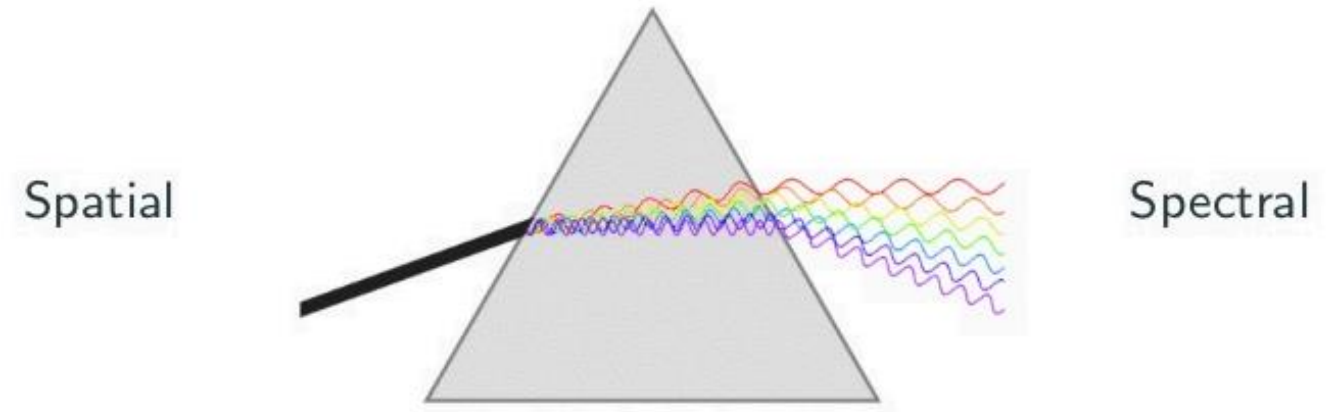
# AI6121 Computer Vision

## **Spectral Image Filtering**

# Contents and Learning Objectives

1. Spectral Representation
2. Discrete Fourier Transform
3. Spectral Filtering
4. Applications



# 1. Spectral Representations

A sine wave $f(t) = a\cos(2\pi u t + \varphi)$ is periodical

$$f(t) = t(t+T) \quad \text{for } T = 1/u \qquad\qquad \text{for all } t \in \mathbb{R}$$

And characterized by

$u$: frequency ($u = 1/T$)

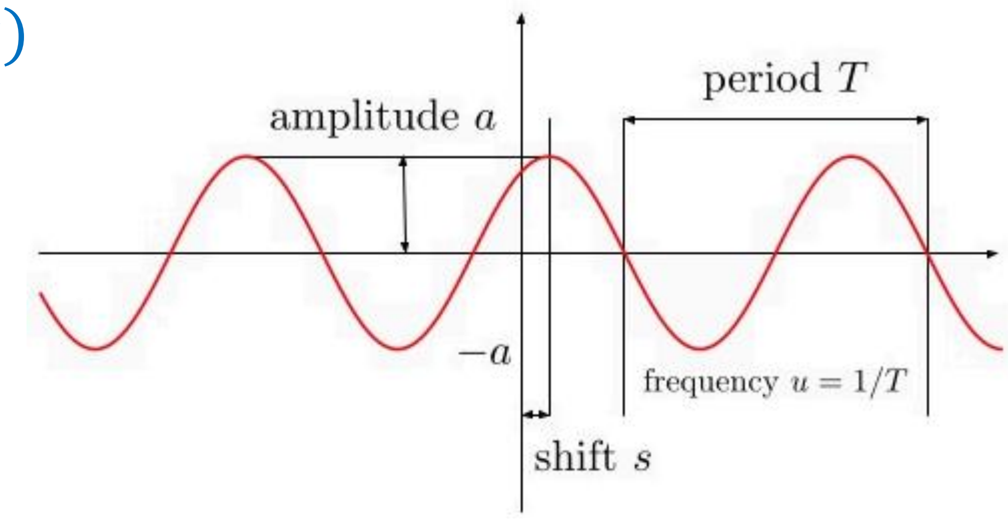
$a$: amplitude

$\varphi$: phase ($\varphi = -2\pi u s$)

Where

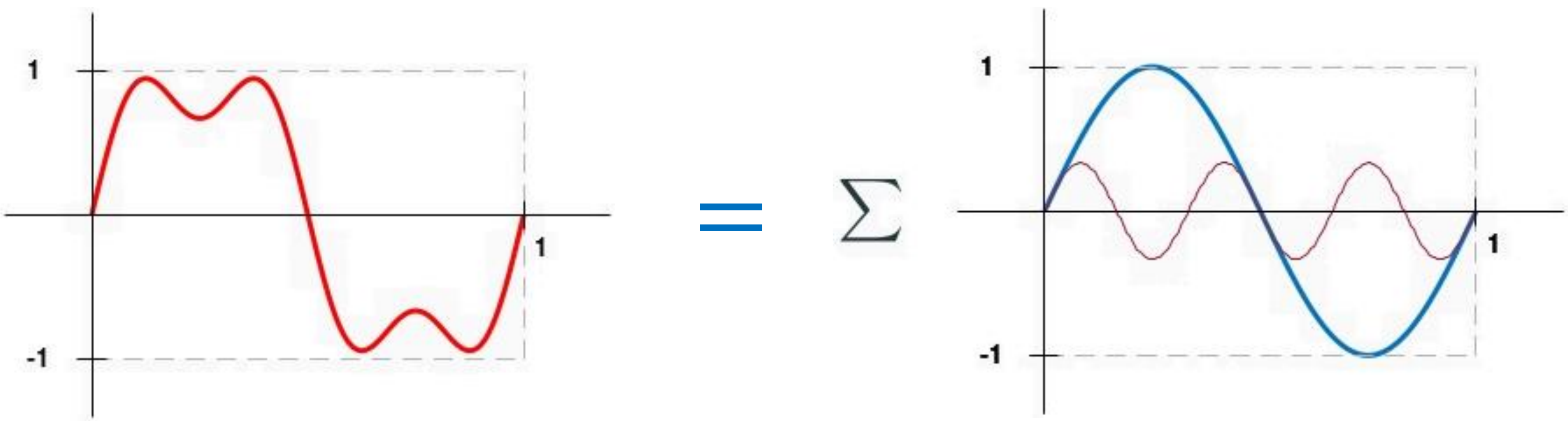
$T$: period

$s$: shift



# 1. Spectral Representations

A complex periodical signal can be approximated by multiple simple periodical sine waves



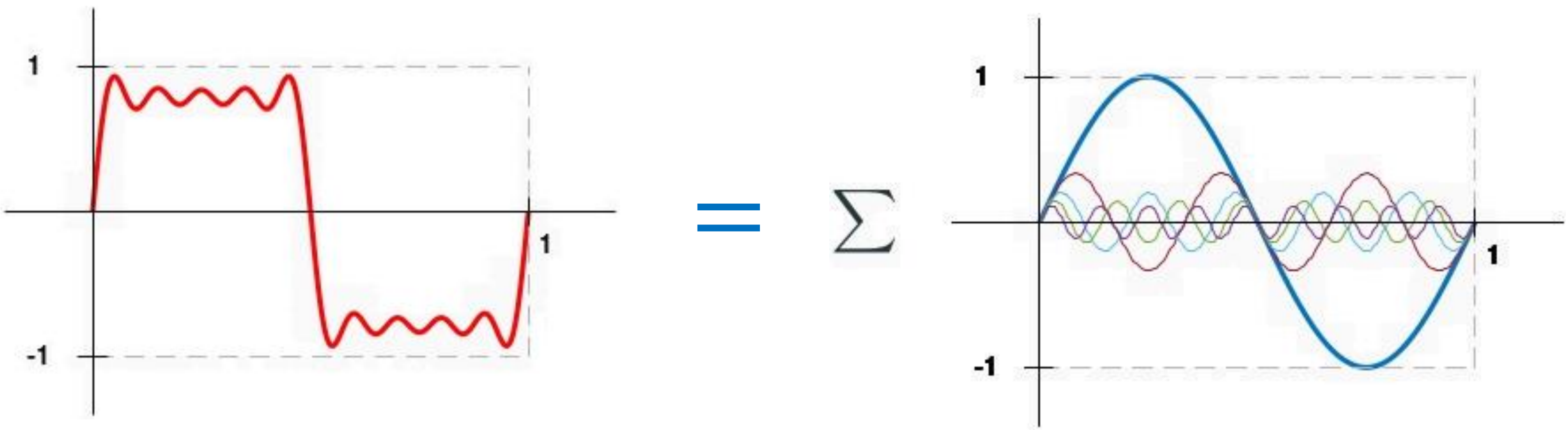

$u_1 = 1, a_1 = 1, \varphi_1 = 3\pi/2$

$u_2 = 3, a_2 = 1/3, \varphi_2 = 3\pi/2$

$$f(t) = a_1 \cos(2\pi u_1 t + \varphi_1) + a_2 \cos(2\pi u_2 t + \varphi_2)$$

# 1. Spectral Representations

## With a more complex periodical signal




$$u_1 = 1, a_1 = 1, \varphi_1 = 3\pi/2$$

$$u_3 = 3, a_3 = 1/3, \varphi_3 = 3\pi/2$$

$$u_5 = 5, a_5 = 1/5, \varphi_5 = 3\pi/2$$

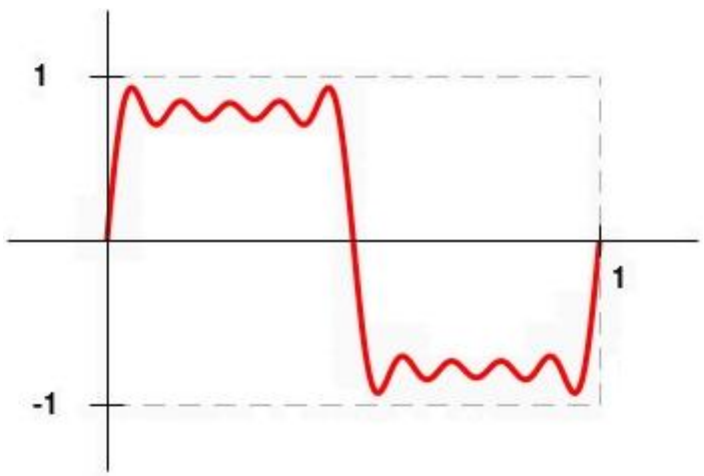
$$u_7 = 7, a_7 = 1/7, \varphi_7 = 3\pi/2$$

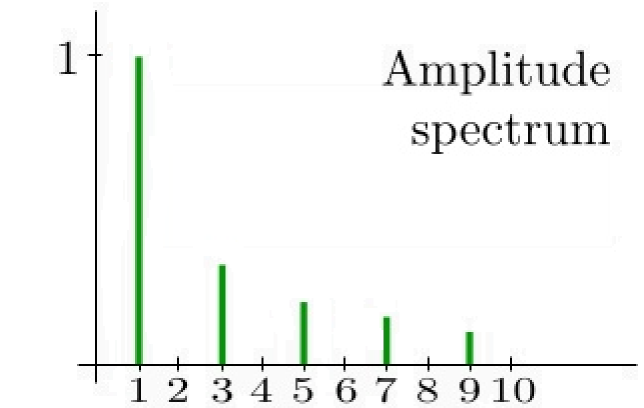
$$u_9 = 9, a_9 = 1/9, \varphi_9 = 3\pi/2$$

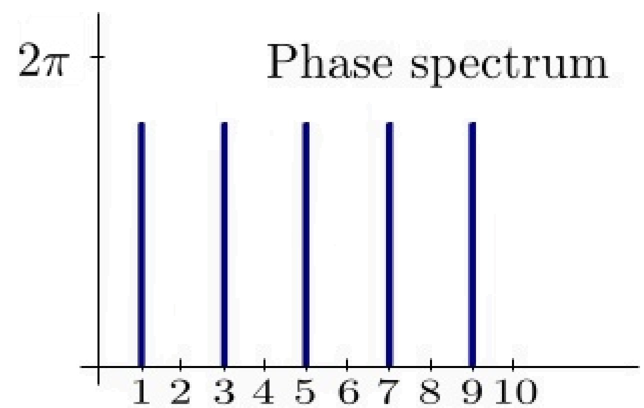
$$f(t) = \sum_{k=1}^{5} a_k \cos(2\pi u_k t + \varphi_k)$$

# 1. Spectral Representations

## With a complex periodical signal




$1 \rightarrow (1, 2\pi/3)$ $\quad$ $3 \rightarrow (1/3, 2\pi/3)$

$5 \rightarrow (1/5, 2\pi/3)$ $\quad$ $7 \rightarrow (1/7, 2\pi/3)$

$9 \rightarrow (1/9, 2\pi/3)$

# 1. Spectral Representations

Let $f$ be a T-periodic function: $f(t) = f(t+T)$ with $u = 1/T$ being the fundamental frequency, we have

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(2\pi u_k t + \varphi_k)$$

The frequency $u_k = u \cdot k$ is called harmonics. The coefficients $(a_k, \varphi_k)$ associated with $u_k$ define $f$.

