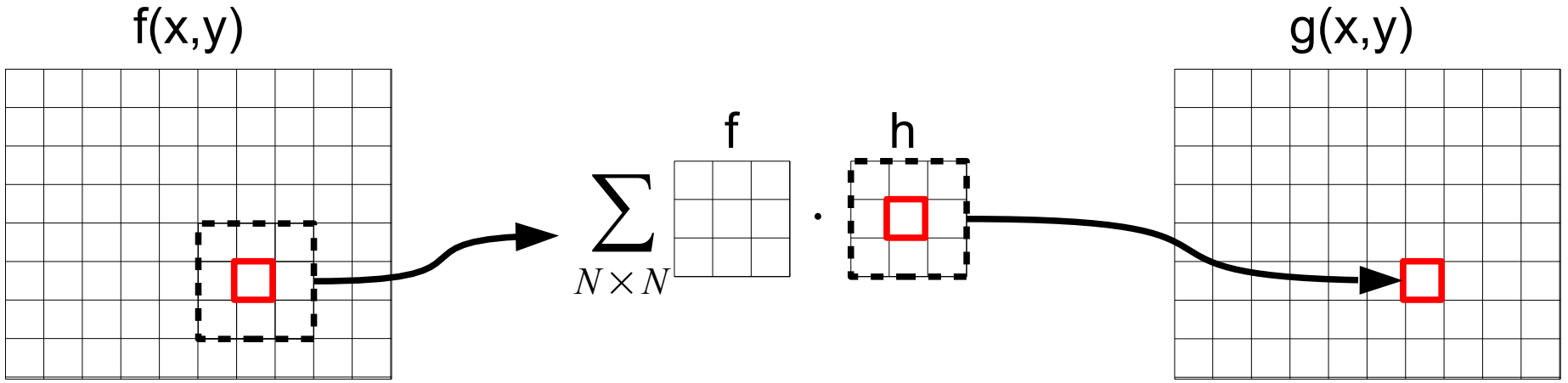
Digital Image Processing

Berlin University of Technology (TUB),
Computer Vision and Remote Sensing Group
Berlin, Germany

Spatial Domain Convolution

$$g(\alpha,\beta)=\sum_{x=1}^{N}\sum_{y=1}^{N} f(x,y)\cdot h(x-\alpha,y-\beta)$$



Runtime complexity:

→ Assuming kernel size == image size

→ $O(N^4)$

Frequency Domain

The Discrete Fourier Transform

Time domain
(complex signal)

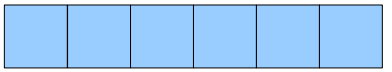
$a(x) =$ 

$a(0), a(1), \ldots, a(N-1)$

DFT

IDFT

Frequency domain
(complex signal)

$A(\mu) =$ 

$A(0), A(1), \ldots, A(N-1)$

$$A(\mu) = \sum_{x=0}^{N-1} a(x) \exp\left(-\frac{2\pi i}{N} \mu x\right)$$ Forward DFT

$$a(x) = \frac{1}{N} \sum_{\mu=0}^{N-1} A(\mu) \exp\left(\frac{2\pi i}{N} \mu x\right)$$ Inverse DFT

Frequency Domain

The Discrete Fourier Transform

Time domain
(complex signal)

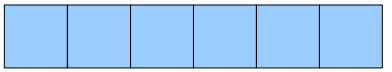
$a(x) =$ 

$a(0), a(1), \ldots, a(N-1)$

DFT

IDFT

Frequency domain
(complex signal)

$A(\mu) =$ 

$A(0), A(1), \ldots, A(N-1)$

$$A(\mu) = \sum_{x=0}^{N-1} a(x) \exp\left(-\frac{2\pi i}{N} \mu x\right)$$ Forward DFT

$$a(x) = \frac{1}{N} \sum_{\mu=0}^{N-1} A(\mu) \exp\left(\frac{2\pi i}{N} \mu x\right)$$ Inverse DFT

Frequency Domain

The Discrete Fourier Transform

Time domain
(complex signal)

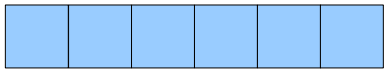
$a(x) =$ 

$a(0), a(1), \ldots, a(N-1)$

DFT

IDFT

Frequency domain
(complex signal)

$A(\mu) =$ 

$A(0), A(1), \ldots, A(N-1)$

$$A(\mu) = \sum_{x=0}^{N-1} a(x) \exp\left(-\frac{2\pi i}{N} \mu x\right)$$ Forward DFT

$$a(x) = \frac{1}{N} \sum_{\mu=0}^{N-1} A(\mu) \exp\left(\frac{2\pi i}{N} \mu x\right)$$ Inverse DFT

Basis Function

Complex exponential
(oscillation)

Frequency Domain

The Discrete Fourier Transform

Time domain
(complex signal)

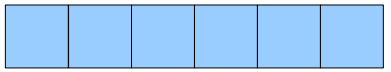
$a(x) =$ 

$a(0), a(1), \ldots, a(N-1)$

DFT

IDFT

Frequency domain
(complex signal)

$A(\mu) =$ 

$A(0), A(1), \ldots, A(N-1)$

$$A(\mu) = \sum_{x=0}^{N-1} a(x) \exp\left(-\frac{2\pi i}{N} \mu x\right)$$ Forward DFT

$$a(x) = \frac{1}{N} \sum_{\mu=0}^{N-1} A(\mu) \exp\left(\frac{2\pi i}{N} \mu x\right)$$ Inverse DFT

Spectrum

Frequency domain representation: Weight and phase of frequency

Basis Function

Complex exponential (oscillation)

Frequency Domain

The Discrete Fourier Transform

Time domain
(complex signal)

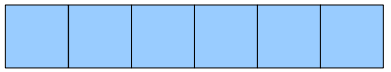
$a(x) =$ 

$a(0), a(1), \ldots, a(N-1)$

DFT

IDFT

Frequency domain
(complex signal)

$A(\mu) =$ 

$A(0), A(1), \ldots, A(N-1)$

$$A(\mu) = \sum_{x=0}^{N-1} a(x) \exp\left(-\frac{2\pi i}{N} \mu x\right)$$ Forward DFT

$$a(x) = \frac{1}{N} \sum_{\mu=0}^{N-1} A(\mu) \exp\left(\frac{2\pi i}{N} \mu x\right)$$ Inverse DFT

Signal

Reconstructed from the spectrum

Spectrum

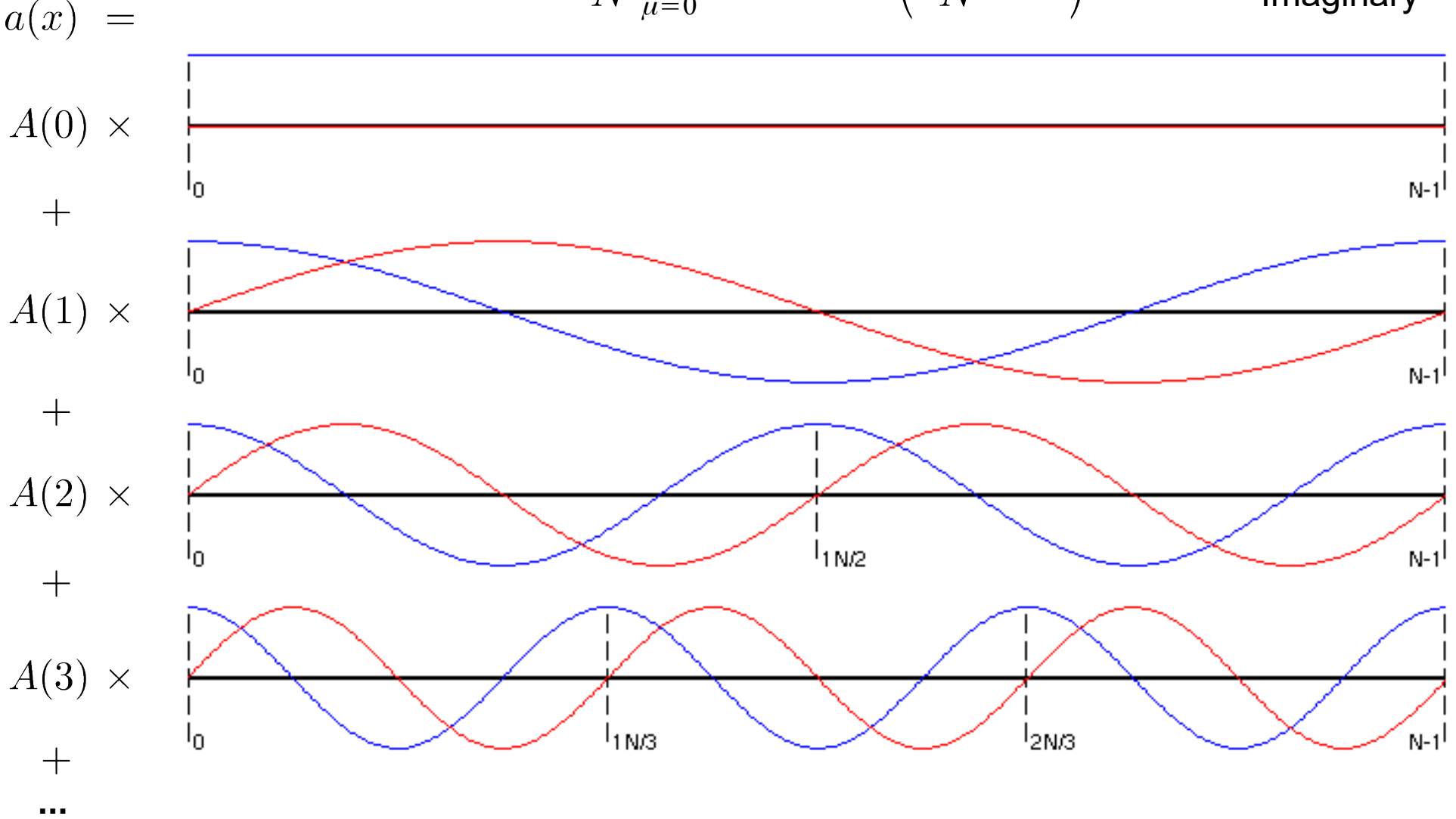
Frequency domain representation: Weight and phase of frequency

