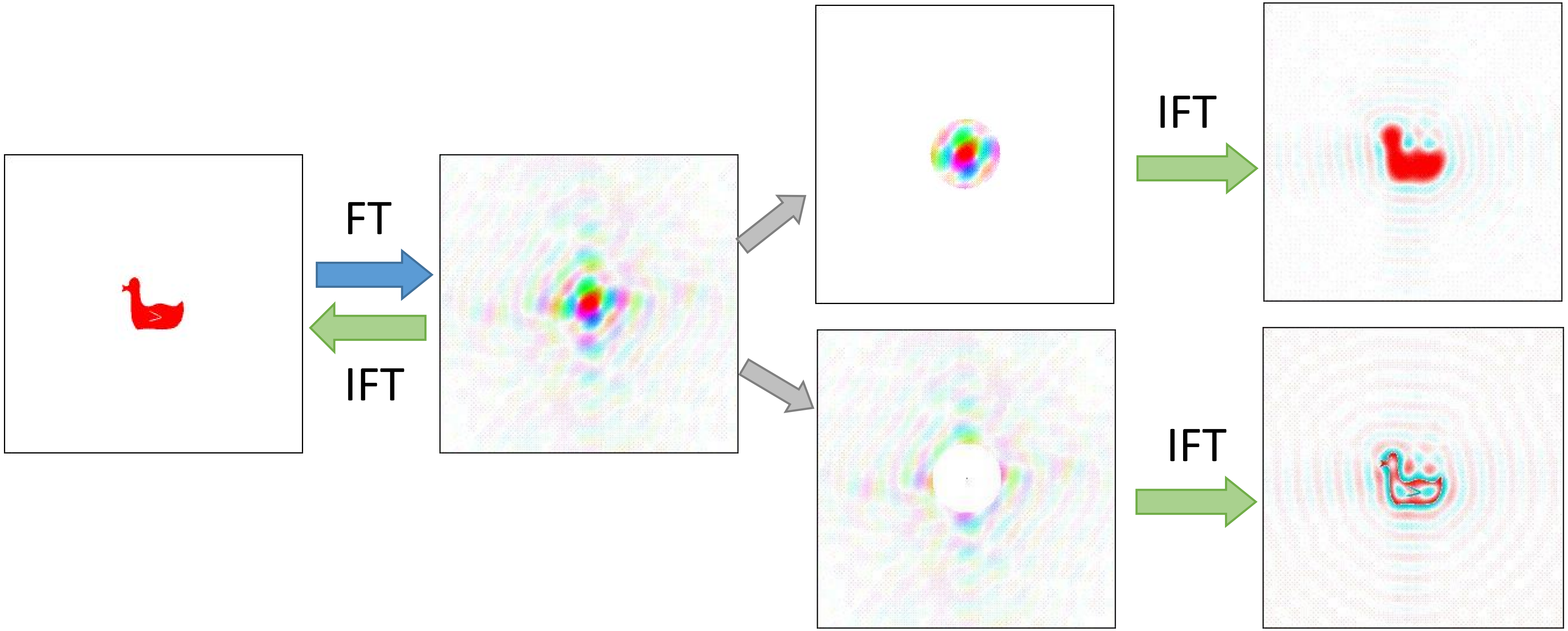
# 2D DFT of an Image and its Inverse



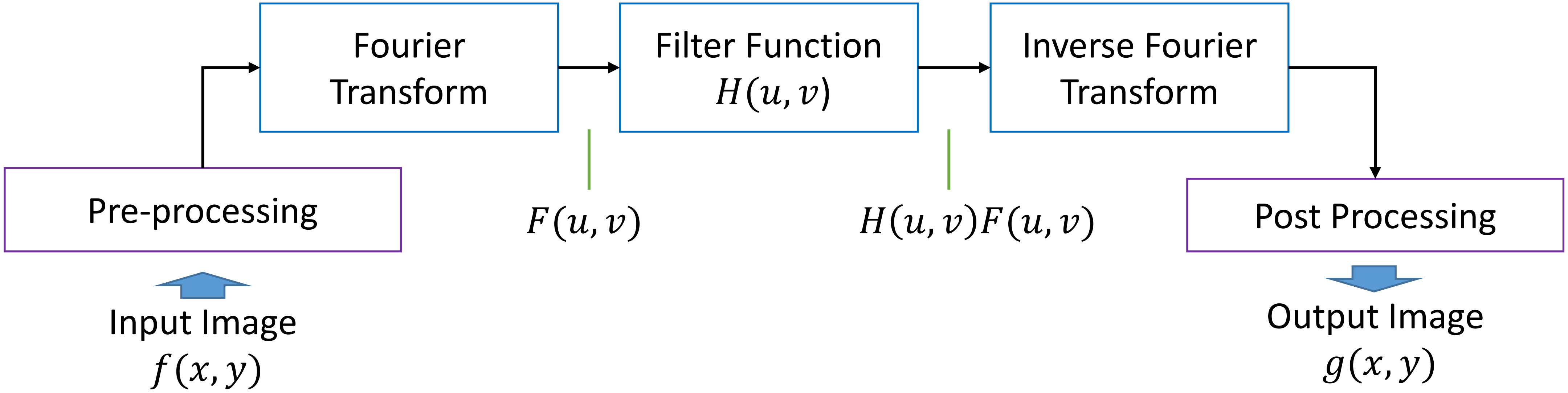

The central part of FT, i.e. the low frequency components are responsible for the general gray-level appearance of an image.

The high frequency components of FT are responsible for the detail information of an image.

# Filtering the Frequencies



# Filtering Fourier Transform

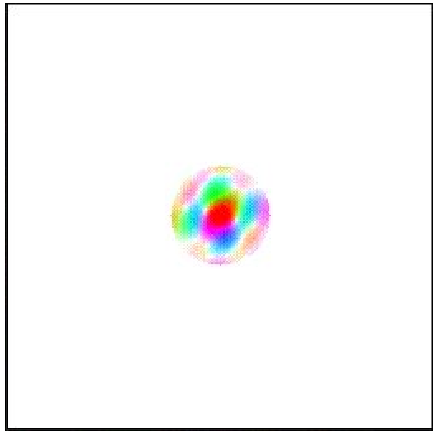



- Common Filters:
  - Band Pass Filters (Low and high pass filters)
  - Band Reject Filters

# Low Pass Filters

Low pass tengah aja:

Yang low dibiarkan lewat (hanya frequency low yang masih ada di image).