With Euler's formula $\cos(x) = \frac{e^{ix}+e^{-ix}}{2}$, $f(t)$ can be rewritten by

$$f(t) = \sum_{k=-\infty}^{+\infty} c_k e^{i2\pi ukt}$$

where $a_k(t) = e^{i2\pi ukt}$ is called Fourier atoms which are non-zero and orthogonal. They form an orthogonal basis for T-periodical functions which are called Fourier basis.

# 1. Spectral Representations

$c_k = \frac{1}{2} a_{|k|} e^{\text{sign}(k) i \varphi_{|k|}}$ is Fourier coefficients which encode $a_{|k|}$and $\varphi_k$ that define *f*.

It can be further formulated by

$$c_k = \frac{\langle f, a_k \rangle}{\|a_k\|_2^2} = \frac{1}{T} \int_{-\infty}^{+\infty} f(t) e^{-i2\pi ukt} dt$$

Which gives the Fourier transform

# Summary

1. Frequency, amplitude, phase, and harmonics
2. Fourier basis and Fourier coefficients
3. Function representation in spectral domain

# 2. Discrete Fourier Transform (DFT)

Consider $f$ be discrete signal $f \in \mathbb{R}^n$, and a periodical signal $f_{k+n} = f_k$, it can be characterized by its $n$ harmonics of the form:

$$0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n/2}{2}, \ldots \frac{n-1}{n}$$

The discrete Fourier transform (DFT) is thus given by

$$f_u = \sum_{k=0}^{n-1} f_k e^{-i2\pi\frac{uk}{n}}, \quad u = 0 \ldots n-1$$

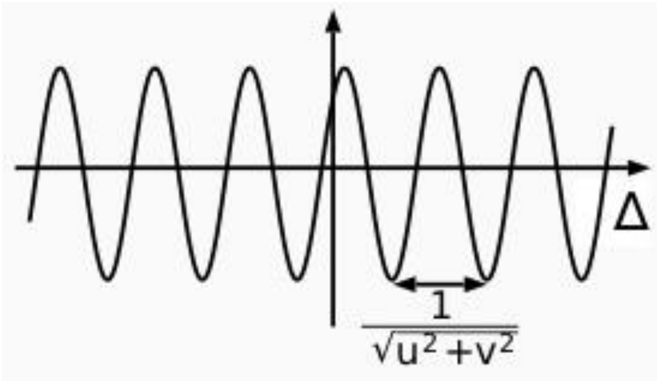
where $f_k$ is the spatial domain signal.

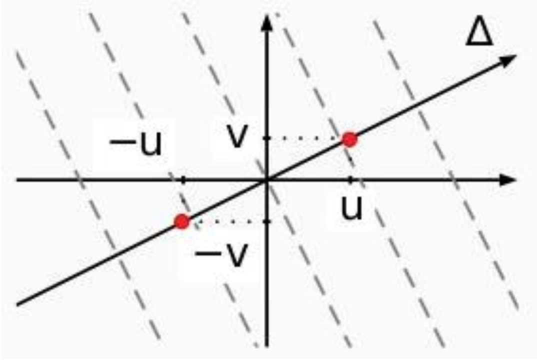
# 2. Discrete Fourier Transform (DFT)

Let $f \in \mathbb{R}^{M\times N}$ be a discrete image, and consider it be periodical $f_{k+M,l+N} = f_{k,l}$, the 2D DFT is given by

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

The exponential term is basis function for each point *F(u, v).* It is a sine curve of frequency $\sqrt{u^2+v^2}$ along a direction defined by (u, v)

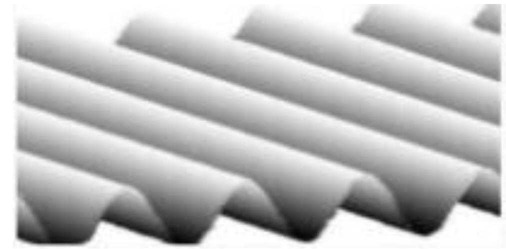





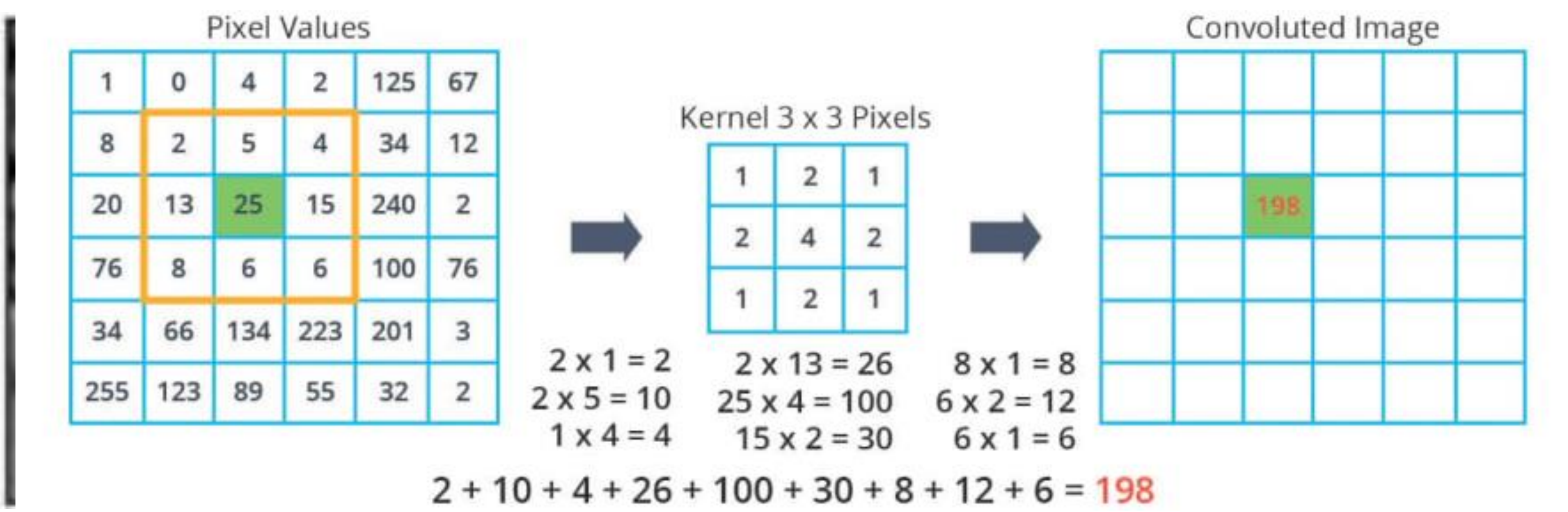
# 2. Discrete Fourier Transform (DFT)

According to the 2D DFT, *F(u, v)* is obtained by multiplying the spatial image with the base function and summing the result.

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

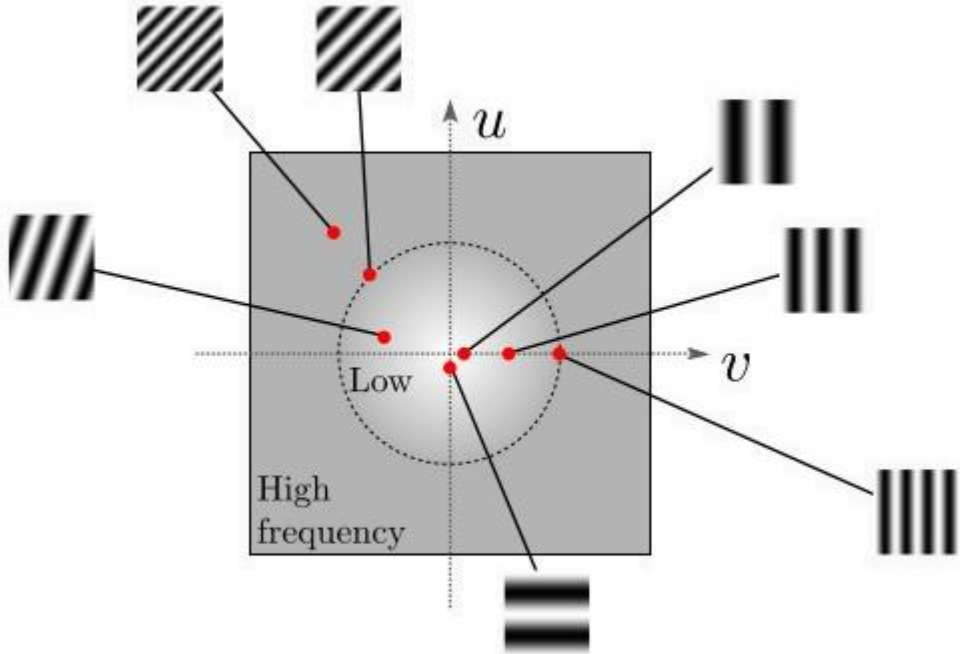


From another view points, F(u, v) is obtained by performing convolution of 2D sine wave over the spatial image.



# 2. Discrete Fourier Transform (DFT)

DFT of an image represents the Fourier coefficients in a 2D grid of the same size as the image.



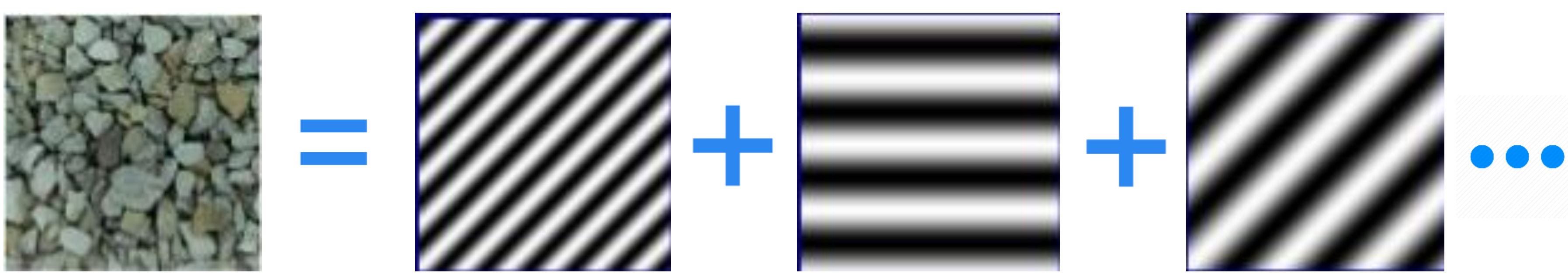

Each point $f_{u,v}$ corresponds to a 2D sine wave with frequency $|f_{u,v}| = \sqrt{u^2 + v^2}$ with a direction decided by $(u, v)$. The center has low frequency and periphery has high frequency.

# 2. Discrete Fourier Transform (DFT)

- DFT is the sampled Fourier Transform with a set of samples that is large enough to describe the spatial domain image. The number of frequencies is equal to the number of pixels in the spatial image.
- The spatial domain image can be recovered via inverse DFT by:

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{2\pi j(\frac{ux}{M}+\frac{vy}{N})}$$

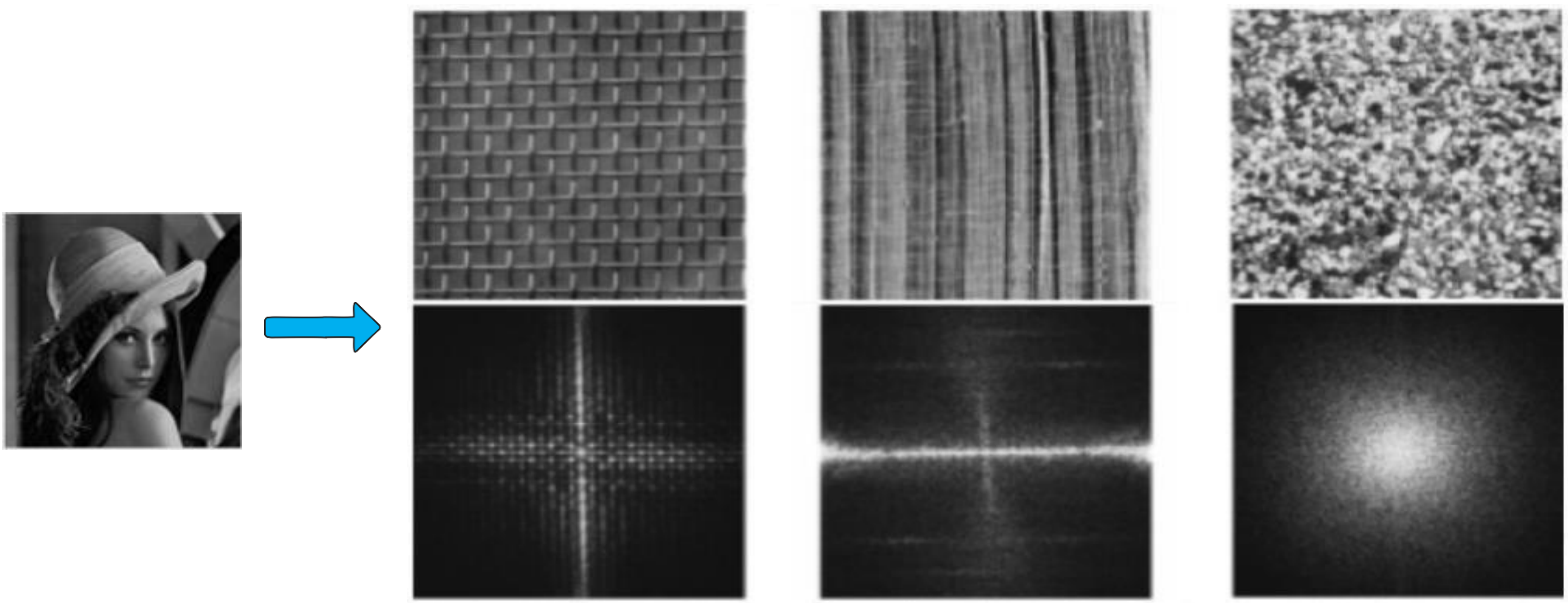
- The spatial image is thus like a weighted sum of sine curves



$$f_{u_1,v_1} \cdot e^{i2\pi(\frac{u_1 k}{M}+\frac{V_1 l}{N})} \qquad f_{u_1,v_2} \cdot e^{i2\pi(\frac{u_1 k}{M}+\frac{V_2 l}{N})} \qquad f_{u_i,v_j} \cdot e^{i2\pi(\frac{u_i k}{M}+\frac{V_j l}{N})}$$