Basis Function

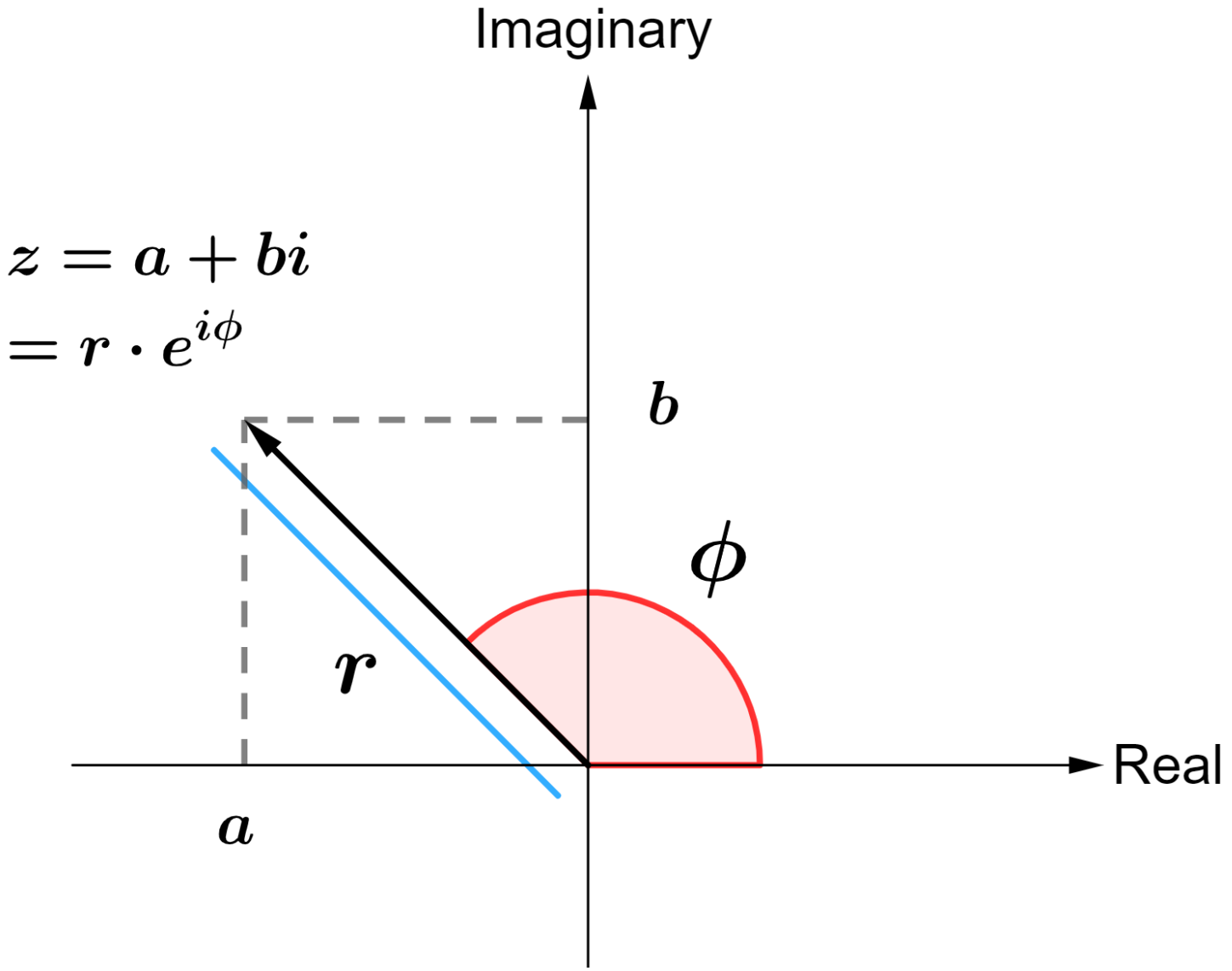
Complex exponential (oscillation)

Frequency Domain

$$a(x) = \frac{1}{N} \sum_{\mu=0}^{N-1} A(\mu) \exp\left(\frac{2\pi i}{N} \mu x\right)$$

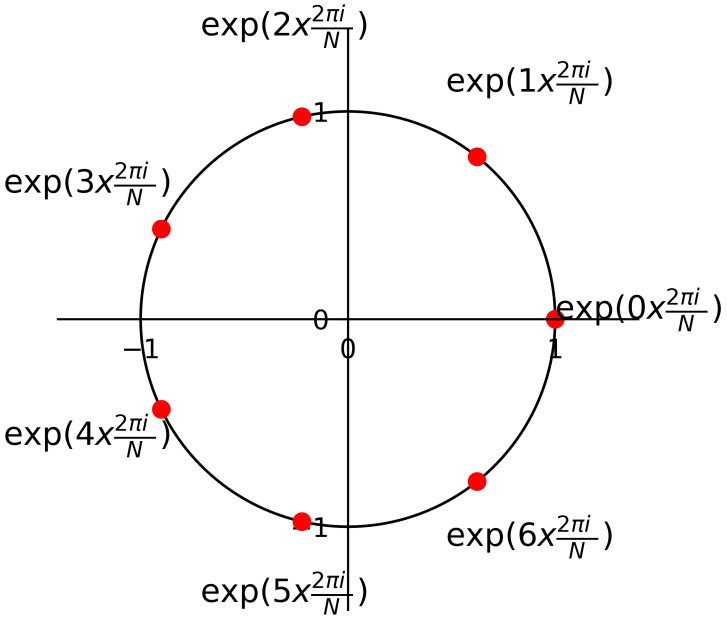


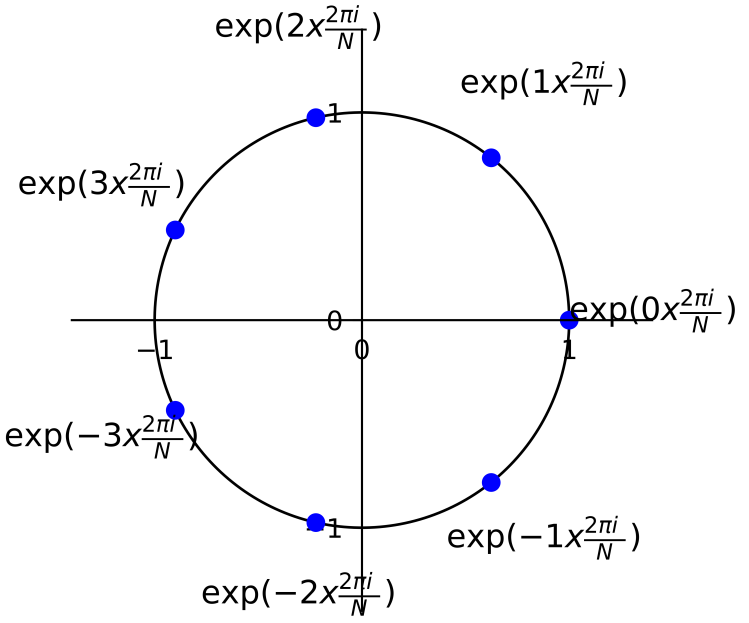
Frequency Domain
Negative Frequencies



Frequency Domain

Negative Frequencies



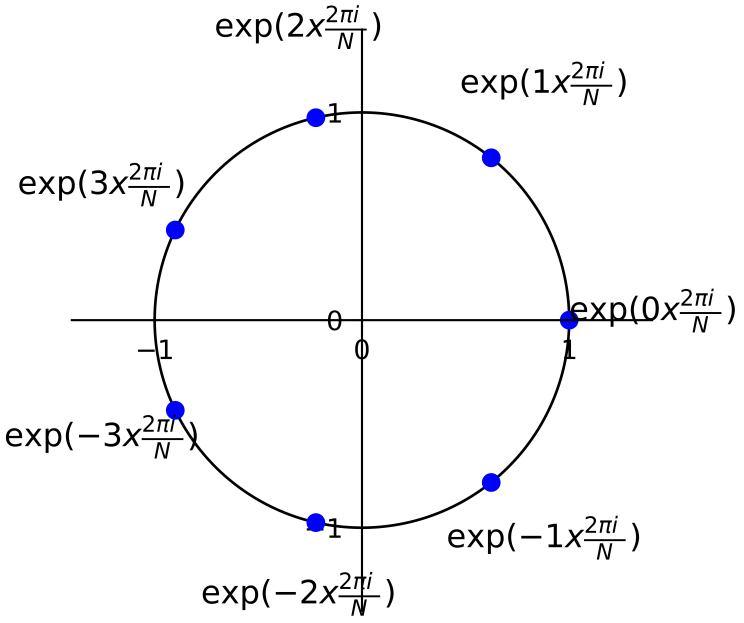


$$A(\mu) = \sum_{x=0}^{N-1} a(x) \exp\left(-\frac{2\pi i}{N}\mu x\right) = \sum_{x=0}^{N-1} a(x) \exp\left(-\frac{2\pi i}{N}(\mu + kN)x\right) = A(\mu + kN)$$

$$a(x) = \frac{1}{N}\sum_{\mu=0}^{N-1} A(\mu) \exp(\frac{2\pi i}{N}\mu x) = \sum_{x=-N/2}^{N/2} a(x) \exp(-\frac{2\pi i}{N}\mu x)$$

Frequency Domain

Negative Frequencies



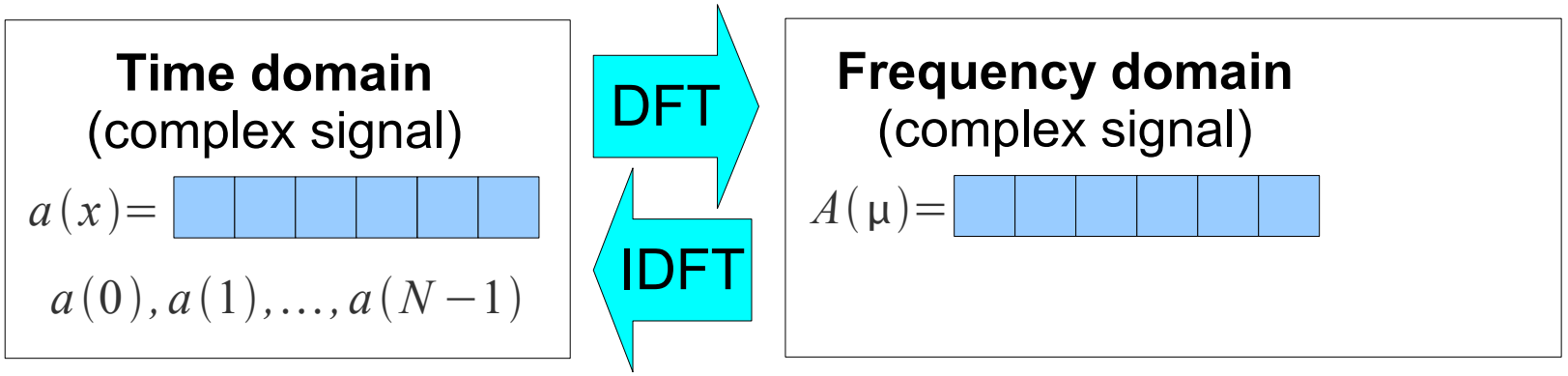
$$A(\mu) = \sum_{x=0}^{N-1} a(x) \exp\left(-\frac{2\pi i}{N} \mu x\right)$$

$$a(x) = \frac{1}{N} \sum_{\mu=0}^{N-1} A(\mu) \exp\left(\frac{2\pi i}{N} \mu x\right)$$