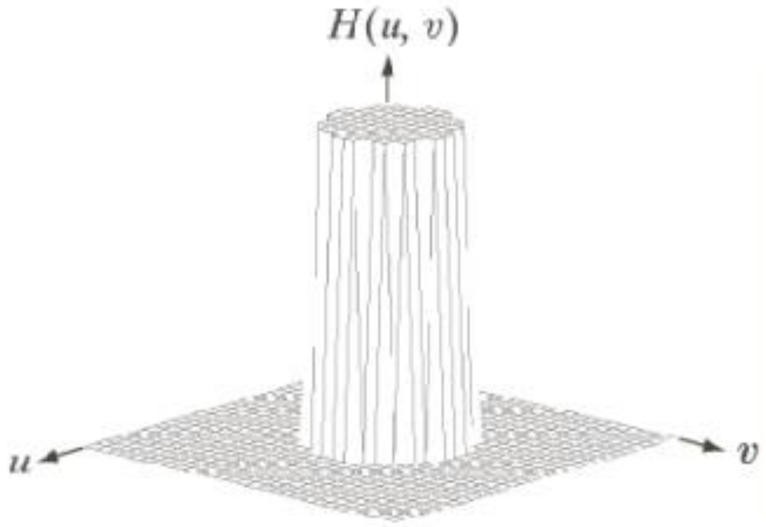


- What happens?
  - Only the patterns with low frequencies are retained in the image
  - The high frequency (details) are removed
- Smoothing!
  - Smoothing happens because noise is high-frequency. The low pass filter removes that
  - But, we can also loose detail.

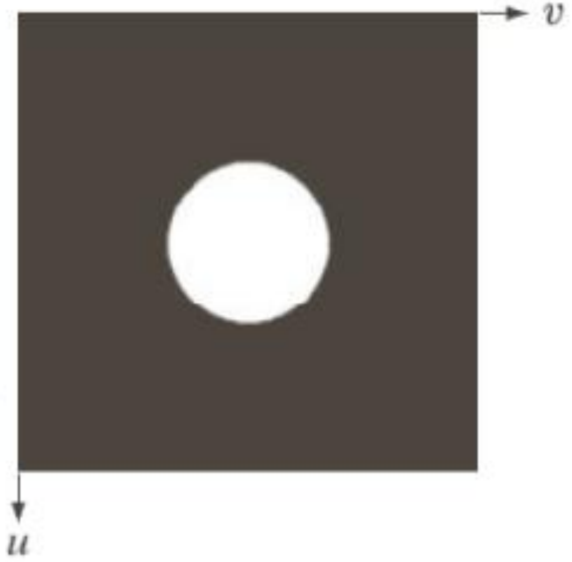
- $H(u,v) \quad = 1 \; if \; D(u,v) \leq D_0$
  $\quad\quad\quad\quad = 0 \; if \; D(u,v) > D_0$
- $D_0$ is the threshold for frequency cutoff, $D_0 > 0$)
- $D(u,v)$ is the distance from origin. $D(u,v) = \sqrt{(u^2 + v^2)}$

# Ideal Low Pass Filter

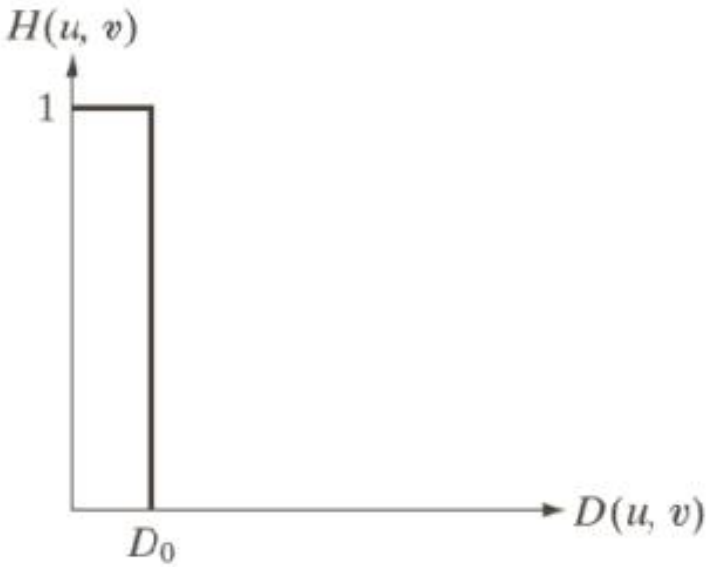
Kalau frequency dibawah threshold tertentu, masuk digambar, otherwise not.