# 2. Discrete Fourier Transform (DFT)

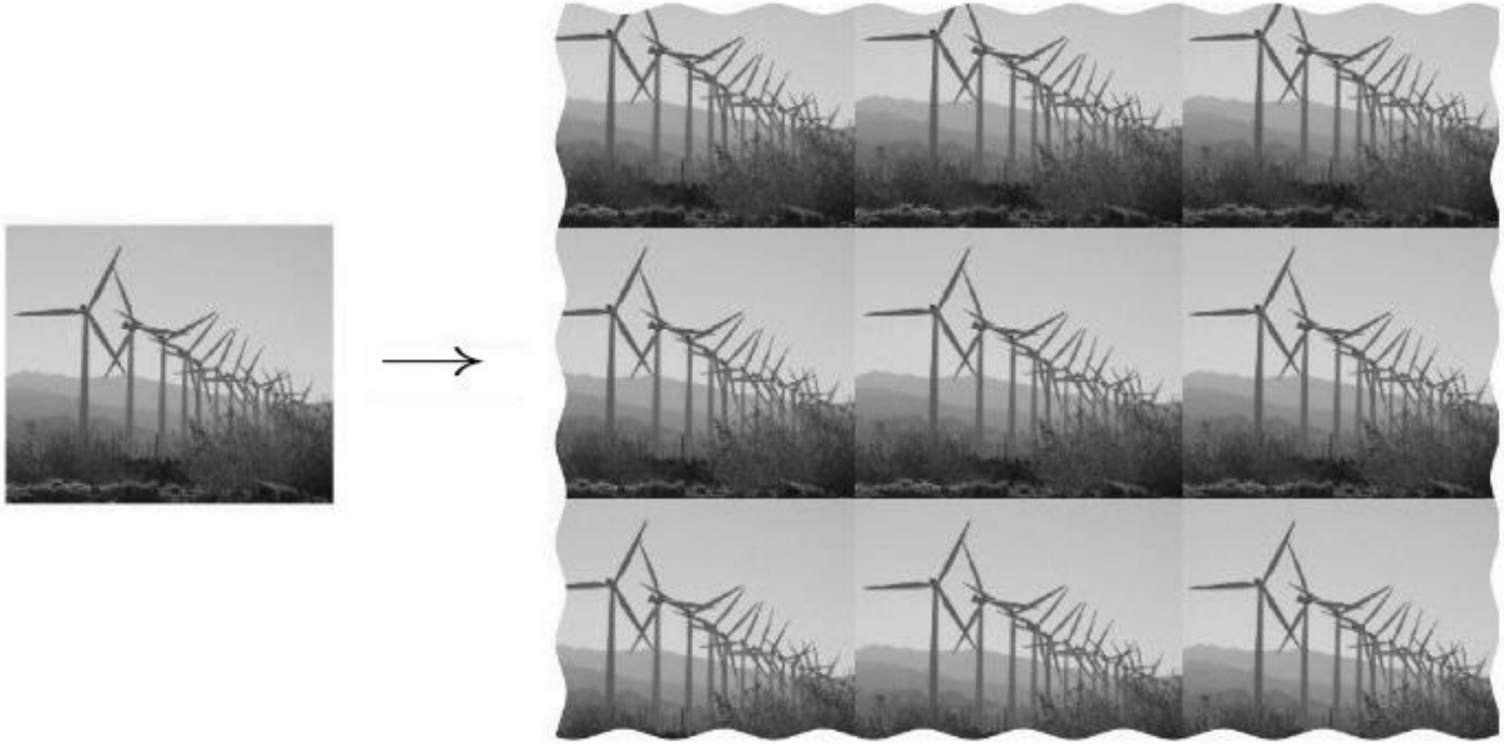
- The basis function in DFT is a complex number as $e^{i\varphi} = \cos(\varphi) + i \cdot sin(\varphi)$. The DFT has two parts including an amplitude and a phase.



- The amplitude highlights the direction of spatial patterns. Edges are represented by all harmonics in its orthogonal direction.
- The major information is contained in the amplitude, though phase also encodes a large amount of information.

# 2. Discrete Fourier Transform (DFT)

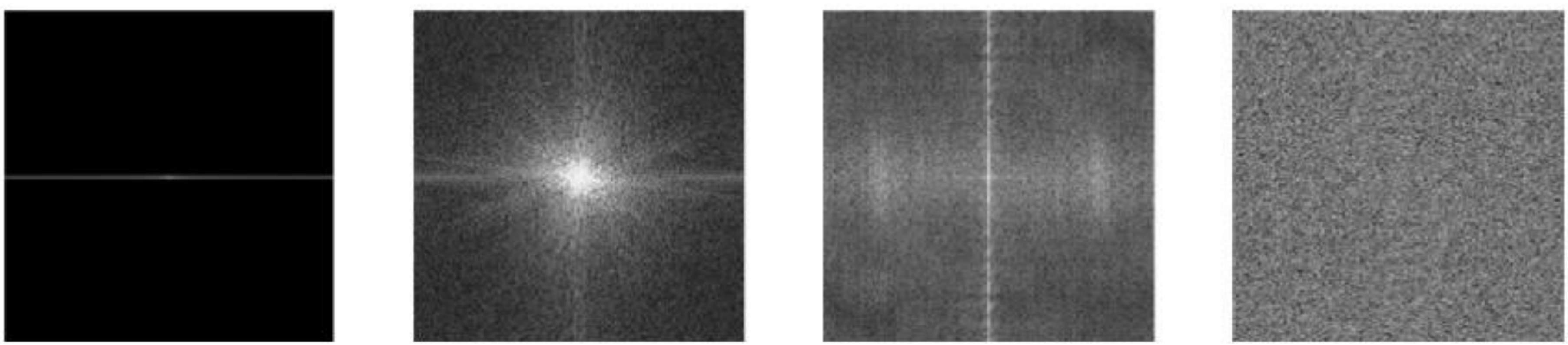
- The amplitude usually has strong vertical and horizontal edges. They are formed as we assume images are periodical, where image borders often create strong edges which cause the strong vertical and horizontal direction in amplitude map.



# 2. Discrete Fourier Transform (DFT)

## Which amplitude map is derived from which image?

A B C D



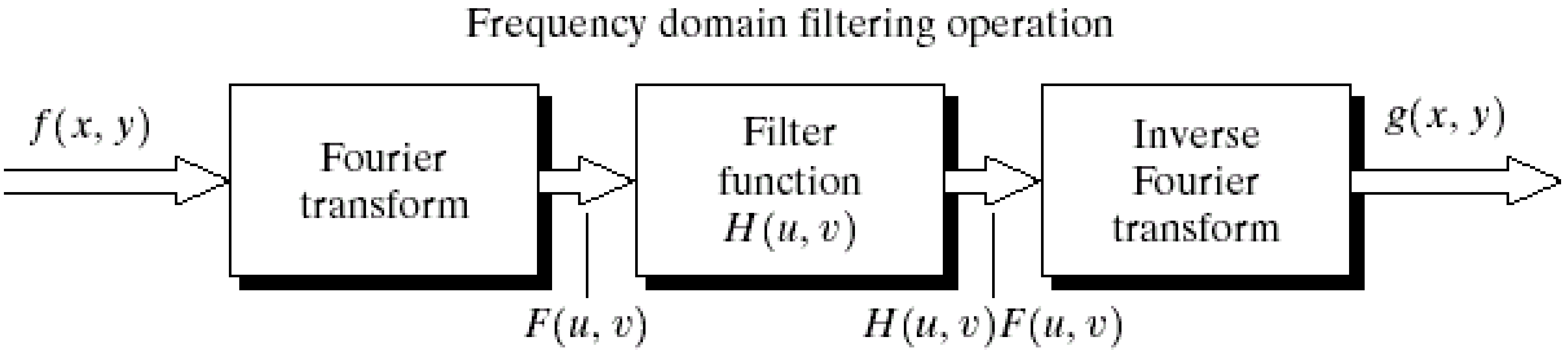
1 2 3 4

# Summary

1. How to get 2D DFT from a spatial image
2. How is a spatial image represented by spectrum
3. DFT amplitude and phase

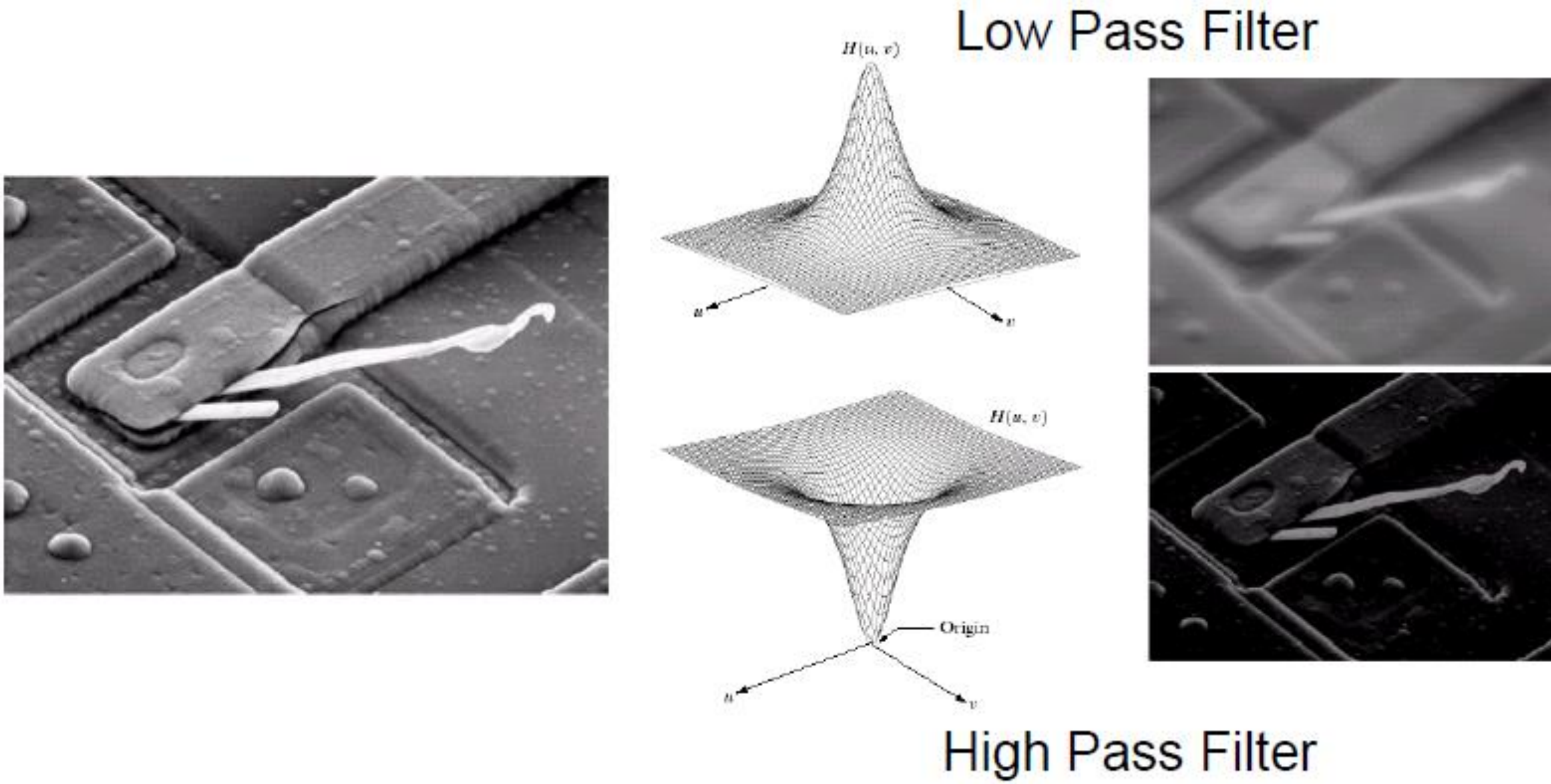
# 3. Spectral Filtering

- With DFT, we can filter spatial images by keeping low-frequency signals (low pass filter), high-frequency signals (high pass filter), and frequency-band signals (band pass filter).
- The filtering consists of three major steps:
  1. Compute $f(u, v)$, i.e. the DFT of a spatial image $f(x, y)$
  2. Multiply $|f(u, v)|$ by a filter function $H(u, v)$
  3. Compute the inverse DFT to get the filtered image $g(x, y)$



# 3. Discrete Fourier Transform (DFT)

The figure below illustrates how low-pass filter and high-pass filter can smooth and sharpen spatial domain images.