Frequency Domain
Negative Frequencies

$$A(\mu) = \sum_{x=0}^{N-1} a(x) \exp\left(-\frac{2\pi i}{N} \mu x\right)$$ Forward DFT

$$a(x) = \frac{1}{N} \sum_{\mu=0}^{N-1} A(\mu) \exp\left(\frac{2\pi i}{N} \mu x\right)$$ Inverse DFT



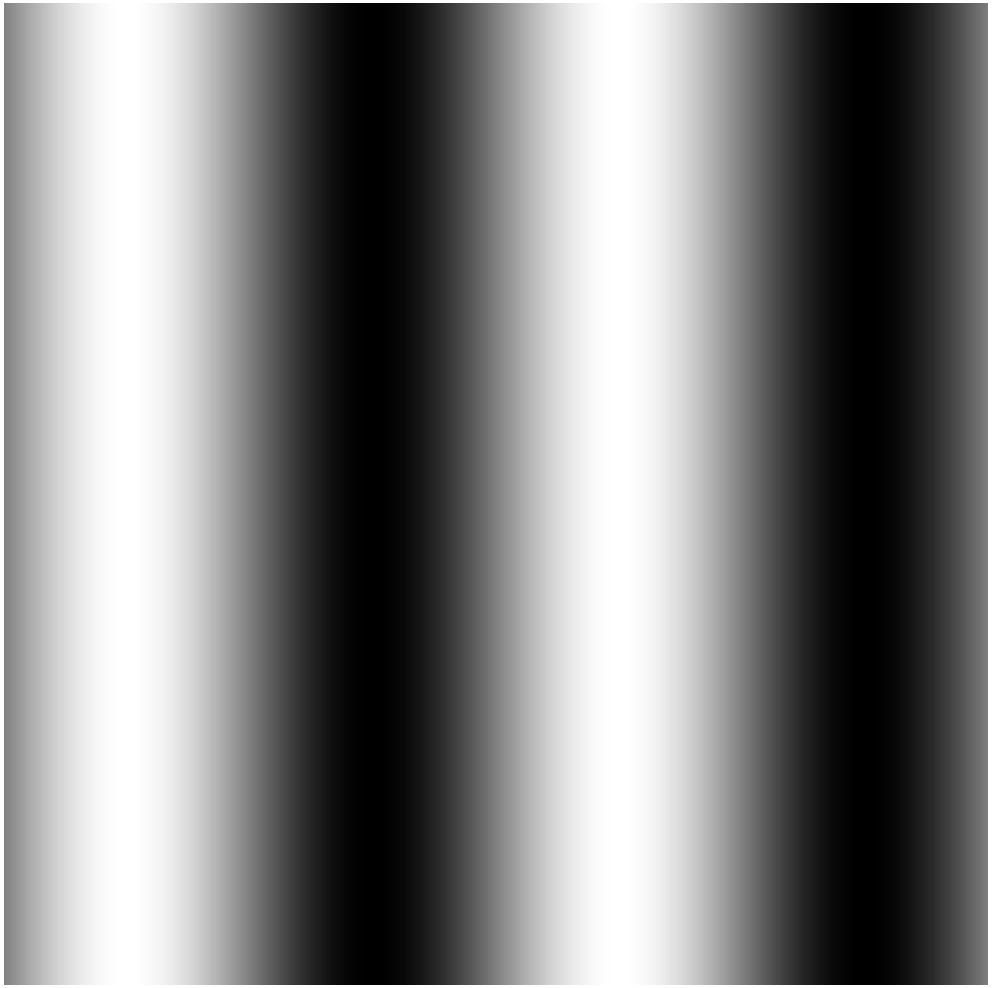
- The DFT can be redefined for negative frequencies
- This can also be done for other frequency ranges

DFT

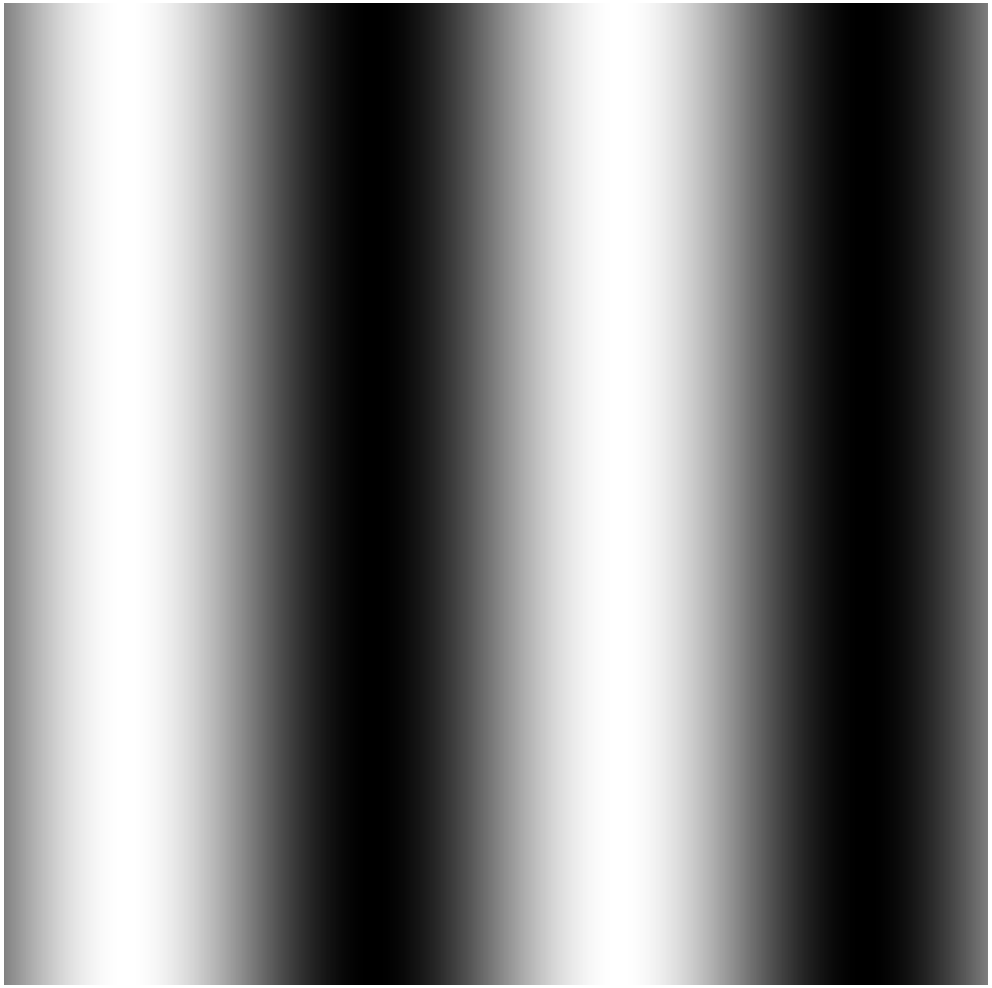
- **Discrete spectrum implies periodic signal**
  - Image values are infinitely repeating
- **Discrete signal implies periodic spectrum**
  - Fourier spectrum is also infinitely repeating
  - We can shift the frequency domain
- **Number of samples identical in both domains**
  - Input and output of DFT have the same dimensions
- **DFT is defined on complex signals**
  - Imaginary part of complex frequencies must cancel each other out

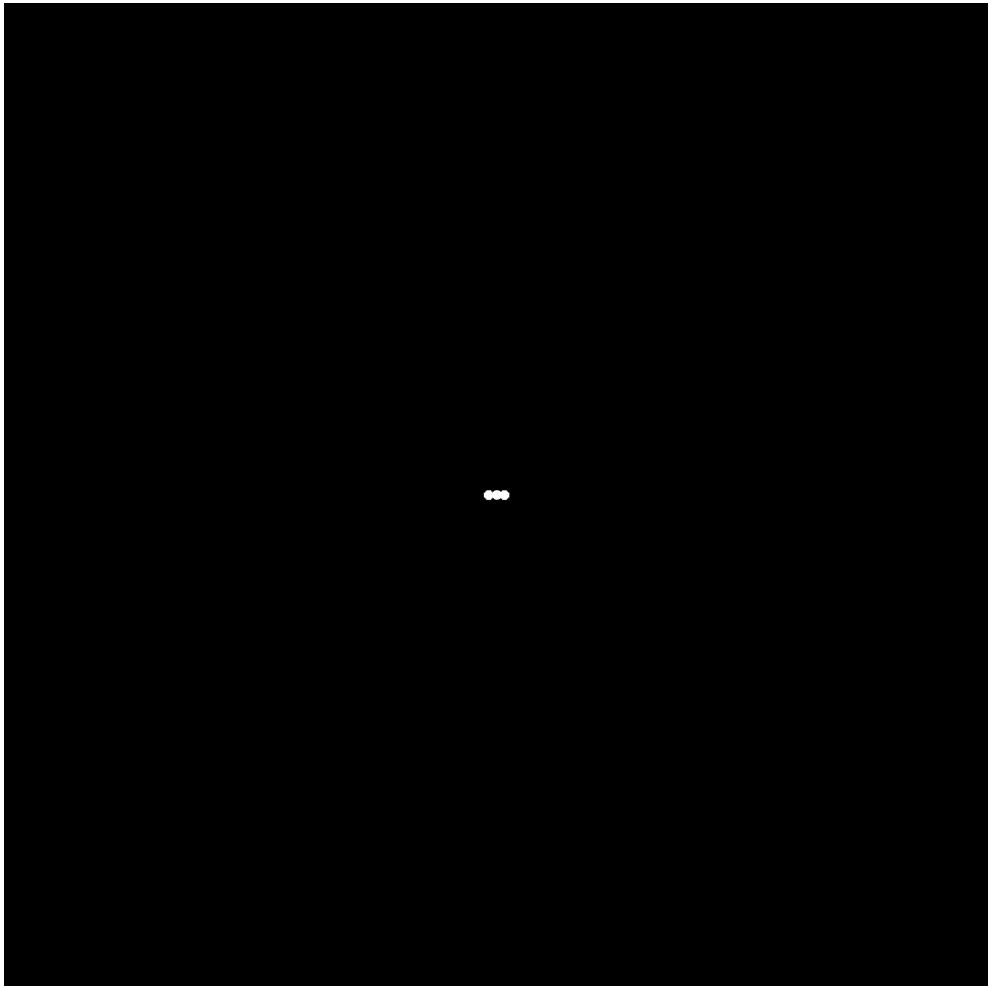
$$f(x) \in \mathbb{R} \implies F(-\mu) = F(\mu)^*$$