Perspective plot of ideal LPF

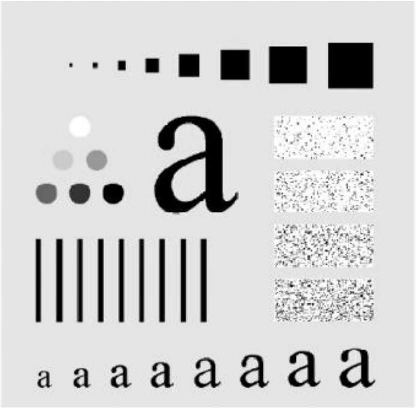


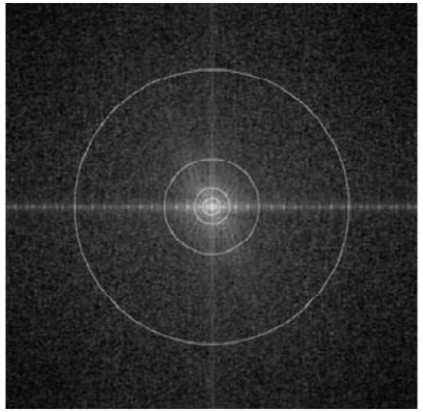

Filter as an image




Radial Cross Section

# Ideal Low Pass Filter Example



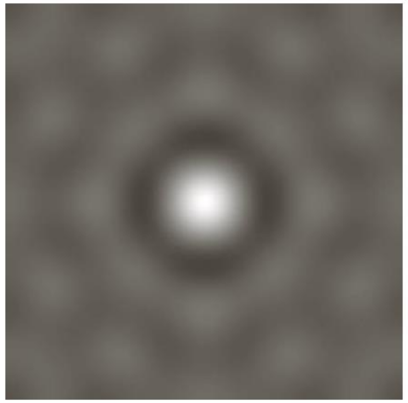


- Radius low pas filter: 10,30,60,160, 460

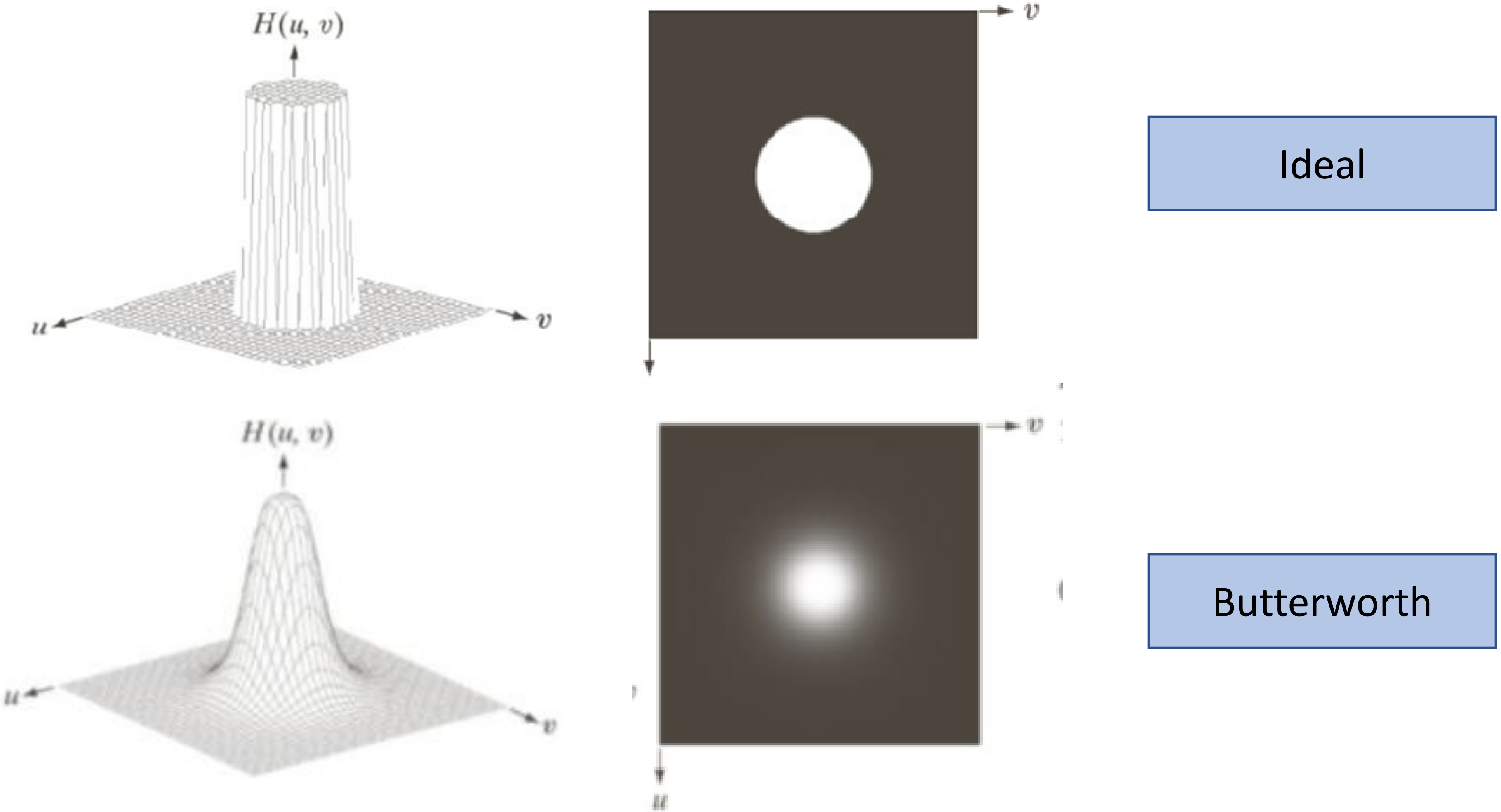
hasil low pass filter

# Ringing Effect

- Why?
- Visually, they appear as bands or "ghosts" near edges.
- The term "ringing" is because the output signal oscillates at a fading rate around a sharp transition in the input, similar to a bell after being struck