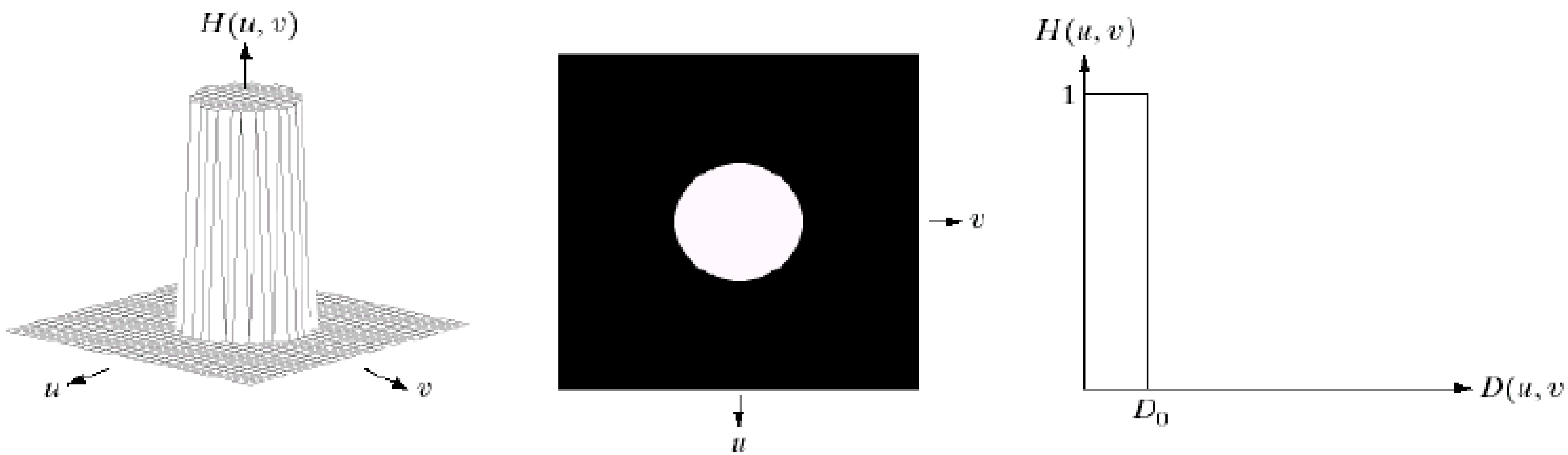
# 3. Discrete Fourier Transform (DFT)

- Ideal low pass filter directly cut off all high frequency components that are beyond a specified distance $D_0$ from the origin:

$$H(u,v) = \begin{cases} 1 \;\; if \; D(u,v) \leq D_0 \\ 0 \;\; if \, D(u,v) > D_0 \end{cases}$$

- where D(u, v) is defined by

$$D(u,v) = [(u - M/2)^2 + (u - N/2)^2]^{1/2}$$

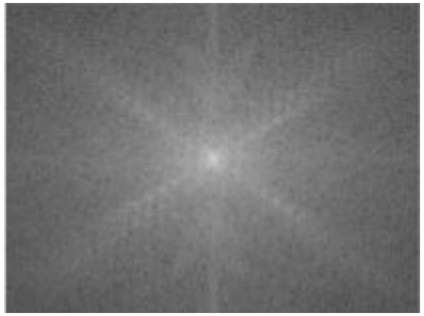


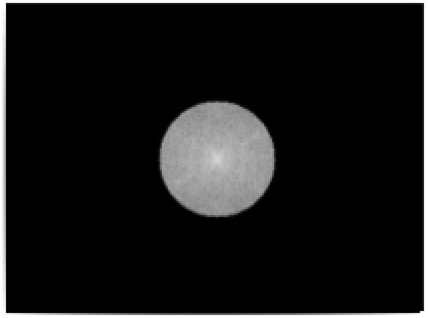
# 3. Discrete Fourier Transform (DFT)

## Python demo – Low-pass filter

```
import numpy.fft as nf
import imagetools as im

f      = plt.imread('butterfly.png')
n1, n2 = f.shape
tf     = nf.fft2(f, axes=(0, 1))
a      = np.abs(tf)
phi    = np.angle(tf)
u, v   = im.fftgrid(n1, n2)
dist2  = u**2 + v**2
mask   = dist2 <= r**2
ap     = mask * a
tfp    = ap * np.exp(1j * phi)
fp     = np.real(nf.ifft2(tfp, axes=(0, 1)))
```

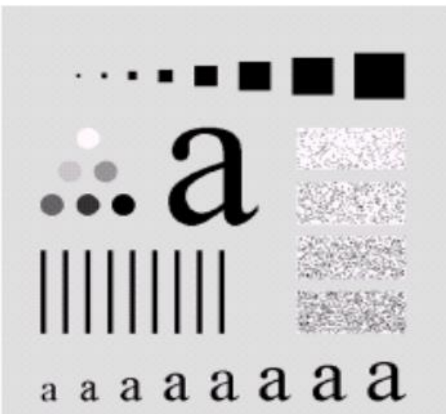




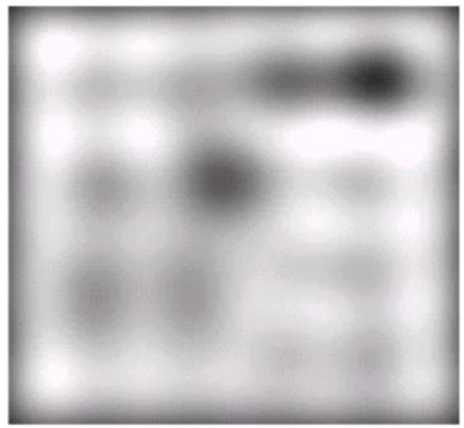
# 3. Discrete Fourier Transform (DFT)

- Images filtered by the ideal low pass filter often have undesired ringing effects, largely due to the sudden stop and pass:

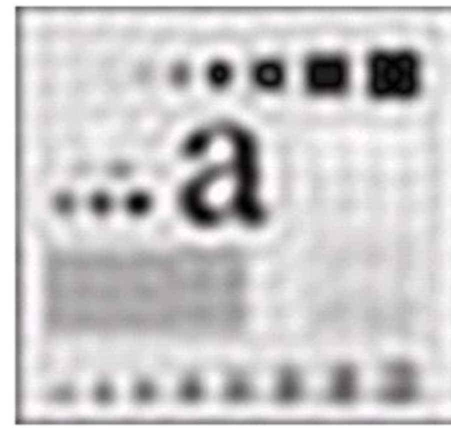
**Original image**



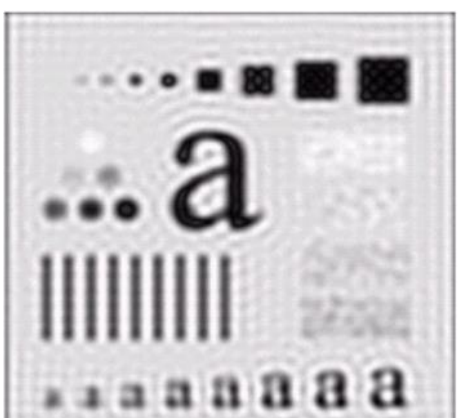
**$D_0 = 5$**



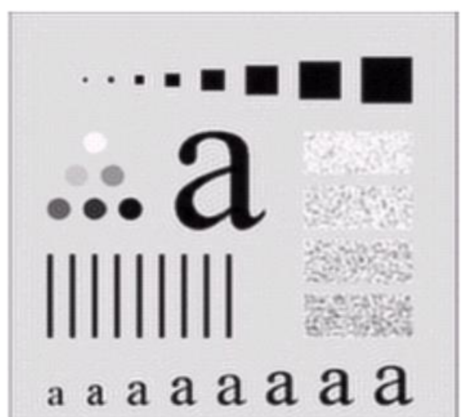
**$D_0 = 15$**



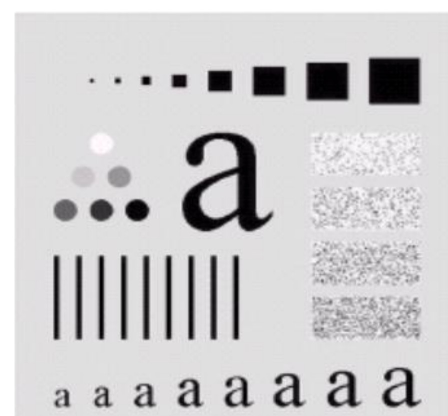
**$D_0 = 30$**



**$D_0 = 80$**



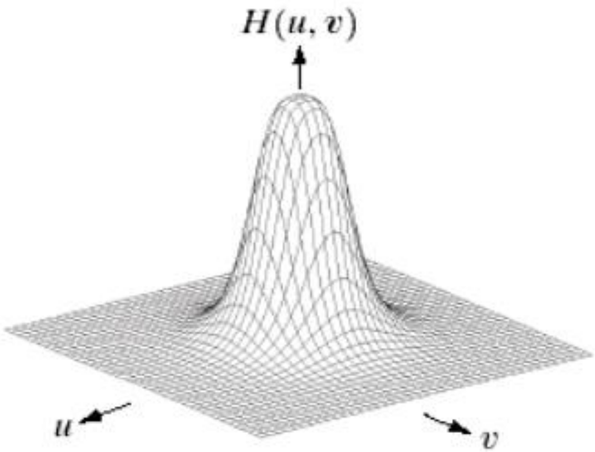
**$D_0 = 230$**

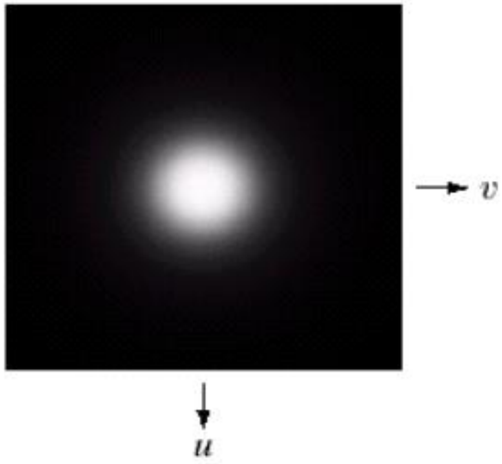


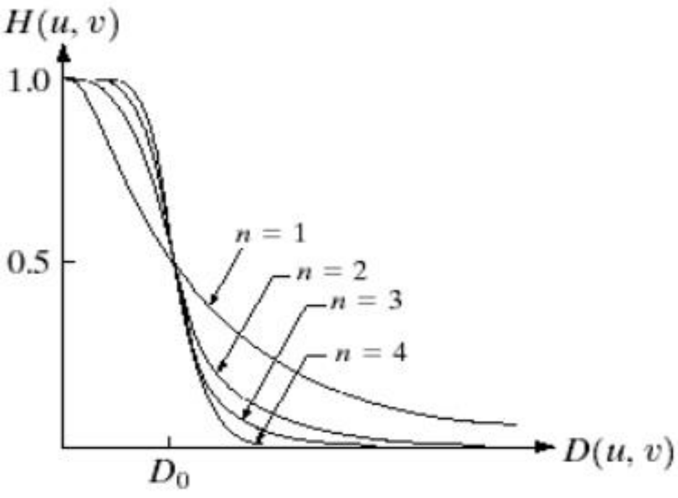
# 3. Discrete Fourier Transform (DFT)

- Idea low pass filter has ringing effects which can be mitigated via certain transfer function that cut frequencies smoothly.
- Butterworth low pass filter employs a transfer function that defines the cutoff frequency at distance $D_0$ from the origin by:

$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}$$



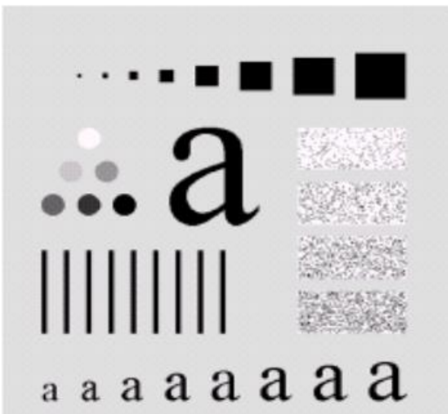



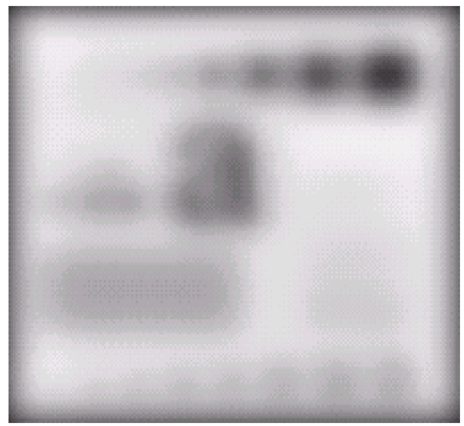
# 3. Discrete Fourier Transform (DFT)

- The Butterworth low pass filter can mitigate the ringing effects effectively as illustrated.