Frequency Domain

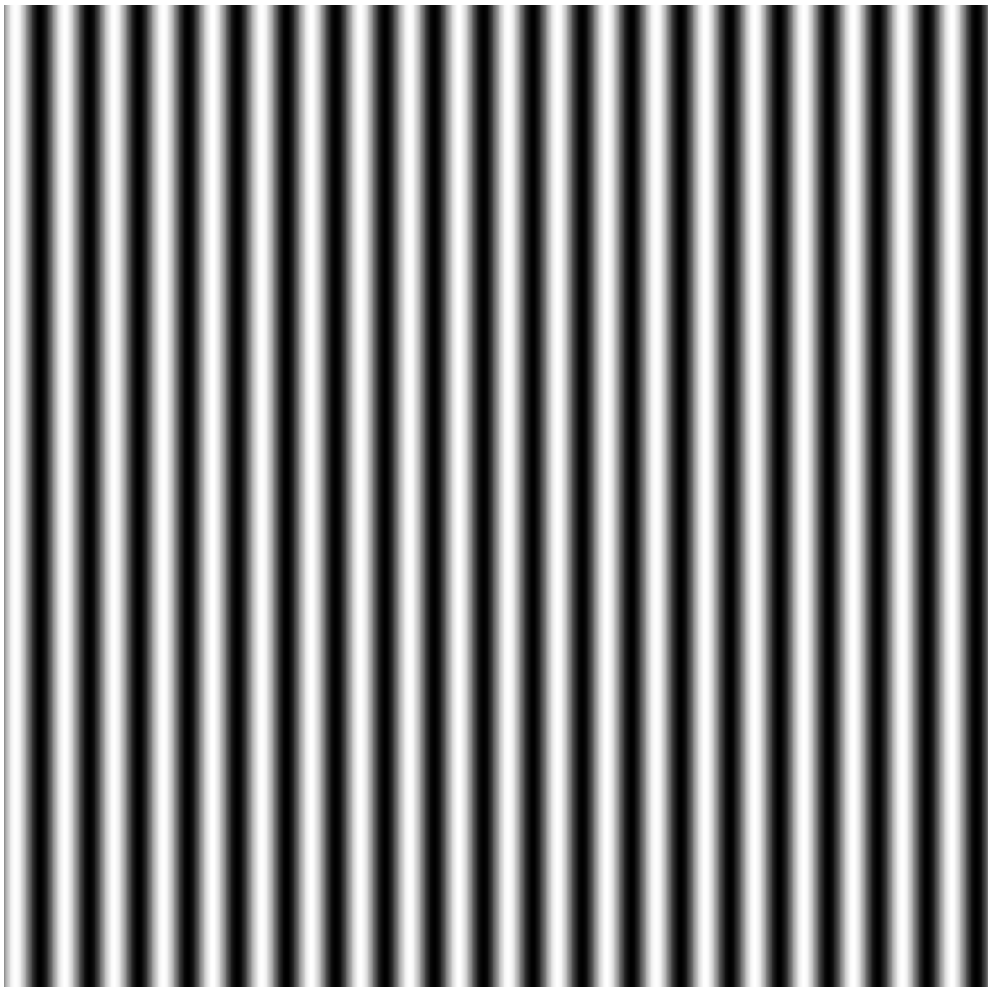


Frequency Domain

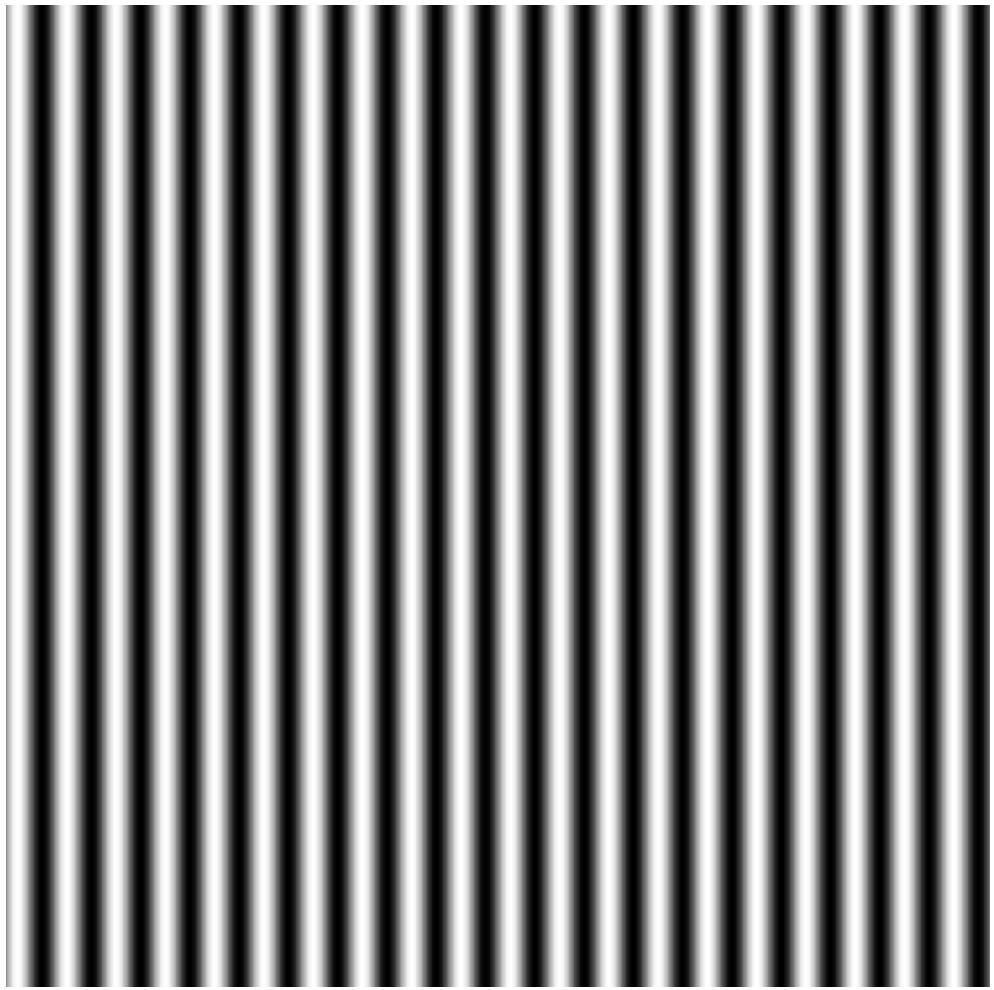


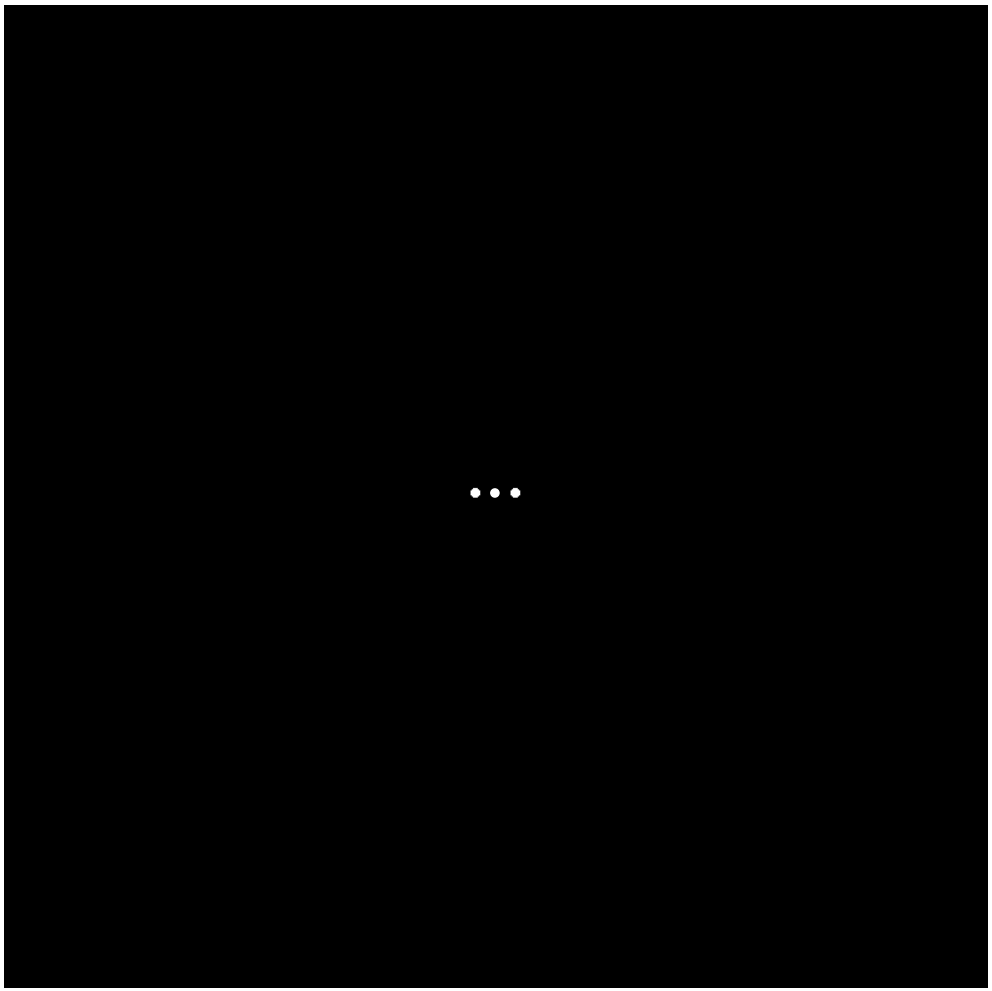


Frequency Domain