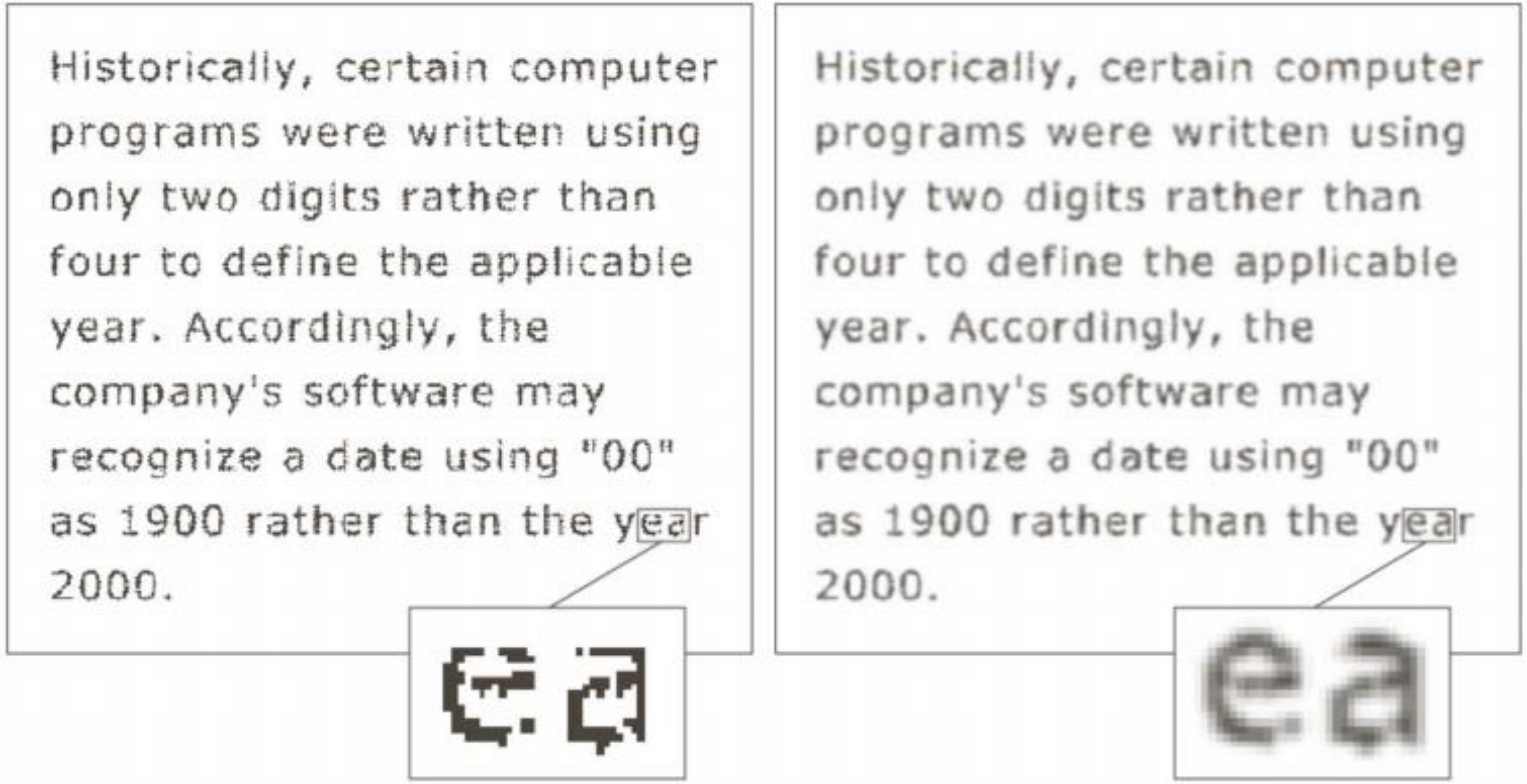


# Butterworth Low Pass Filter



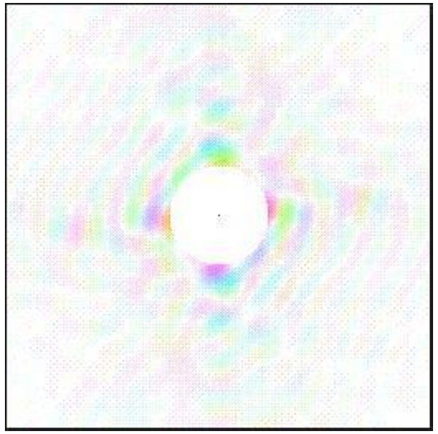
# Potential Applications

- Machine perception and OCR



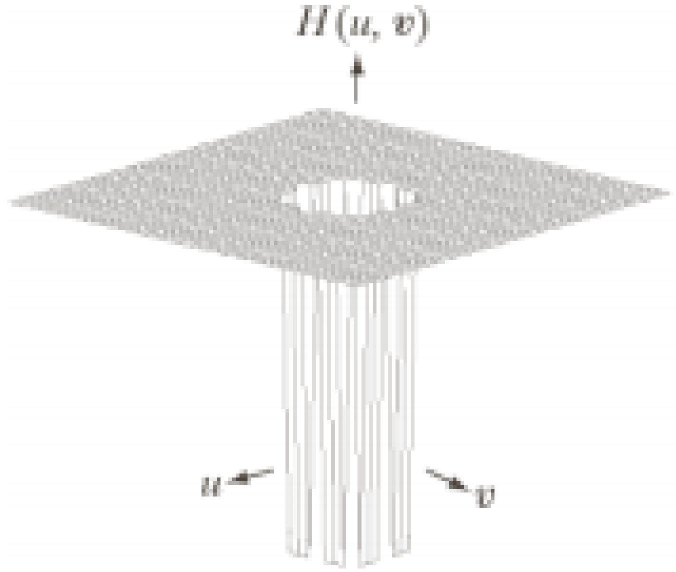
# High Pass Filters

- What happens?
  - Only patterns with high frequencies are retained
  - The low frequencies (big shapes) are removed
- Sharpening!
  - Edges and abrupt changes in intensities are associated with high-frequency.
  - Lessens the low-frequency components without disturbing high-frequency information.

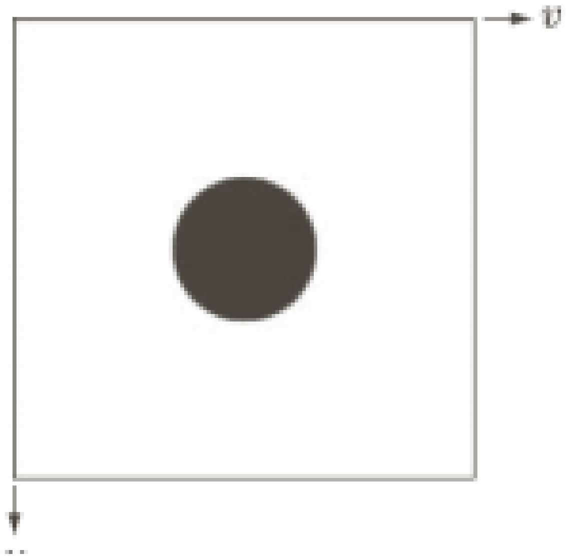


- $H(u,v) \quad = 0 \; if \; D(u,v) \leq D_0$
  $\qquad\qquad = 1 \; if \; D(u,v) > D_0$
- $D_0$ is the threshold for frequency cutoff, $D_0 > 0)$
- $D(u,v)$ is the distance from origin. $D(u,v) = \sqrt{(u^2 + v^2)}$