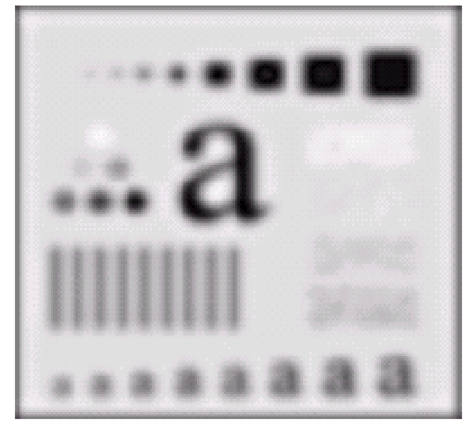
**Original image**



**$D_0 = 5$**

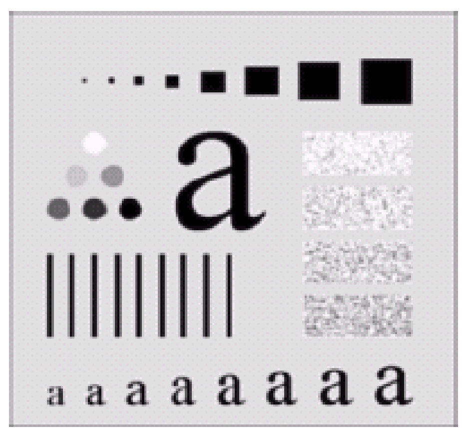


**$D_0 = 15$**

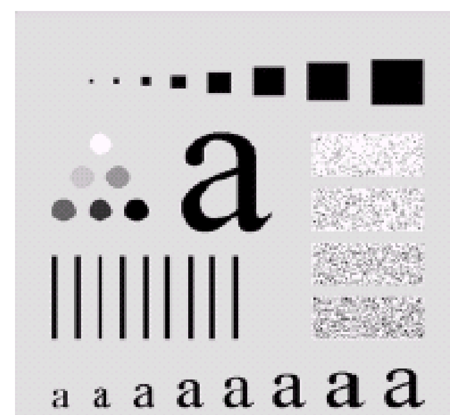


**$D_0 = 30$**

**$D_0 = 80$**



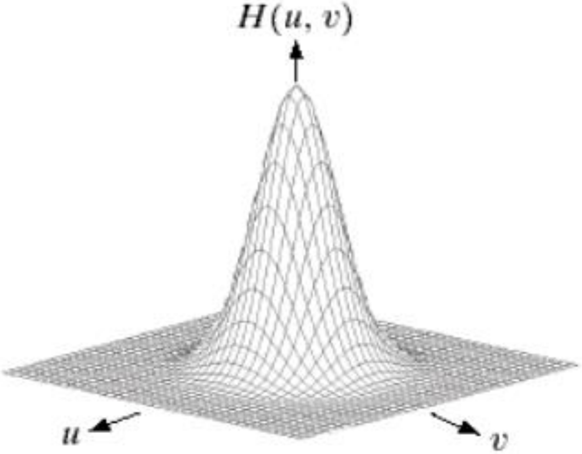
**$D_0 = 230$**

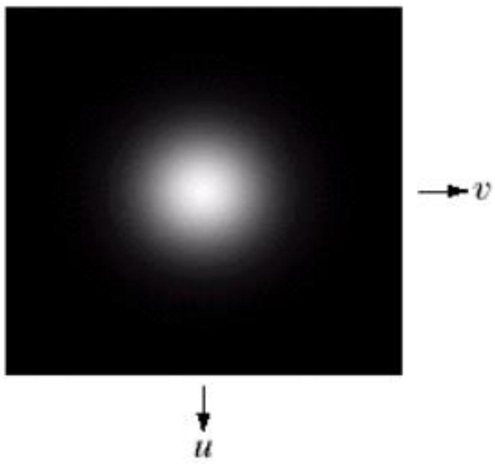


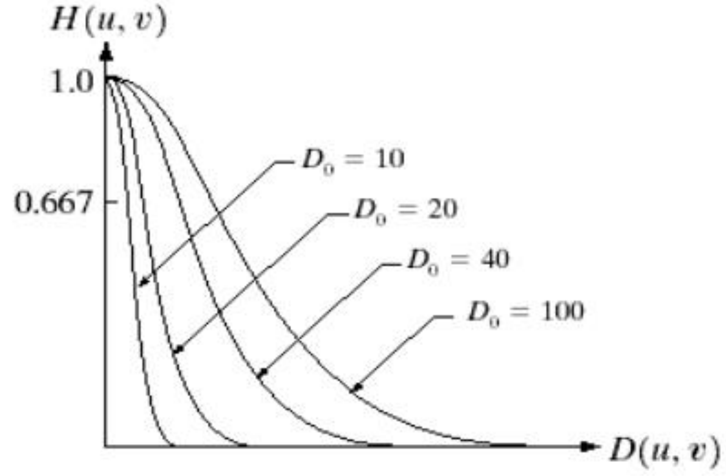
# 3. Discrete Fourier Transform (DFT)

- Gaussian low pass filter employs a transfer function that defines the cutoff frequency at distance $D_0$ from the origin by:

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$



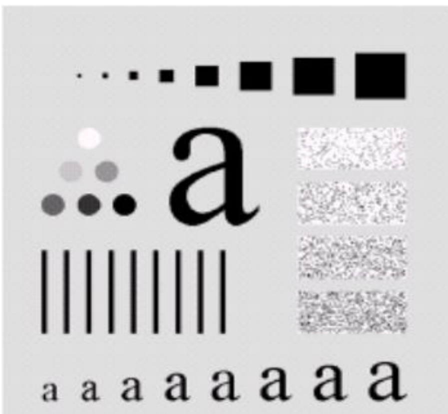



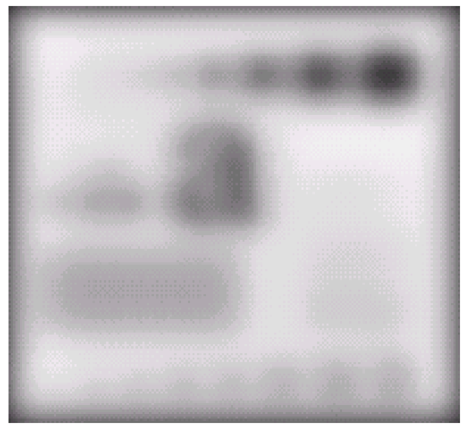
# 3. Discrete Fourier Transform (DFT)

- The Gaussian low pass filter can also mitigate the ringing effects effectively as illustrated.