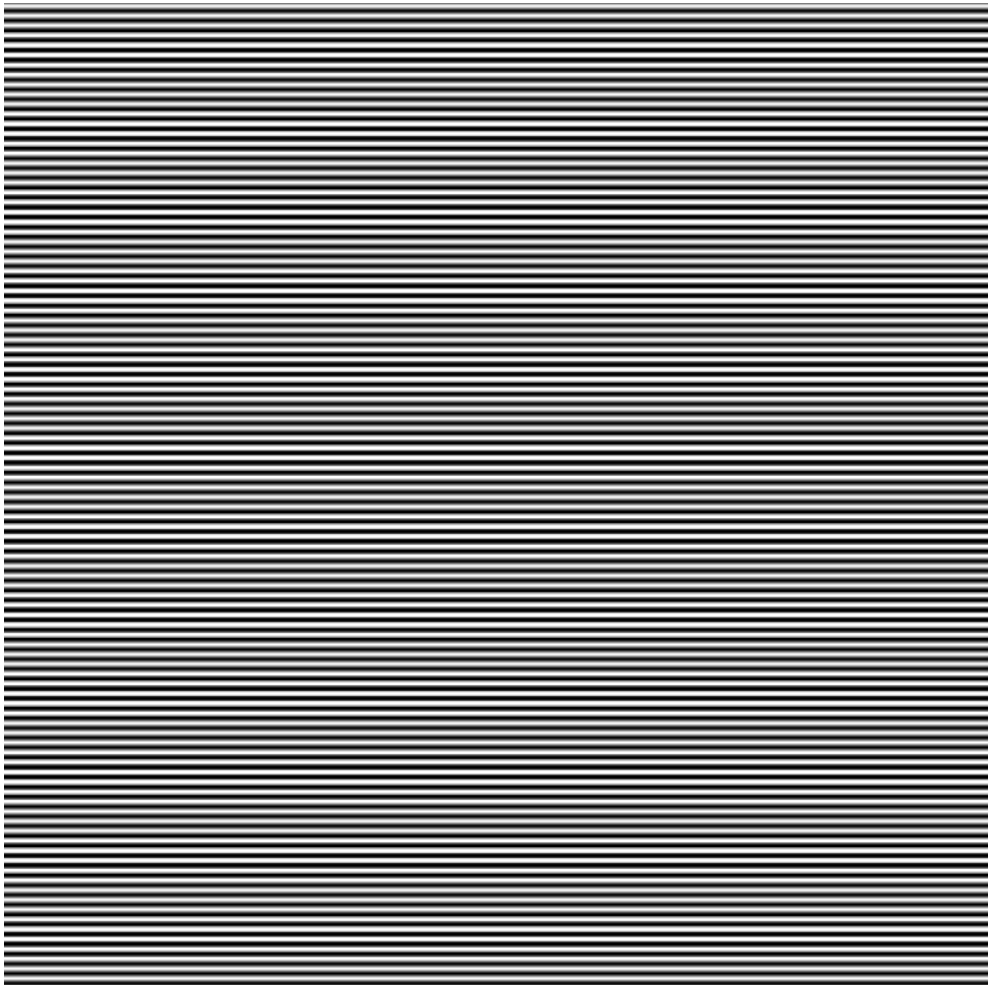


Frequency Domain

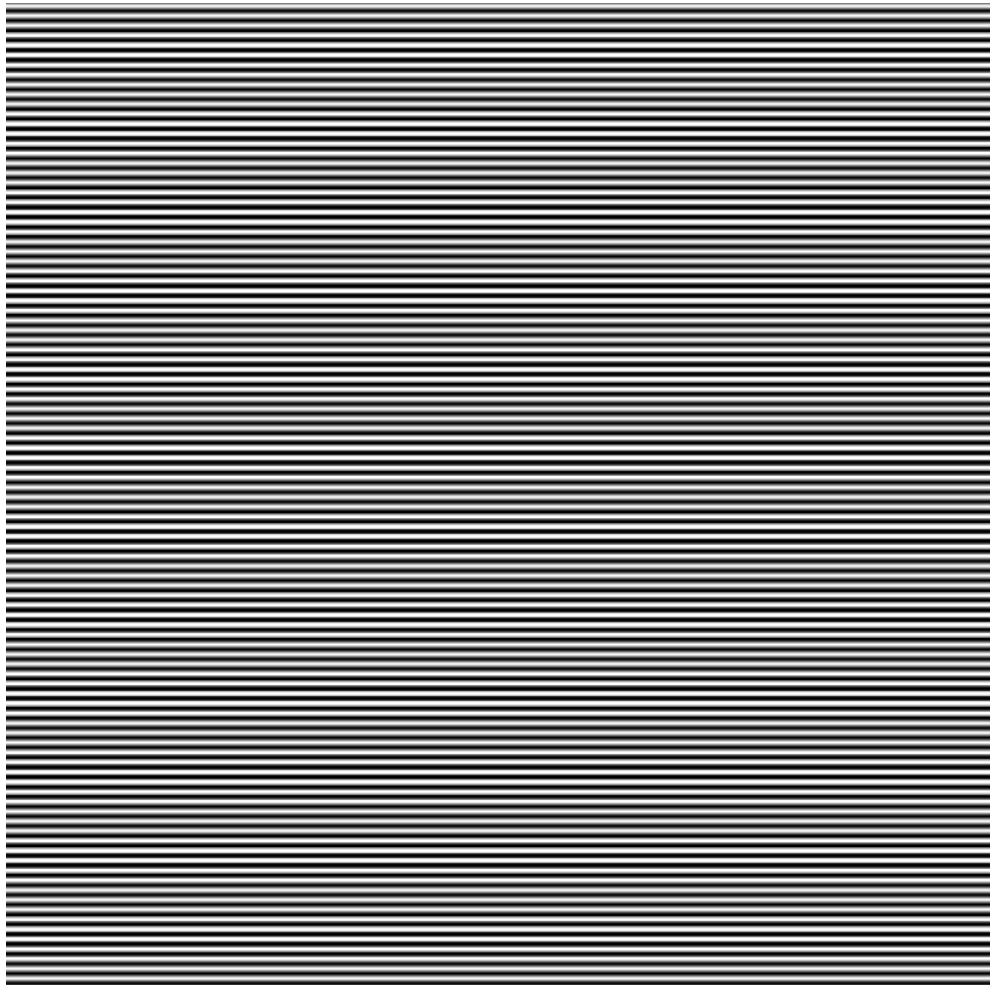


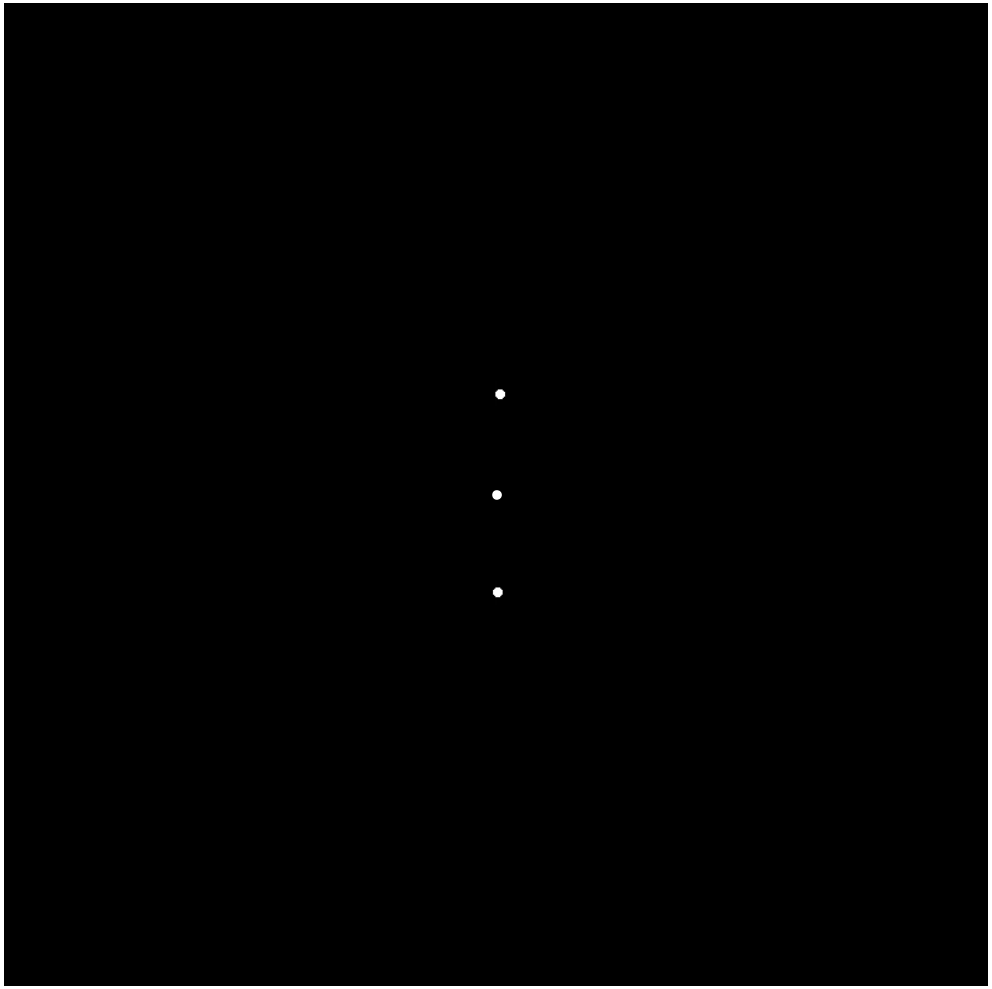


Frequency Domain

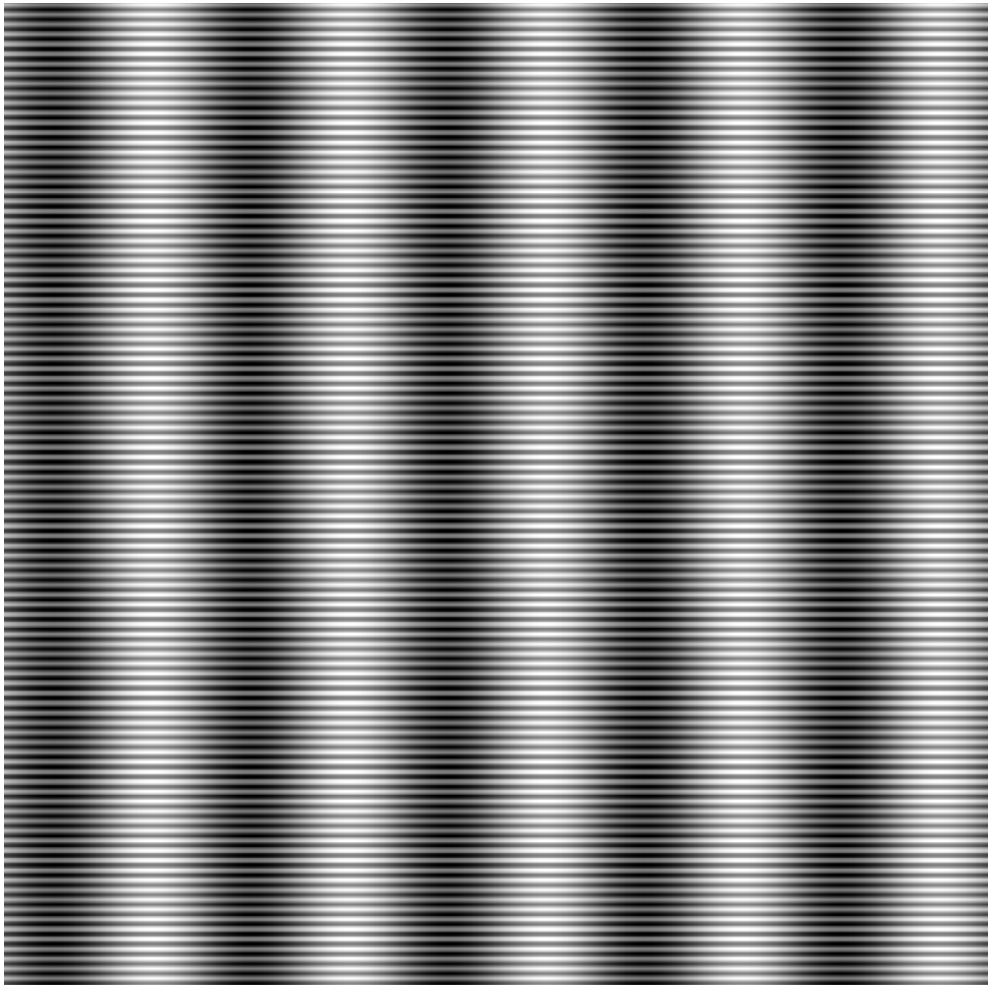


Frequency Domain