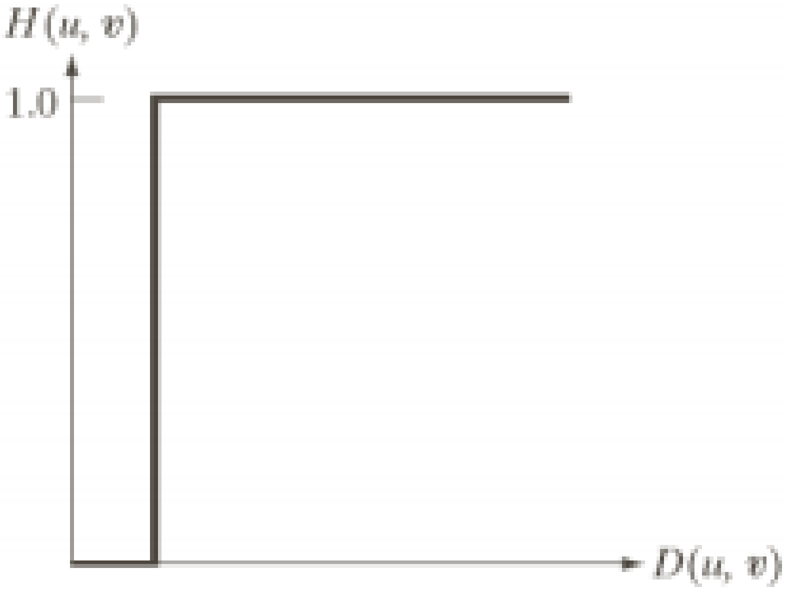
# Ideal High Pass Filter




Perspective plot of ideal HPF

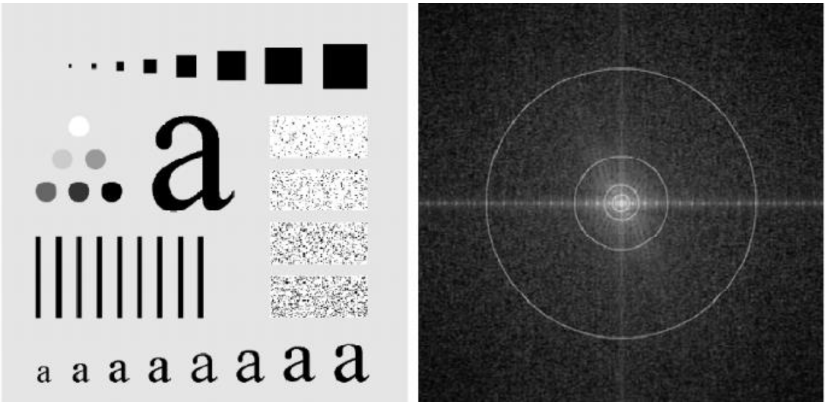



Filter as an Image



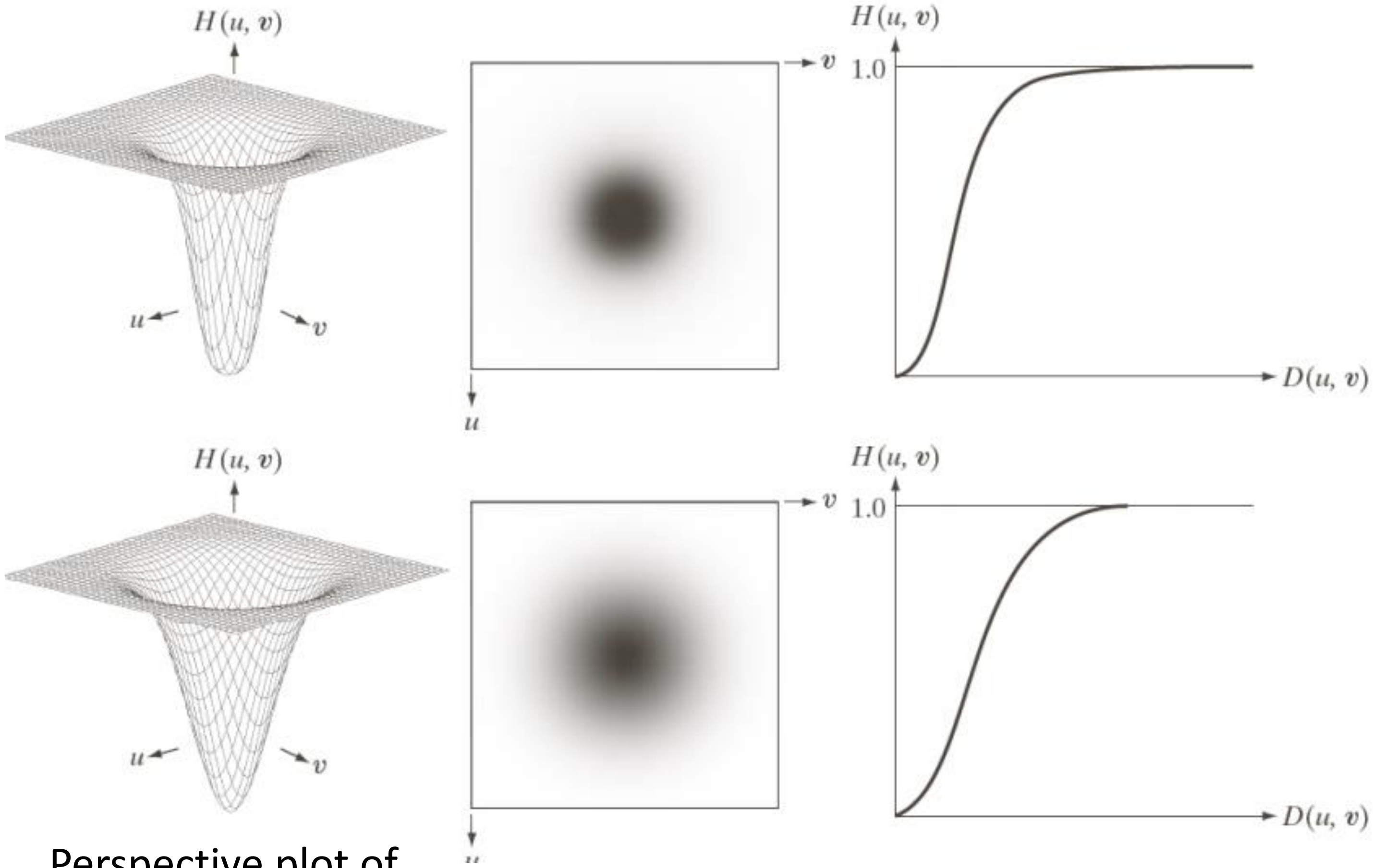

Radial Cross Section

# Ideal High Pass Filter Example



- Ideal High Pass Filter results with $D_0 = 30{,}60, \text{and } 160$

# Butterworth and Gaussian High Pass Filter




Butterworth

Gaussian

Perspective plot of ideal HPF

Filter as an image

Radial Cross Section

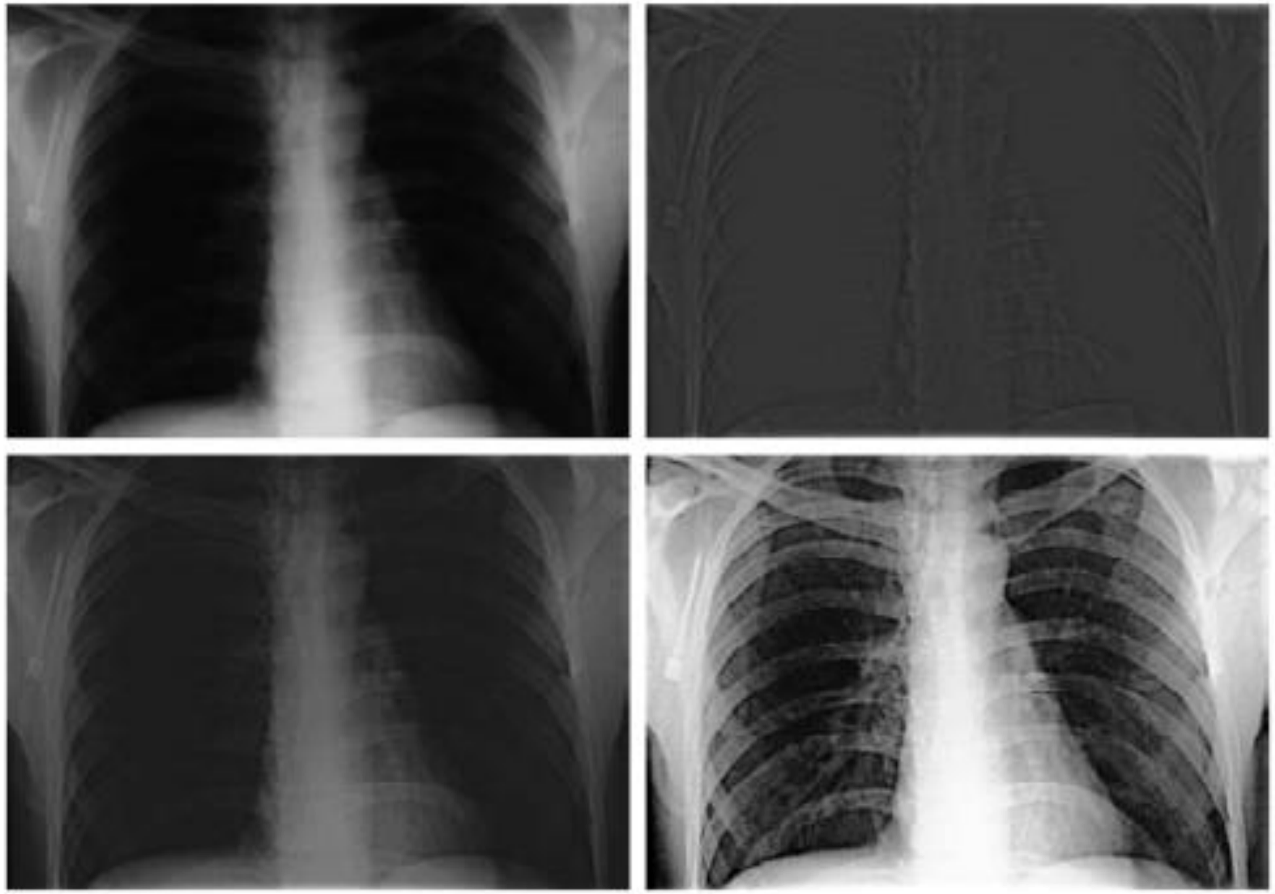
# Butterworth and Gaussian High Pass Filter Example