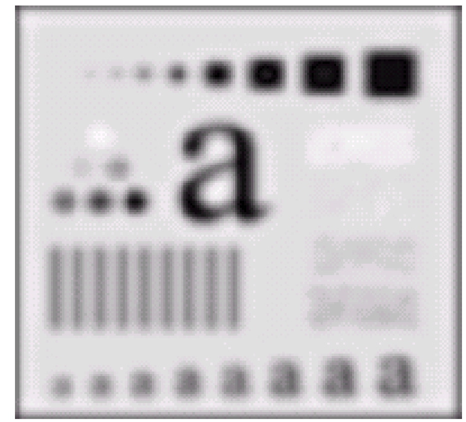
**Original image**



**$D_0 = 5$**

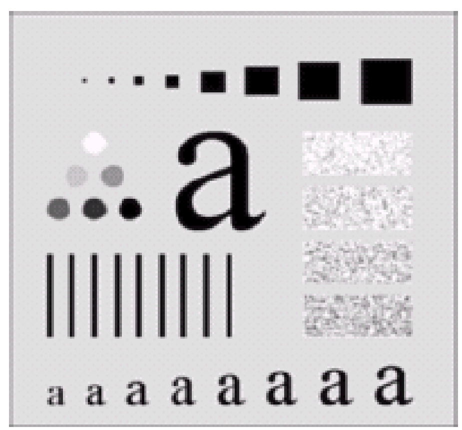


**$D_0 = 15$**

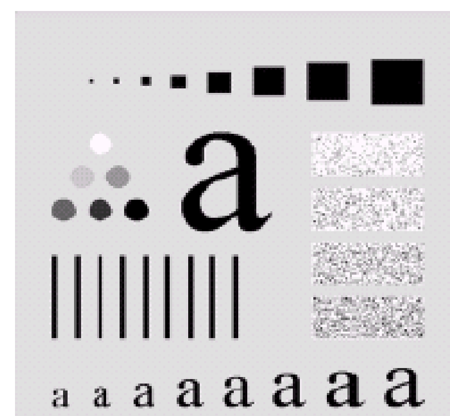


**$D_0 = 30$**

**$D_0 = 80$**

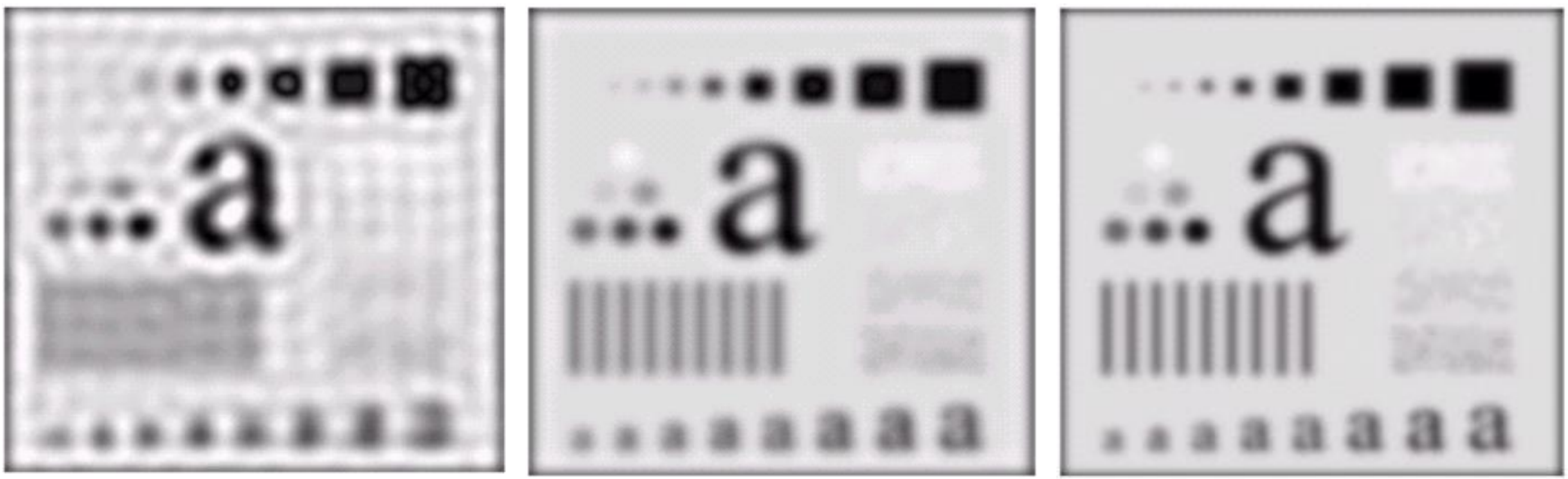


**$D_0 = 230$**



# 3. Discrete Fourier Transform (DFT)

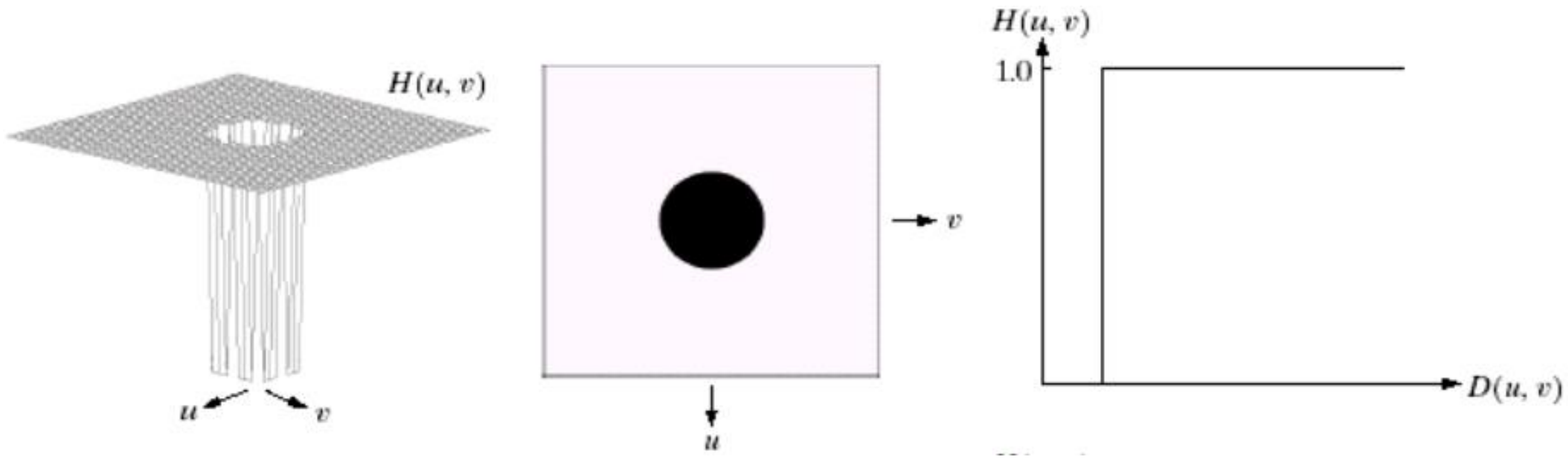
The figure below compares the filtering by using an ideal low pass filter, Butterworth low pass filter, and Gaussian low pass filter (from left to right) while the cutoff frequency is the same with $D_0$ = 15.



# 3. Discrete Fourier Transform (DFT)

- High pass filters pass high frequencies and drop low ones which are the reverse of low pass filters $H_{hp}(u,v) = 1 - H_{lp}(u,v)$.
- Ideal high pass filter directly cut off all low frequency components that are within a specified distance $D_0$ from the origin:

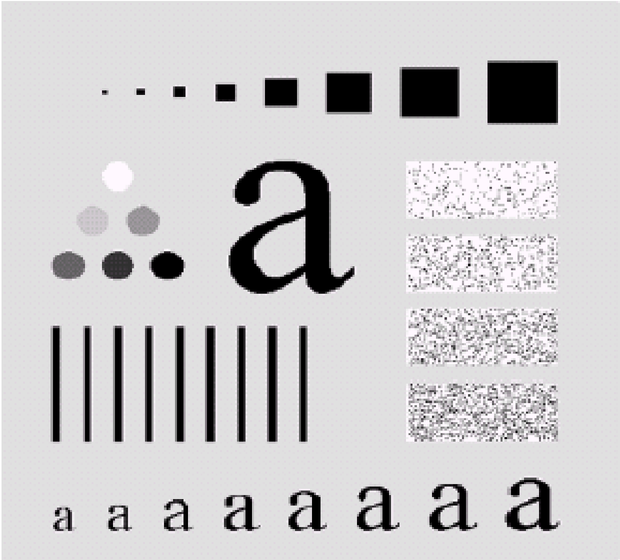
$$H(u,v) = \begin{cases} 1 \ \ if \ D(u,v) > D_0 \\ 0 \ \ if D(u,v) \leq D_0 \end{cases}$$



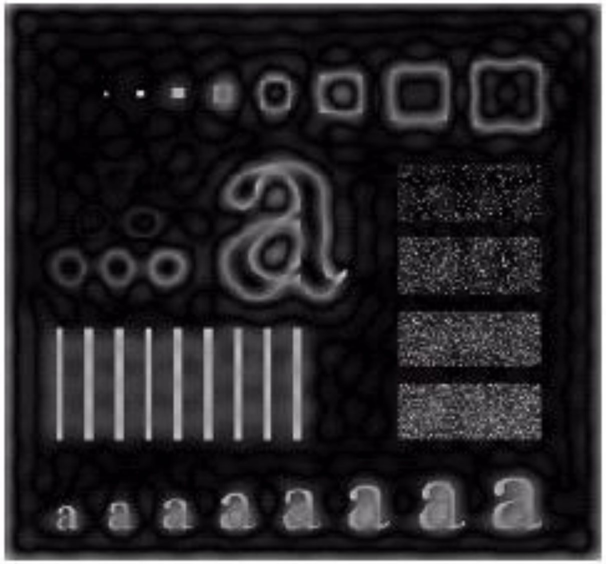
# 3. Discrete Fourier Transform (DFT)

- Image edges and fine details are associated with high frequency components which can be extracted by high pass filters.
- Similar to low pass filters, ideal high pass filters also produce the undesired ringing effects as illustrated.

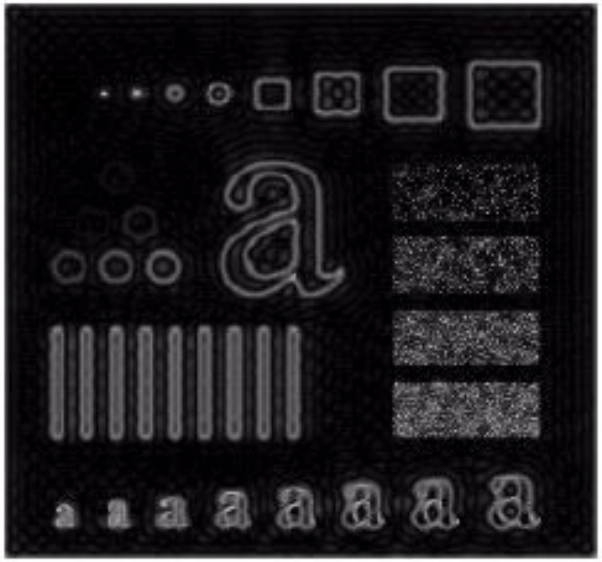
**Original image**




**$D_0$ = 15**



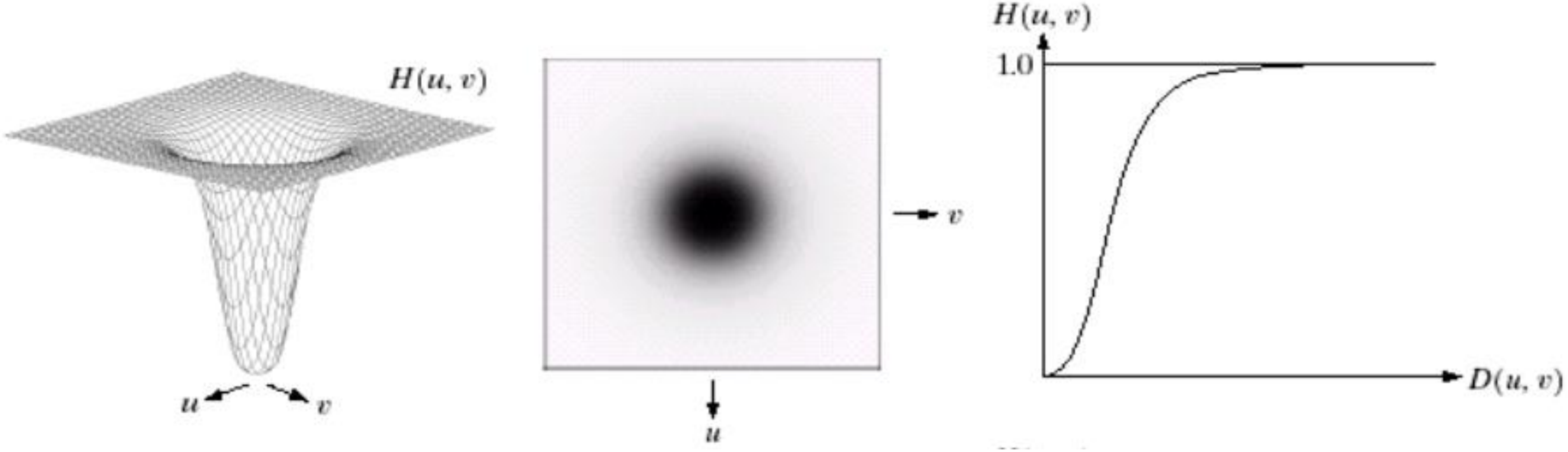
**$D_0$ = 30**



# 3. Discrete Fourier Transform (DFT)

- The ringing effects can be similarly mitigated via certain transfer function that cut frequencies smoothly.
- Butterworth high pass filter employs a transfer function that defines the cutoff frequency at distance $D_0$ from the origin by:

$$H(u,v) = \frac{1}{1 + [D_0/D(u,v)]^{2n}}$$

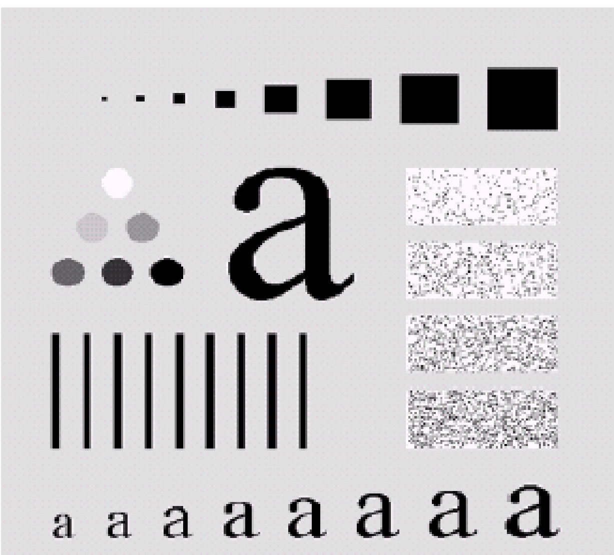


# 3. Discrete Fourier Transform (DFT)

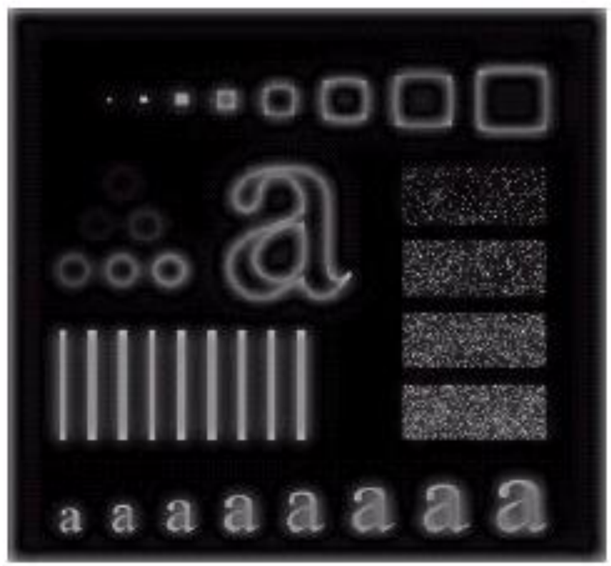
Image edges and fine details are associated with high frequency components which can be extracted by high pass filters as illustrated.

Similar to low pass filters, the Butterworth high pass filter help effectively mitigate the ring effects.