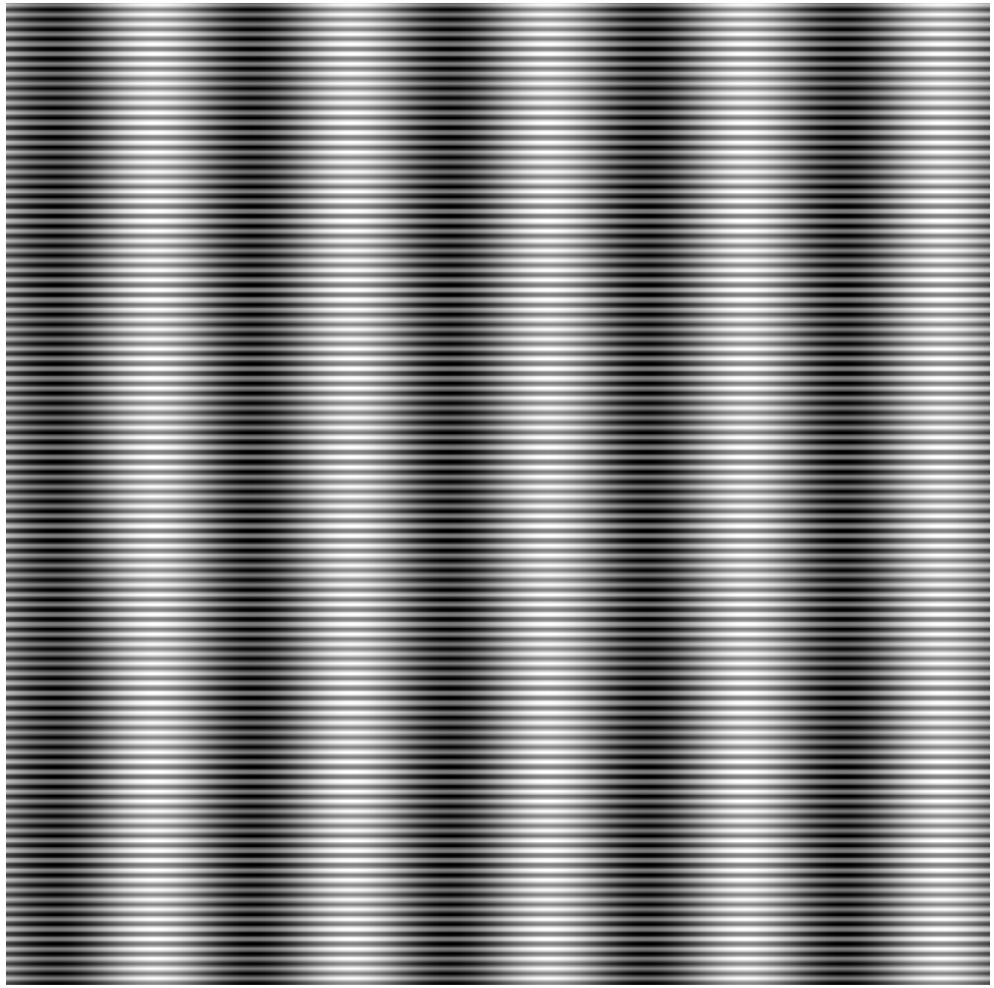


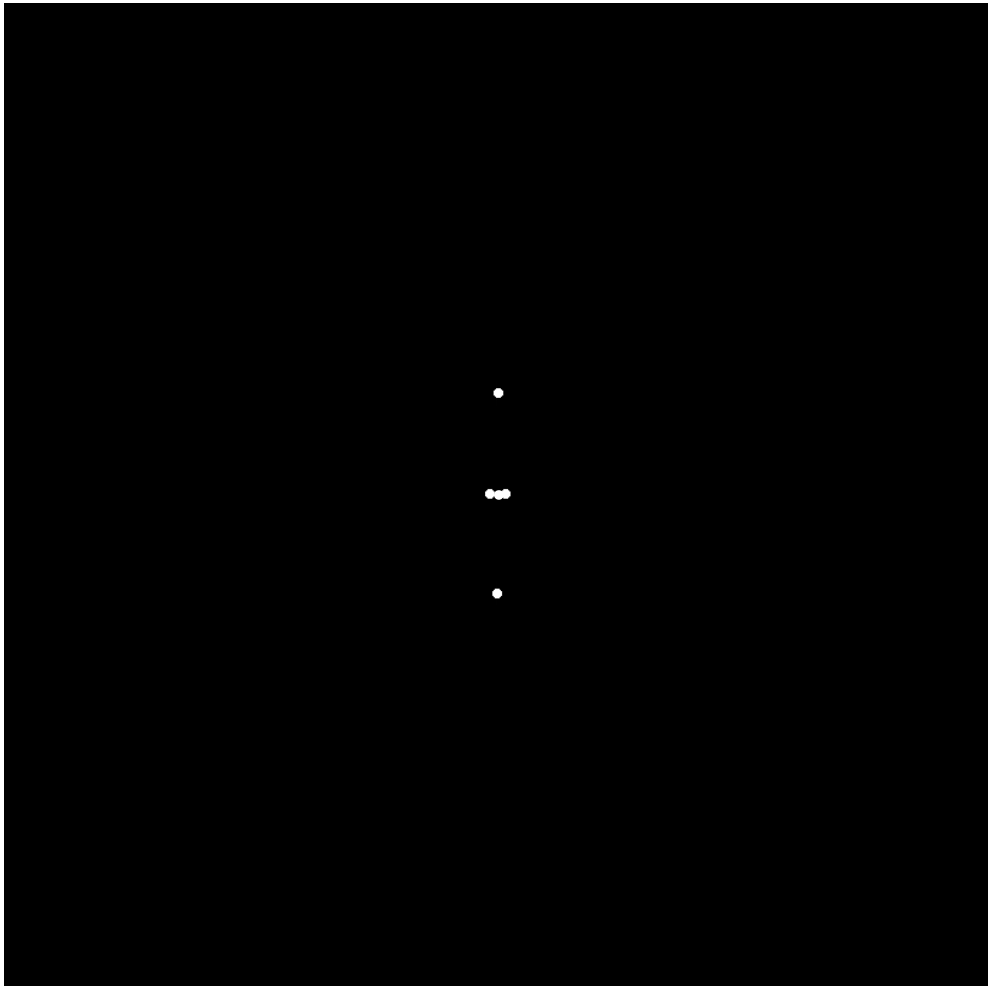


Frequency Domain

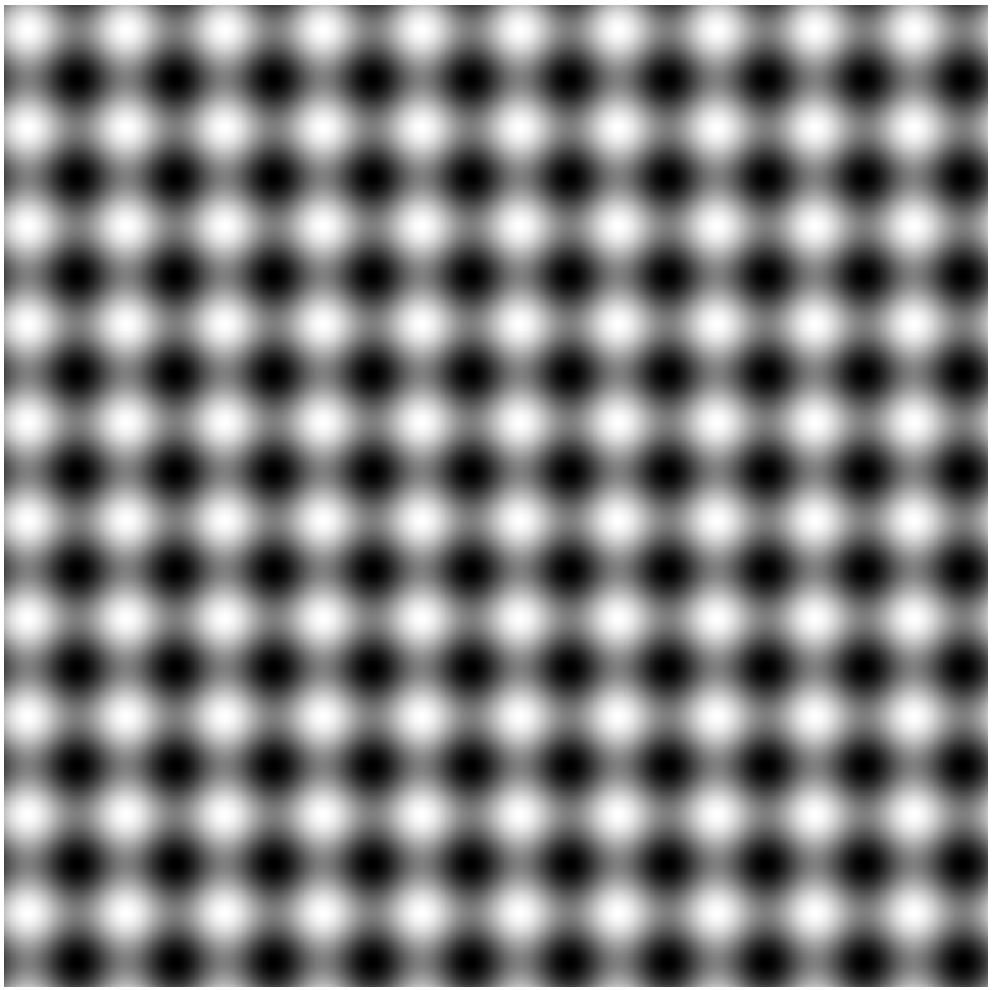


Frequency Domain

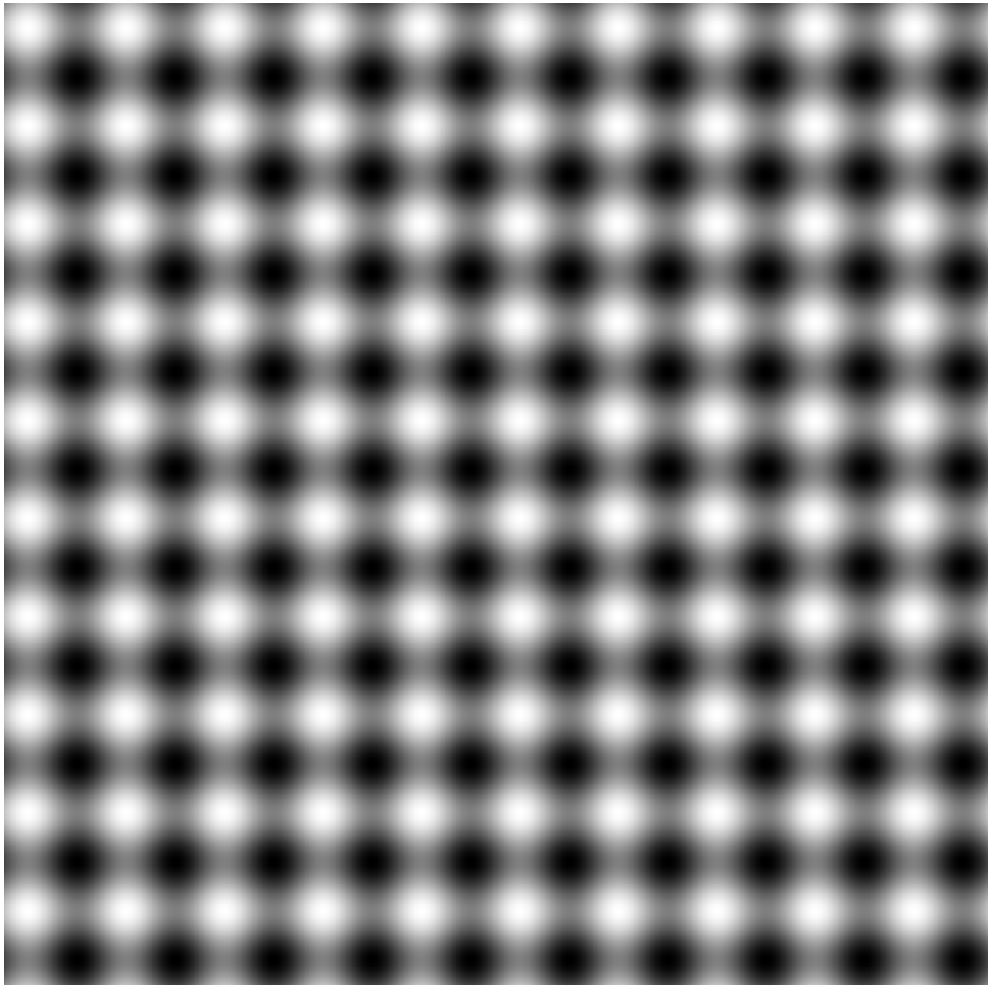