- Butterworth High Pass Filter results with $D_0 = 30{,}60, 160$

- Gaussian High Pass Filter results with $D_0 = 30{,}60, \text{and } 160$

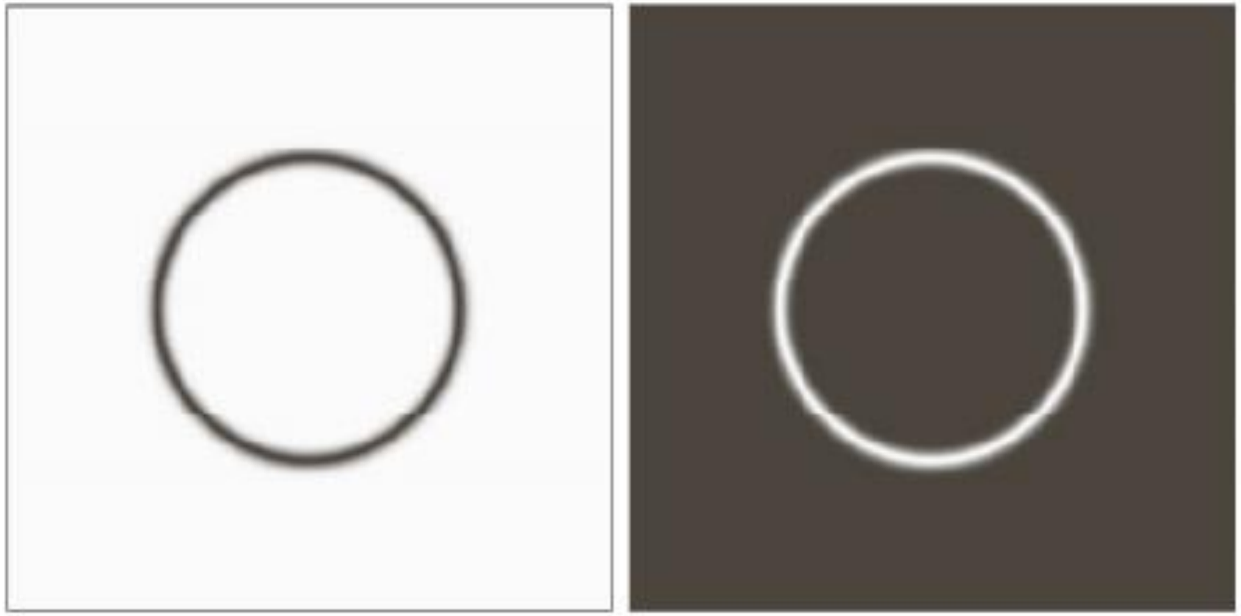
# Potential Applications

- Medical Field
  - HPF, HPF emphasis, and histogram equalization



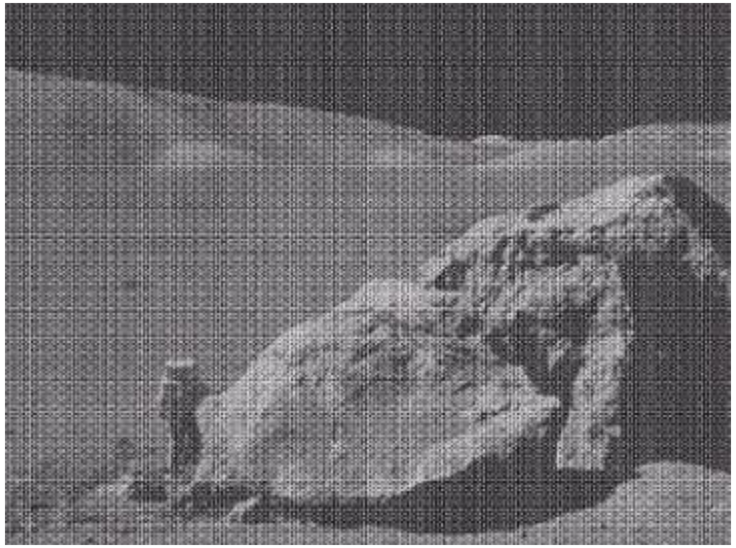
# Selective Filtering

- Band Reject Filters / Band Pass Filters
  - What do you think will happen?

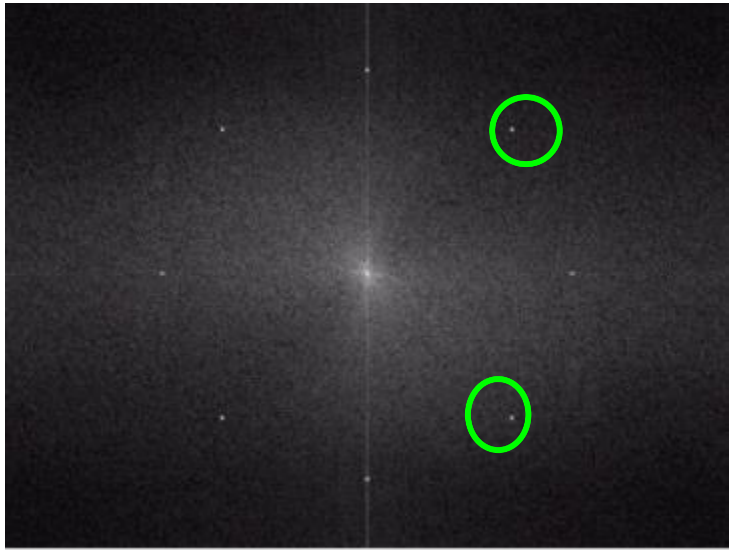


# Periodic Noise

- Periodic noise is spatially dependent
- Can be handled well by frequency domain filtering