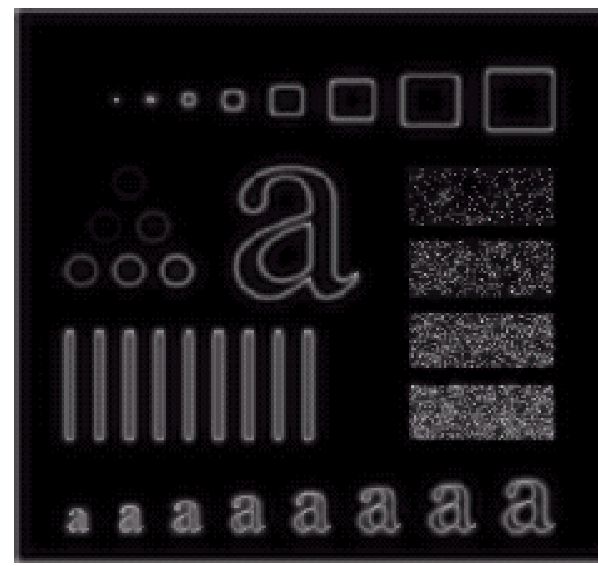
**Original image**



**$D_0$ = 15**



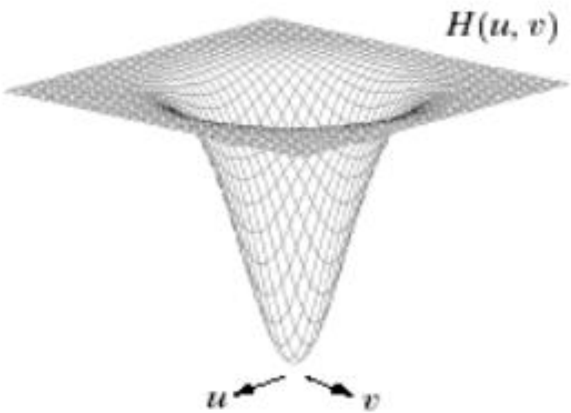
**$D_0$ = 30**

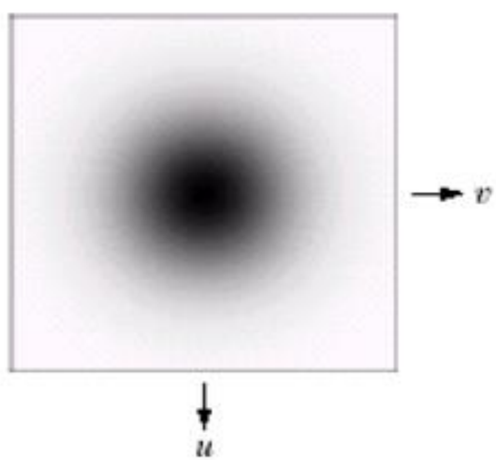


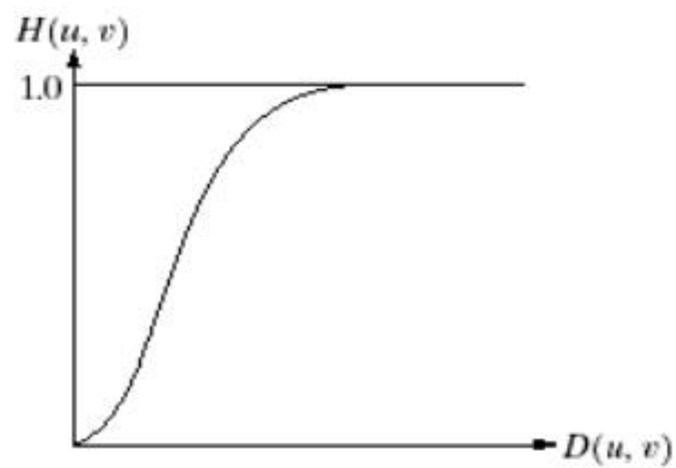
# 2. Discrete Fourier Transform (DFT)

- Gaussian high pass filter employs a transfer function that defines the cutoff frequency at distance $D_0$ from the origin by:

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$



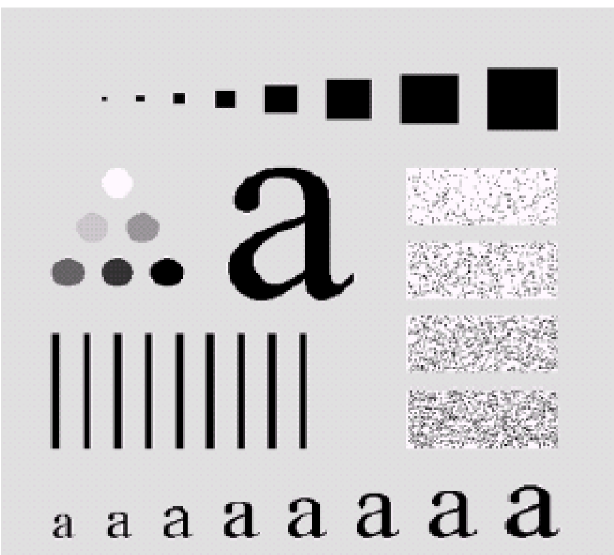



# 3. Discrete Fourier Transform (DFT)

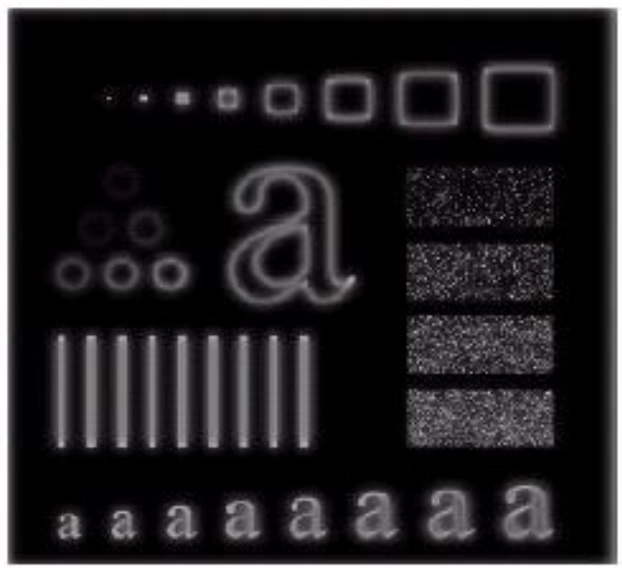
Image edges and fine details are associated with high frequency components which can be extracted by high pass filters as illustrated.

Similar to low pass filters, the Gaussian high pass filter help effectively mitigate the ring effects.

**Original image**



**$D_0$ = 15**



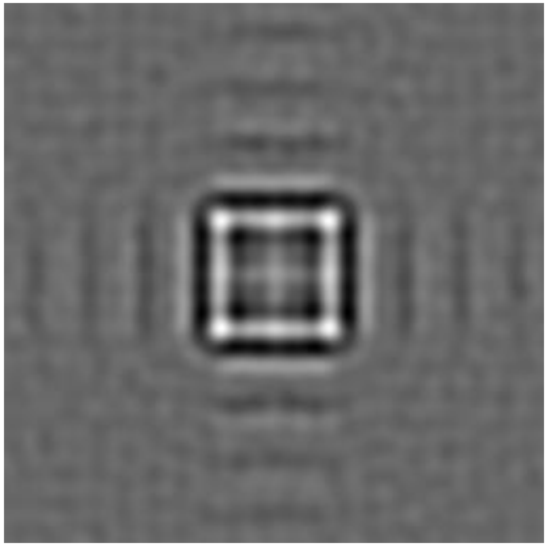
**$D_0$ = 30**

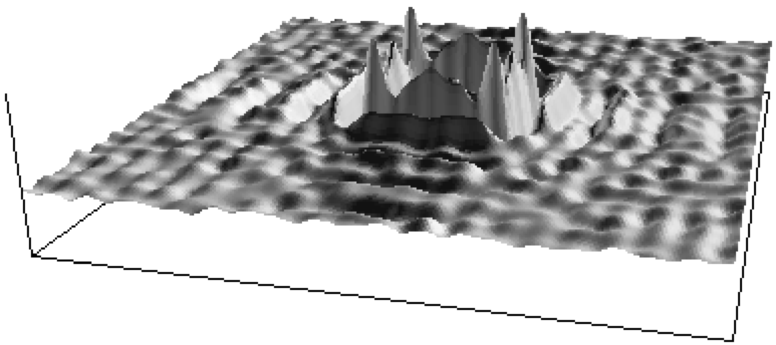
# 3. Discrete Fourier Transform (DFT)

- The spatial domain image can be recovered via inverse DFT by:

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{2\pi j(\frac{ux}{M}+\frac{vy}{N})}$$

- With DFT, we can filter images by keeping low-frequency signals (low pass filter), high-frequency signals (high pass filter), and frequency-band signals (band pass filter).



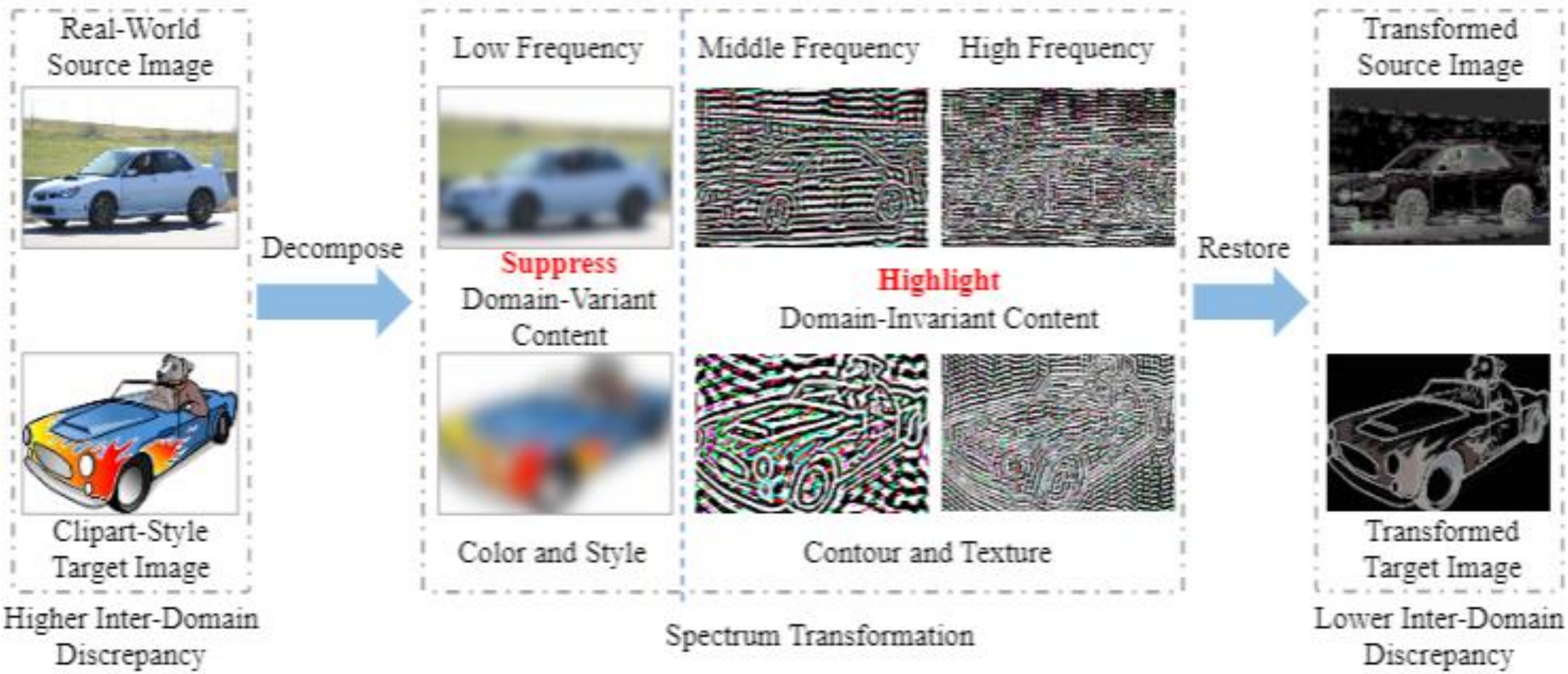


# 4. Applications - Domain Adaptation

- Unsupervised domain adaptation (UDA) aims to learn a well-performed model in an unlabelled target domain by leveraging labelled data from one or multiple related source domains. Different from DG, it accesses (unlabelled) target data during training.
- UDA has been tackled in three typical approaches, namely, image-to-image translation in the input space for reducing domain gaps, adversarial learning for aligning source and target representations in the feature space, and self-training that predicts pseudo labels and re-trains models iteratively.
- As in DG, image-to-image translation tends to undesirably change image structures and semantics which are domain-invariant and good to kept unchanged for UDA.