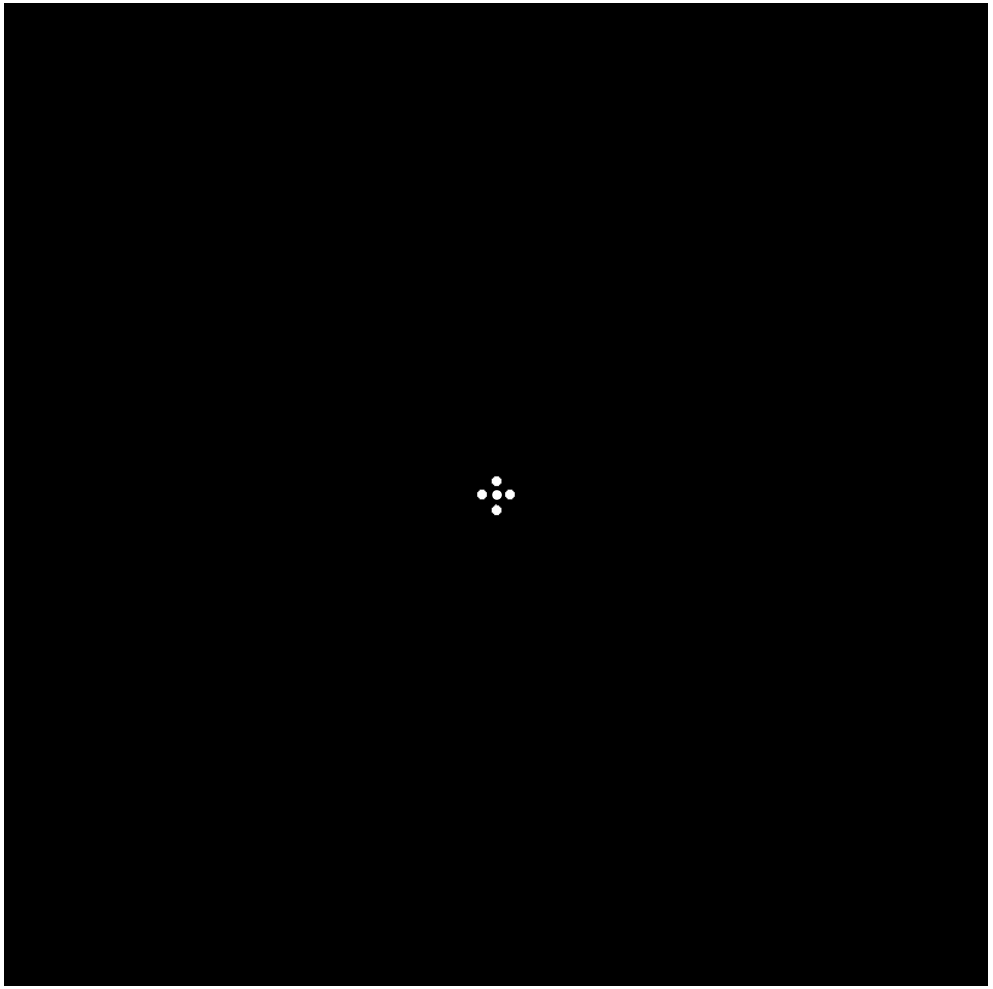


Frequency Domain



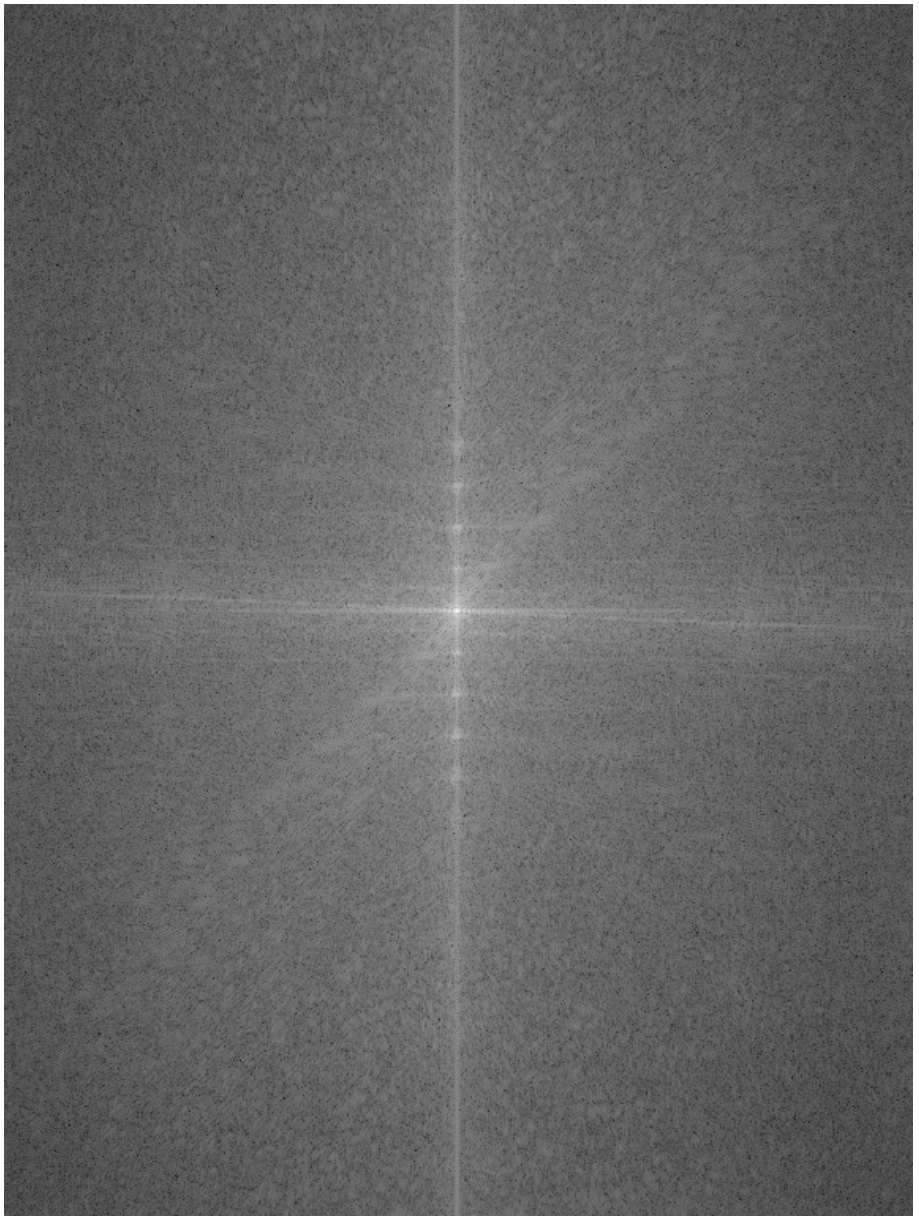
Frequency Domain





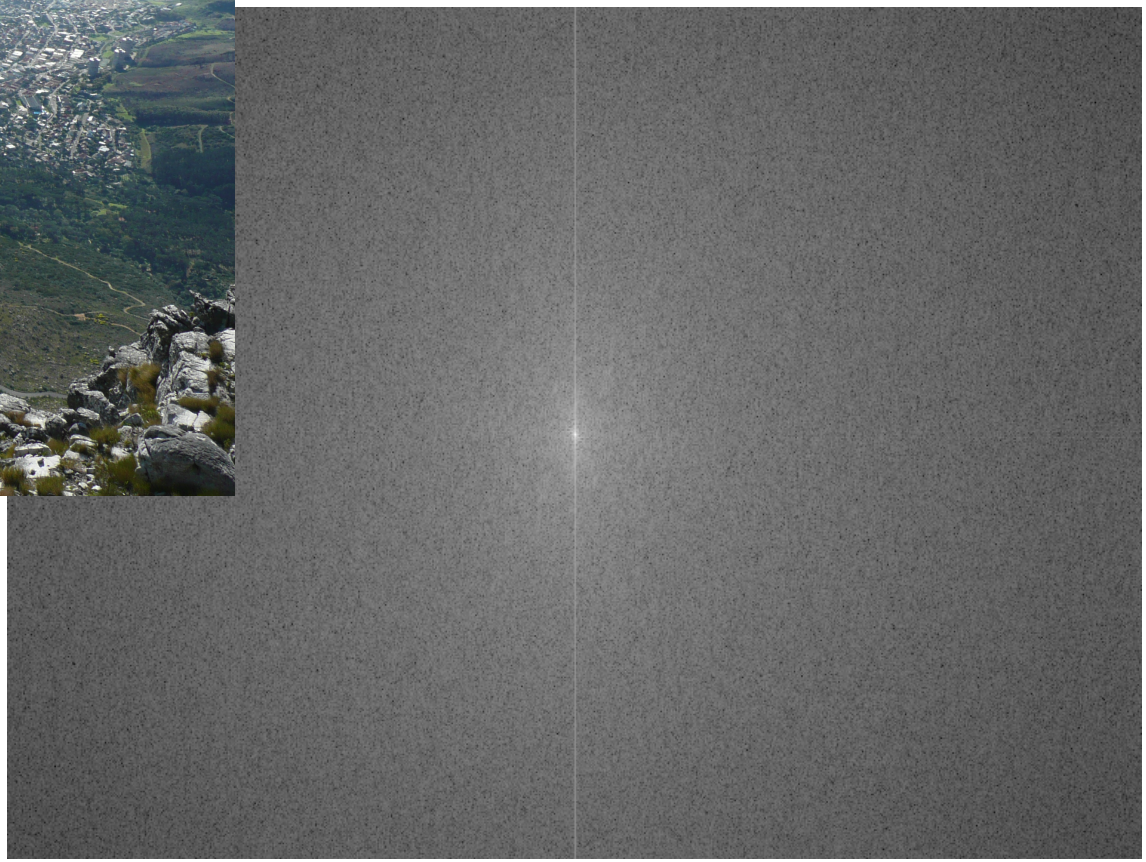
Frequency Domain