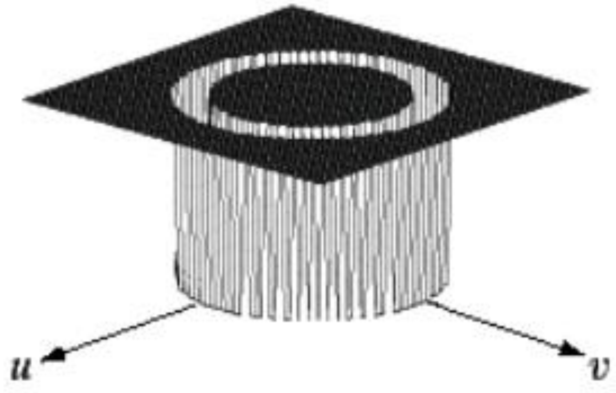
Spatial Image



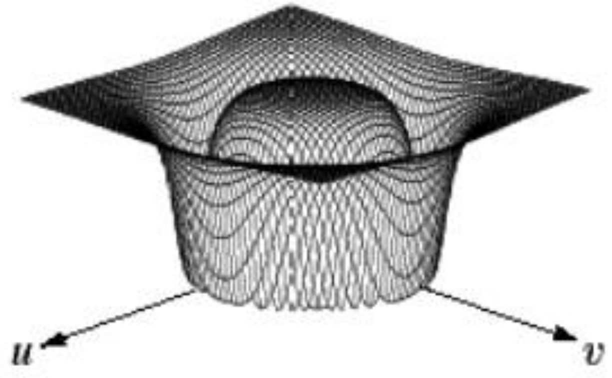
Fourier Transform

Band Reject Filters

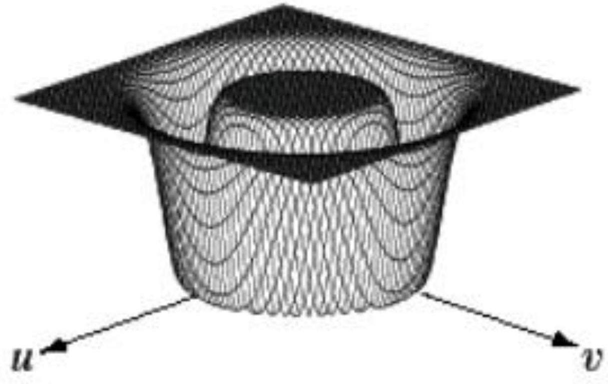
# Band Reject Filters




Ideal Bandreject Filter



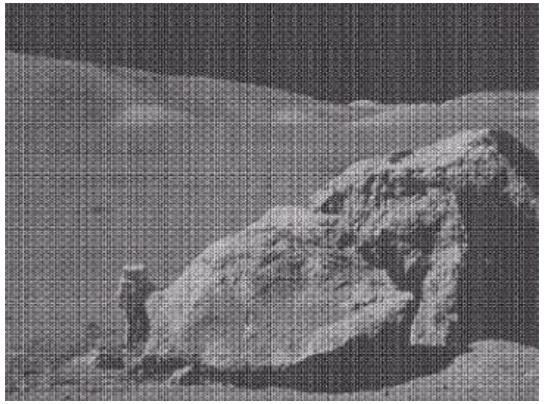

Butterworth Bandreject Filter



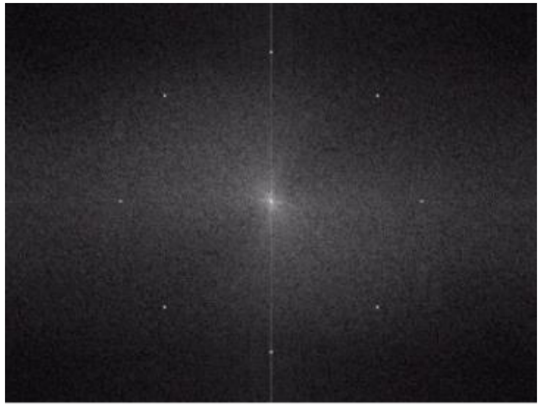

Gaussian Bandreject Filter

# Band Reject Filters (2)

Memfilter da

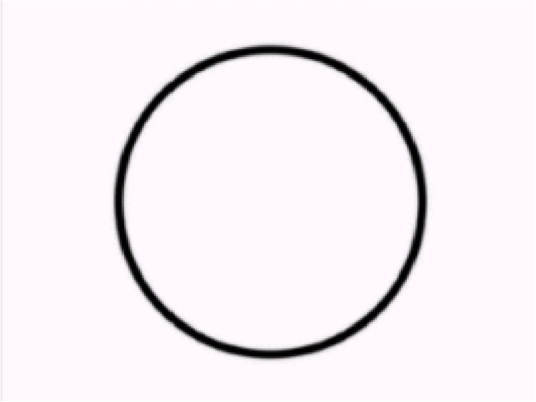


Spatial Image



Fourier Transform

band reject filter untuk mengekstrak citra yang bagus



Band Reject Filter

Restored Image

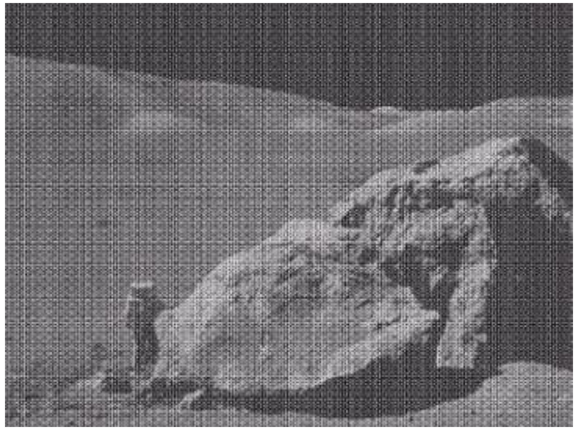
polanya yang akan keluar kalau band pass filter

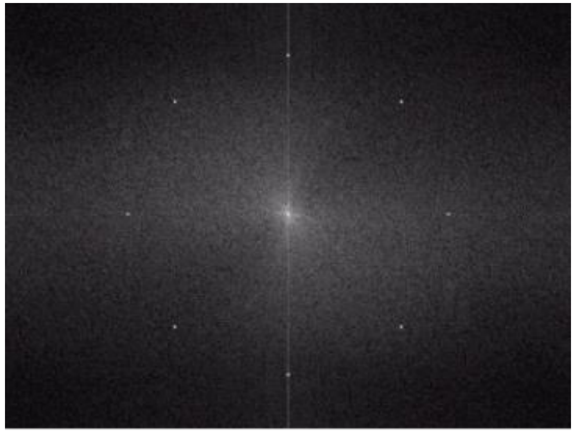
noise bisa dipake untuk pola, jadi kalau next time diambil pakai alat yang sama, maka polanya noise sudah tau dan langsung dihapus tanpa perlu analisa

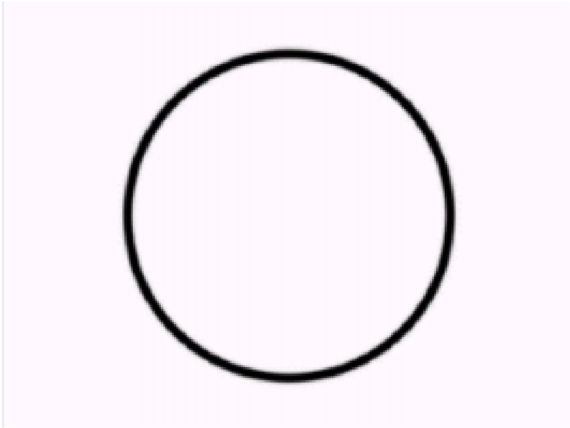
# Band Pass Filters

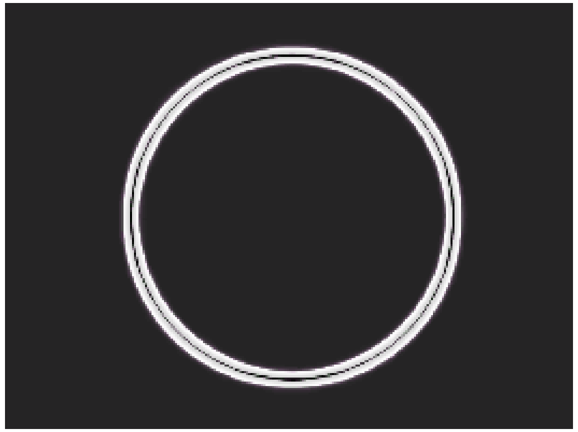
- The opposite of Band Reject Filters

$$H_{BP}(u,v) = 1 - H_{BR}(u,v)$$


Spatial Image

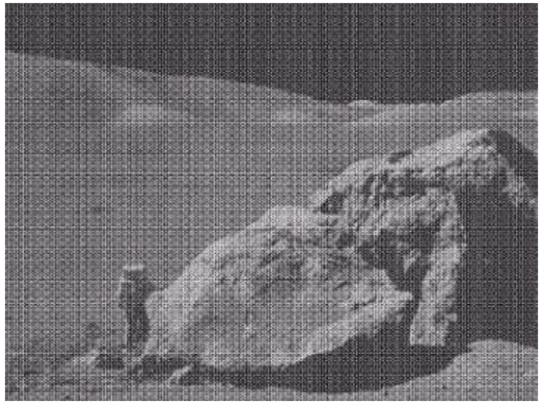

Fourier Transform


Band Reject Filter

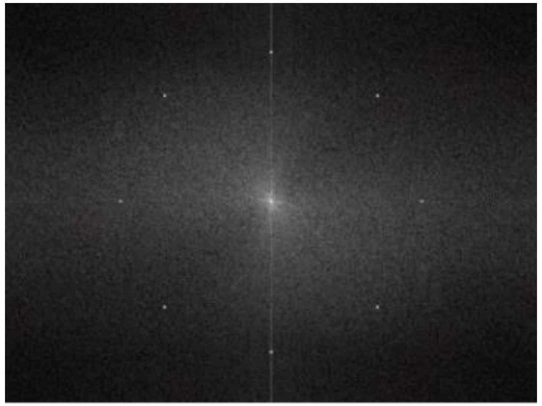

Band Pass Filter

- What will I obtain if I filter the image with the band pass filter?