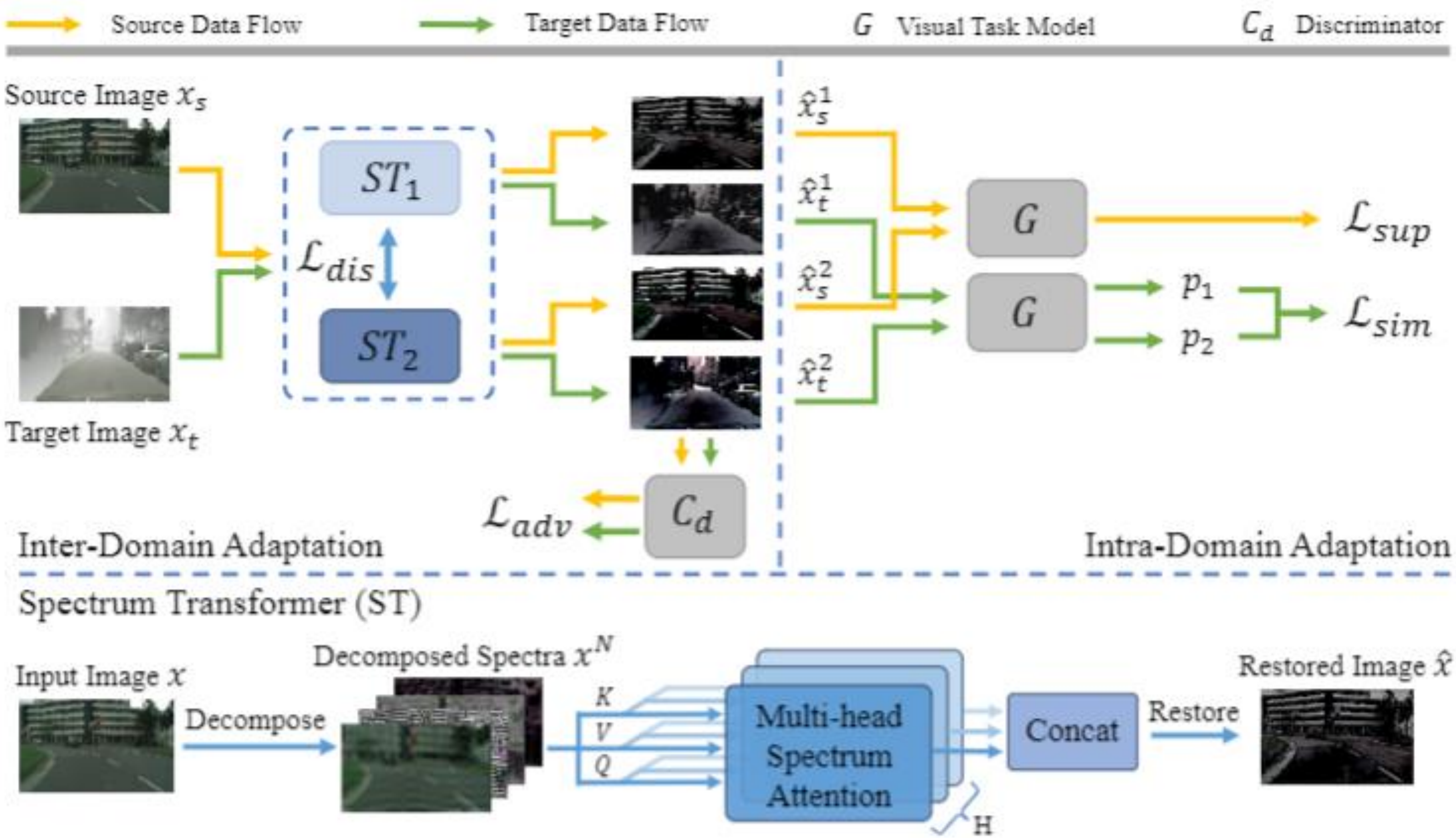
# 4. Applications - Domain Adaptation

We design a spectrum transformer that maps source and target images into spectral space and learns to enhance domain-invariant spectra while suppressing domain-variant spectra simultaneously.



# 4. Applications - Domain Adaptation

The framework shows how the proposed spectrum transformer and spectral learning help mitigate domain gaps and learning more comprehensive and robust representations.



# 4. Applications - Domain Adaptation

The spectrum transformer can learn to weight up domain-invariant FCs and weigh down domain-variant FCs which mitigates inter-domain discrepancy effectively.

# 4. Applications - Domain Adaptation

The spectrum transformer is generic and can work for different domain adaptive tasks such as detection, segmentation, classification, etc. The table shows object detection for Cityscapes →Foggy Cityscapes

| Method | Backbone | person | rider | car | truck | bus | train | mcycle | bicycle | mAP |
|---|---|---|---|---|---|---|---|---|---|---|
| DETR [82] (Baseline) | ResNet-50 | 43.7 | 38.0 | 57.2 | 15.2 | 34.7 | 14.4 | 26.1 | 42.4 | 34.0 |
| DAF [12] | ResNet-50 | 49.4 | 49.7 | 62.1 | 23.6 | 43.8 | 21.6 | 31.3 | 43.1 | 40.6 |
| **+SUDA** | ResNet-50 | 50.5 | 51.7 | 64.1 | 26.7 | 48.5 | 14.2 | 38.1 | 49.5 | **42.9** |
| SWDA [57] | ResNet-50 | 49.0 | 49.0 | 61.4 | 23.9 | 43.1 | 22.9 | 31.0 | 45.2 | 40.7 |
| **+SUDA** | ResNet-50 | 50.7 | 50.3 | 67.3 | 22.3 | 45.2 | 27.4 | 34.0 | 48.9 | **43.3** |
| CRDA [72] | ResNet-50 | 49.8 | 48.4 | 61.9 | 22.3 | 40.7 | 30.0 | 29.9 | 45.4 | 41.1 |
| **+SUDA** | ResNet-50 | 52.3 | 51.6 | 66.7 | 30.4 | 47.1 | 11.9 | 36.8 | 48.7 | **43.2** |
| CF [79] | ResNet-50 | 49.6 | 49.7 | 62.6 | 23.3 | 43.4 | 27.4 | 30.2 | 44.8 | 41.4 |
| **+SUDA** | ResNet-50 | 50.2 | 49.6 | 67.4 | 22.8 | 43.4 | 33.9 | 33.1 | 48.9 | **43.7** |
| SAP [36] | ResNet-50 | 49.3 | 49.9 | 62.5 | 23.0 | 44.1 | 29.4 | 31.3 | 45.8 | 41.9 |
| **+SUDA** | ResNet-50 | 51.2 | 51.4 | 68.5 | 25.3 | 48.0 | 26.5 | 33.8 | 49.9 | **44.3** |
| **SUDA** | ResNet-50 | 50.5 | 51.7 | 64.1 | 26.7 | 48.5 | 13.1 | 38.1 | 49.5 | **42.8** |
| Faster R-CNN [51] (Baseline) | ResNet-50 | 26.9 | 38.2 | 35.6 | 18.3 | 32.4 | 9.6 | 25.8 | 28.6 | 26.9 |
| DAF [12] | ResNet-50 | 29.2 | 40.4 | 43.4 | 19.7 | 38.3 | 28.5 | 23.7 | 32.7 | 32.0 |
| **+SUDA** | ResNet-50 | 38.8 | 44.8 | 54.1 | 29.3 | 50.7 | 44.1 | 31.4 | 39.1 | **41.5** |
| SCDA [81] | ResNet-50 | 33.8 | 42.1 | 52.1 | 26.8 | 42.5 | 26.5 | 29.2 | 34.5 | 35.9 |
| **+SUDA** | ResNet-50 | 38.7 | 47.0 | 54.1 | 27.6 | 51.8 | 46.5 | 29.7 | 39.4 | **41.9** |
| SWDA [57] | ResNet-50 | 31.8 | 44.3 | 48.9 | 21.0 | 43.8 | 28.0 | 28.9 | 35.8 | 35.3 |
| **+SUDA** | ResNet-50 | 39.5 | 48.2 | 57.6 | 29.5 | 52.9 | 37.5 | 34.5 | 41.5 | **42.7** |
| **SUDA** | ResNet-50 | 38.2 | 46.9 | 51.6 | 28.5 | 48.5 | 37.2 | 32.8 | 39.2 | **40.4** |

# 4. Applications - Domain Adaptation

The spectrum transformer is generic and can work for different domain adaptive tasks such as detection, segmentation, classification, etc. The table shows image classification over Office-31 dataset.

| Method | A→W | D→W | W→D | A→D | D→A | W→A | Mean |
|---|---|---|---|---|---|---|---|
| ResNet-50 [25] | 68.4 | 96.7 | 99.3 | 68.9 | 62.5 | 60.7 | 76.1 |
| DAN [42] | 80.5 | 97.1 | 99.6 | 78.6 | 63.6 | 62.8 | 80.4 |
| RTN [43] | 84.5 | 96.8 | 99.4 | 77.5 | 66.2 | 64.8 | 81.6 |
| DANN [20] | 82.0 | 96.9 | 99.1 | 79.7 | 68.2 | 67.4 | 82.2 |
| ADDA [66] | 86.2 | 96.2 | 98.4 | 77.8 | 69.5 | 68.9 | 82.9 |
| JAN [44] | 85.4 | 97.4 | 99.8 | 84.7 | 68.6 | 70.0 | 84.3 |
| GTA [61] | 89.5 | 97.9 | 99.8 | 87.7 | 72.8 | 71.4 | 86.5 |
| CBST [83] | 87.8 | 98.5 | 100 | 86.5 | 71.2 | 70.9 | 85.8 |
| **+SUDA** | 91.2 | 99.2 | 100 | 93.4 | 75.2 | 73.6 | **88.8** |
| CRST [84] | 89.4 | 98.9 | 100 | 88.7 | 72.6 | 70.9 | 86.8 |
| **+SUDA** | 91.6 | 98.6 | 100 | 94.3 | 76.2 | 75.9 | **89.4** |
| **SUDA** | 90.1 | 98.0 | 99.8 | 93.0 | 73.7 | 73.5 | **88.0** |

# 4. Applications - Domain Adaptation

The spectrum transformer is generic and can work for different domain adaptive tasks such as detection, segmentation, classification, etc. The table shows the domain adaptive semantic segmentation over the task SYNTHIA →Cityscapes.

| Method | Road | SW | Build | Wall* | Fence* | Pole* | TL | TS | Veg. | Sky | PR | Rider | Car | Bus | Motor | Bike | mIoU | mIoU* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Baseline [25] | 55.6 | 23.8 | 74.6 | 9.2 | 0.2 | 24.4 | 6.1 | 12.1 | 74.8 | 79.0 | 55.3 | 19.1 | 39.6 | 23.3 | 13.7 | 25.0 | 33.5 | 38.6 |
| PatAlign [65] | 82.4 | 38.0 | 78.6 | 8.7 | 0.6 | 26.0 | 3.9 | 11.1 | 75.5 | 84.6 | 53.5 | 21.6 | 71.4 | 32.6 | 19.3 | 31.7 | 40.0 | 46.5 |
| AdaptSeg [64] | 84.3 | 42.7 | 77.5 | - | - | - | 4.7 | 7.0 | 77.9 | 82.5 | 54.3 | 21.0 | 72.3 | 32.2 | 18.9 | 32.3 | - | 46.7 |
| CLAN [45] | 81.3 | 37.0 | 80.1 | - | - | - | 16.1 | 13.7 | 78.2 | 81.5 | 53.4 | 21.2 | 73.0 | 32.9 | 22.6 | 30.7 | - | 47.8 |
| AdvEnt [68] | 85.6 | 42.2 | 79.7 | 8.7 | 0.4 | 25.9 | 5.4 | 8.1 | 80.4 | 84.1 | 57.9 | 23.8 | 73.3 | 36.4 | 14.2 | 33.0 | 41.2 | 48.0 |
| IDA [46] | 84.3 | 37.7 | 79.5 | 5.3 | 0.4 | 24.9 | 9.2 | 8.4 | 80.0 | 84.1 | 57.2 | 23.0 | 78.0 | 38.1 | 20.3 | 36.5 | 41.7 | 48.9 |
| CrCDA [30] | 86.2 | 44.9 | 79.5 | 8.3 | 0.7 | 27.8 | 9.4 | 11.8 | 78.6 | 86.5 | 57.2 | 26.1 | 76.8 | 39.9 | 21.5 | 32.1 | 42.9 | 50.0 |
| CRST [84] | 67.7 | 32.2 | 73.9 | 10.7 | 1.6 | 37.4 | 22.2 | 31.2 | 80.8 | 80.5 | 60.8 | 29.1 | 82.8 | 25.0 | 19.4 | 45.3 | 43.8 | 50.1 |
| BDL [38] | 86.0 | 46.7 | 80.3 | - | - | - | 14.1 | 11.6 | 79.2 | 81.3 | 54.1 | 27.9 | 73.7 | 42.2 | 25.7 | 45.3 | - | 51.4 |
| SIM [70] | 83.0 | 44.0 | 80.3 | - | - | - | 17.1 | 15.8 | 80.5 | 81.8 | 59.9 | 33.1 | 70.2 | 37.3 | 28.5 | 45.8 | - | 52.1 |
| TIR [32] | 92.6 | 53.2 | 79.2 | - | - | - | 1.6 | 7.5 | 78.6 | 84.4 | 52.6 | 20.0 | 82.1 | 34.8 | 14.6 | 39.4 | - | 49.3 |
| **+SUDA** | 83.9 | 40.1 | 76.9 | 4.5 | 0.1 | 26.1 | 22.9 | 26.4 | 79.6 | 80.7 | 58.1 | 28.3 | 81.0 | 37.4 | 35.1 | 46.8 | 45.5 | **53.6** |
| FDA [76] | 79.3 | 35.0 | 73.2 | - | - | - | 19.9 | 24.0 | 61.7 | 82.6 | 61.4 | 31.1 | 83.9 | 40.8 | 38.4 | 51.1 | - | 52.5 |
| **+SUDA** | 85.6 | 38.8 | 76.7 | 9.2 | 0.2 | 28.4 | 25.4 | 27.0 | 78.4 | 81.7 | 60.4 | 28.6 | 82.8 | 38.8 | 36.2 | 48.1 | 46.7 | **54.5** |
| **SUDA** | 83.4 | 36.0 | 71.3 | 8.7 | 0.1 | 26.0 | 18.2 | 26.7 | 72.4 | 80.2 | 58.4 | 30.8 | 80.6 | 38.7 | 36.1 | 46.1 | 44.6 | **52.2** |