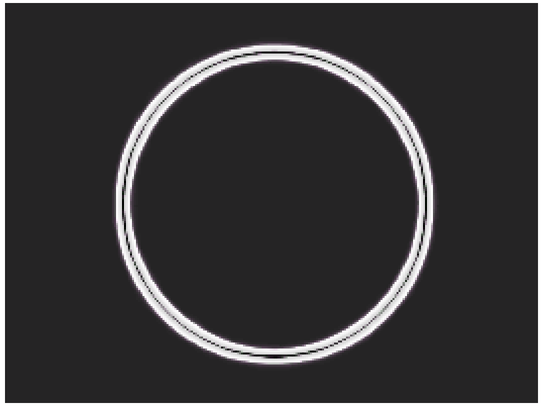
# Band Pass Filters (2)



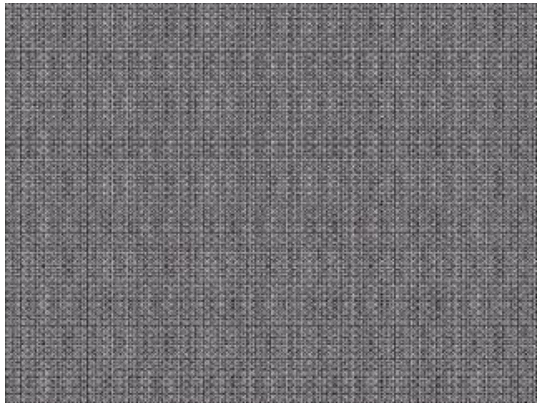
Spatial Image



Fourier Transform



Band Pass Filter



Noise Pattern of Image

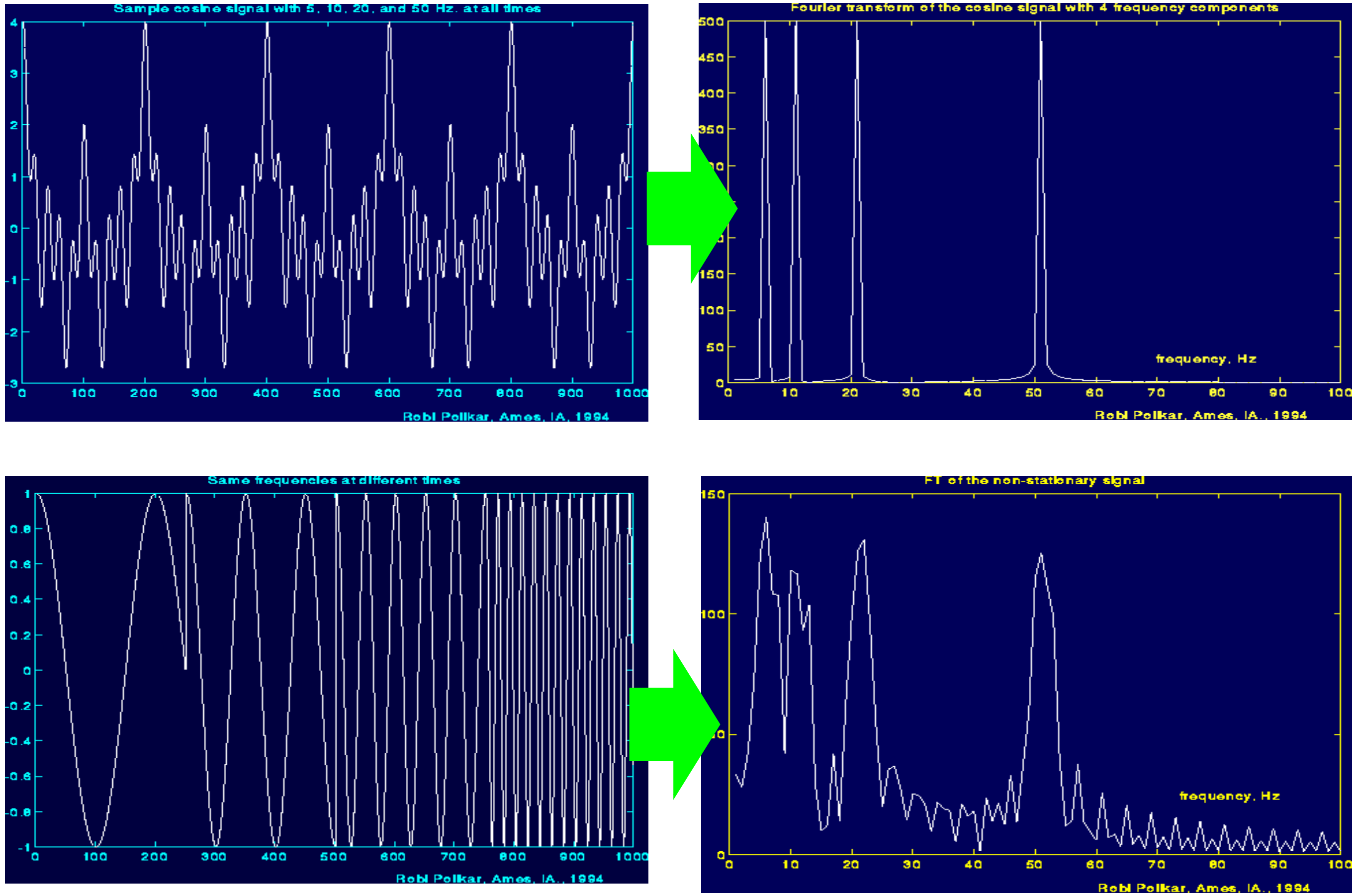
# Disadvantage of Fourier Transform

engga kasih tau kalau frekuensinya muncul dimana (jadi dia cuman tau nih kalo ada frekuensi).

- Wavelet Transform (WT) provides a correction to Fourier Transform (FT).
- Fourier Transform can give information **whether** frequencies exist in a signal but can not give information **where** they occur.
- Fourier transform can give information about **frequency** in a signal, but wavelet transform can give information both **scale and frequency**.
- Fourier transform is based on sin-cos basis which is periodic and continuous in nature, so it is difficult to make a change in specific position (because it will also cause changes in other positions)

# Disadvantage of Fourier Transform (2)

- A signal consists of 4 frequencies (5, 10, 20, and 50) happens at the same time
- A signal consists of 4 frequencies happens in a sequence time
- → The results of FT for both signals are almost the same




(http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html)