Frequency Domain

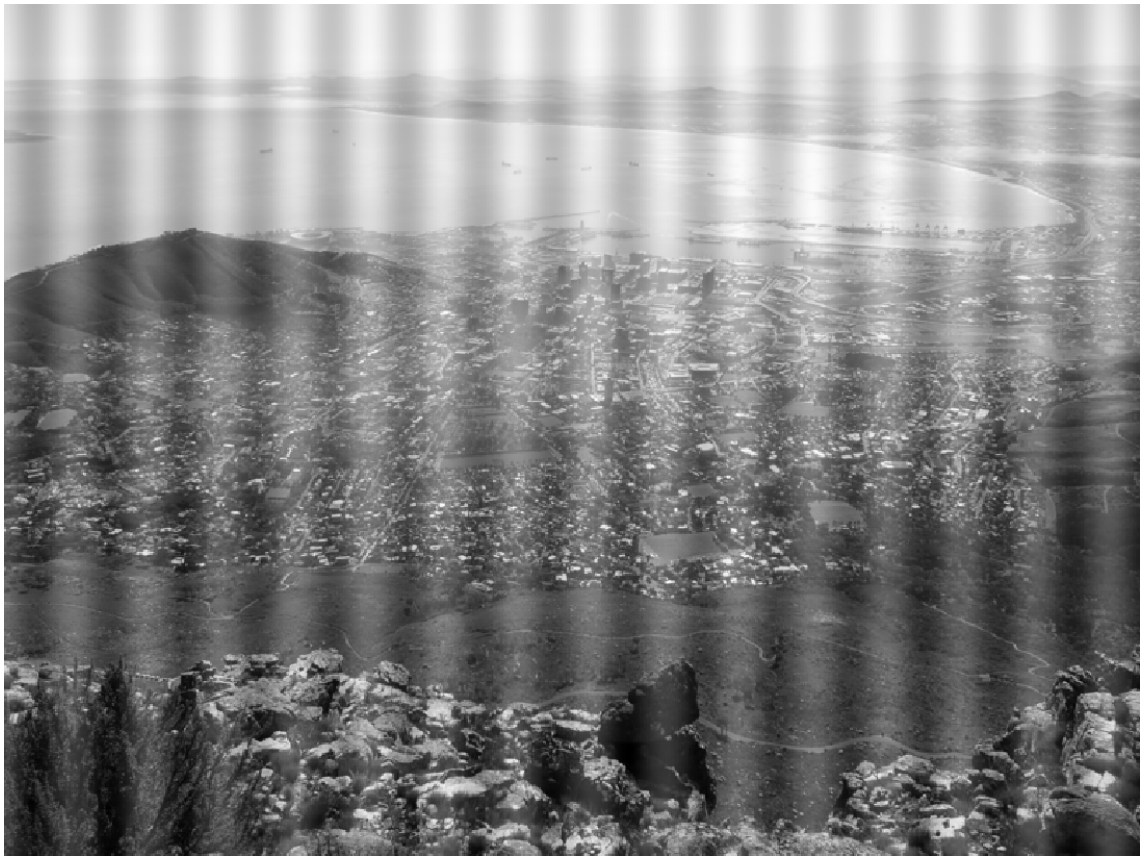


Frequency Domain

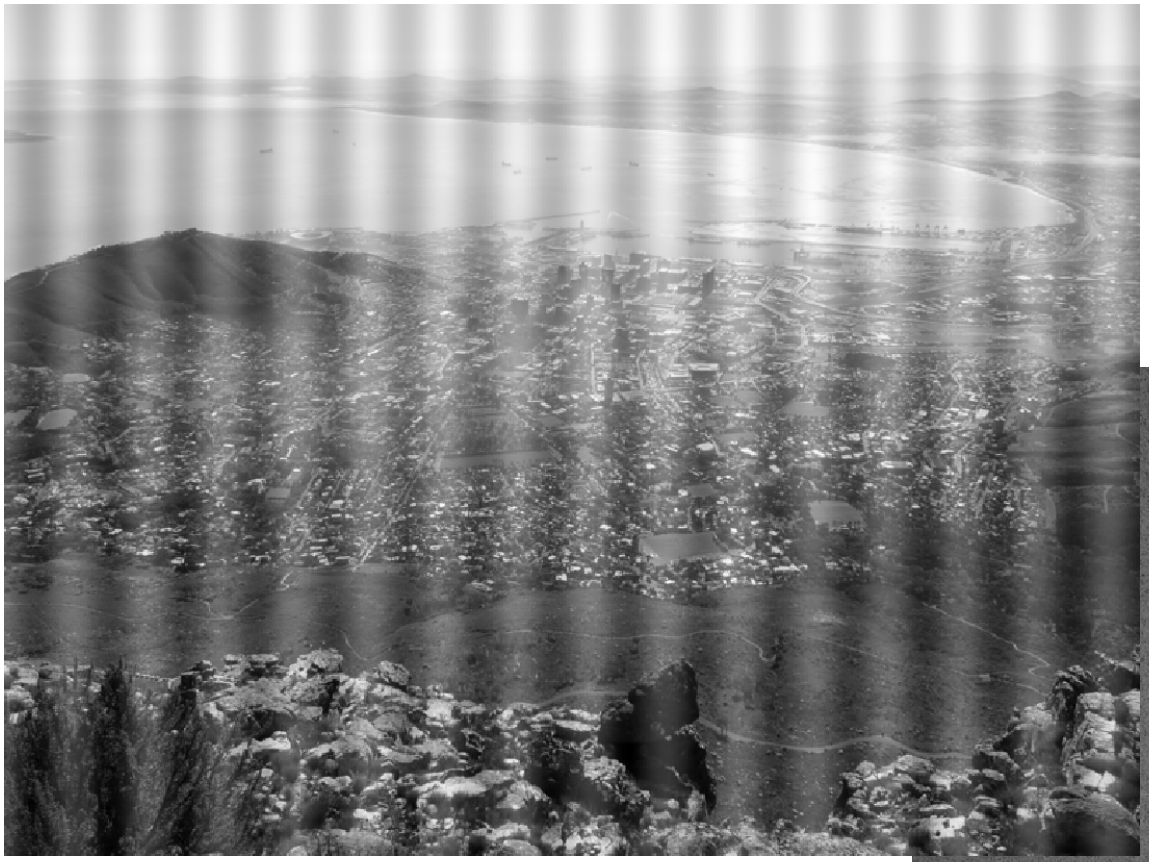
Frequency Domain

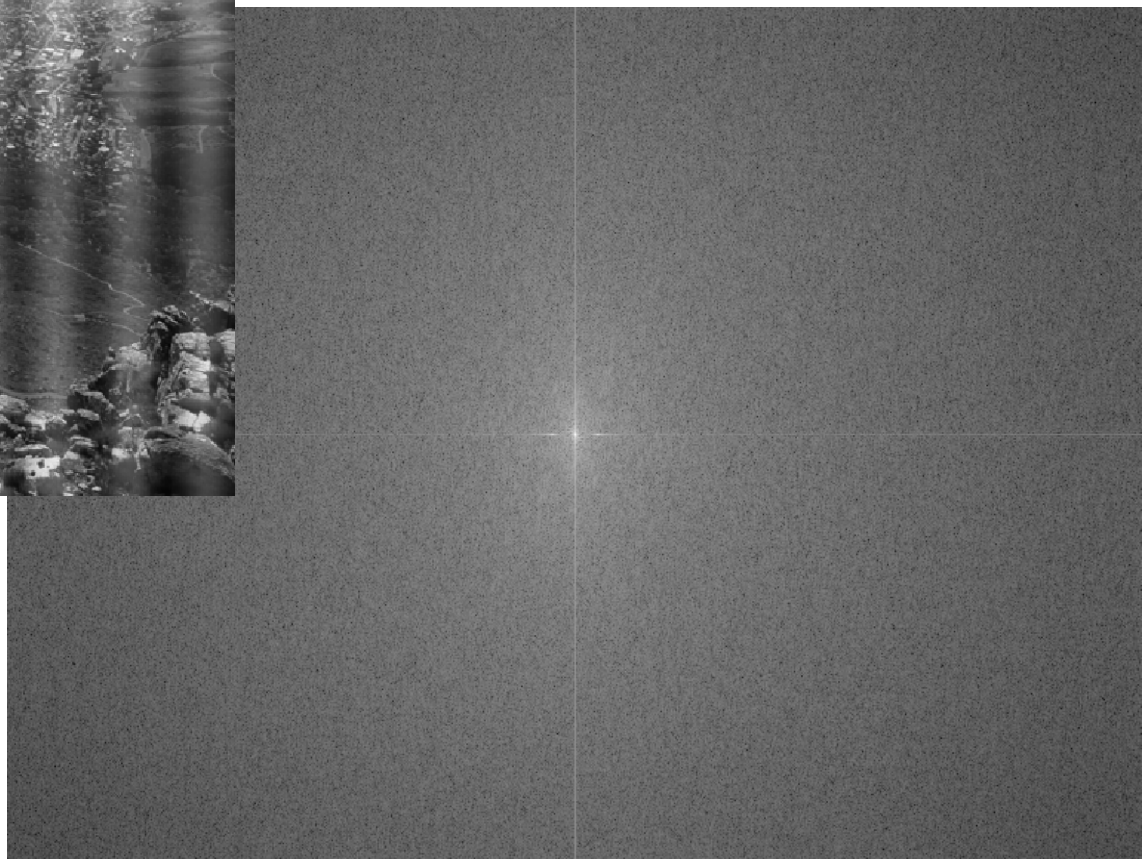


Frequency Domain

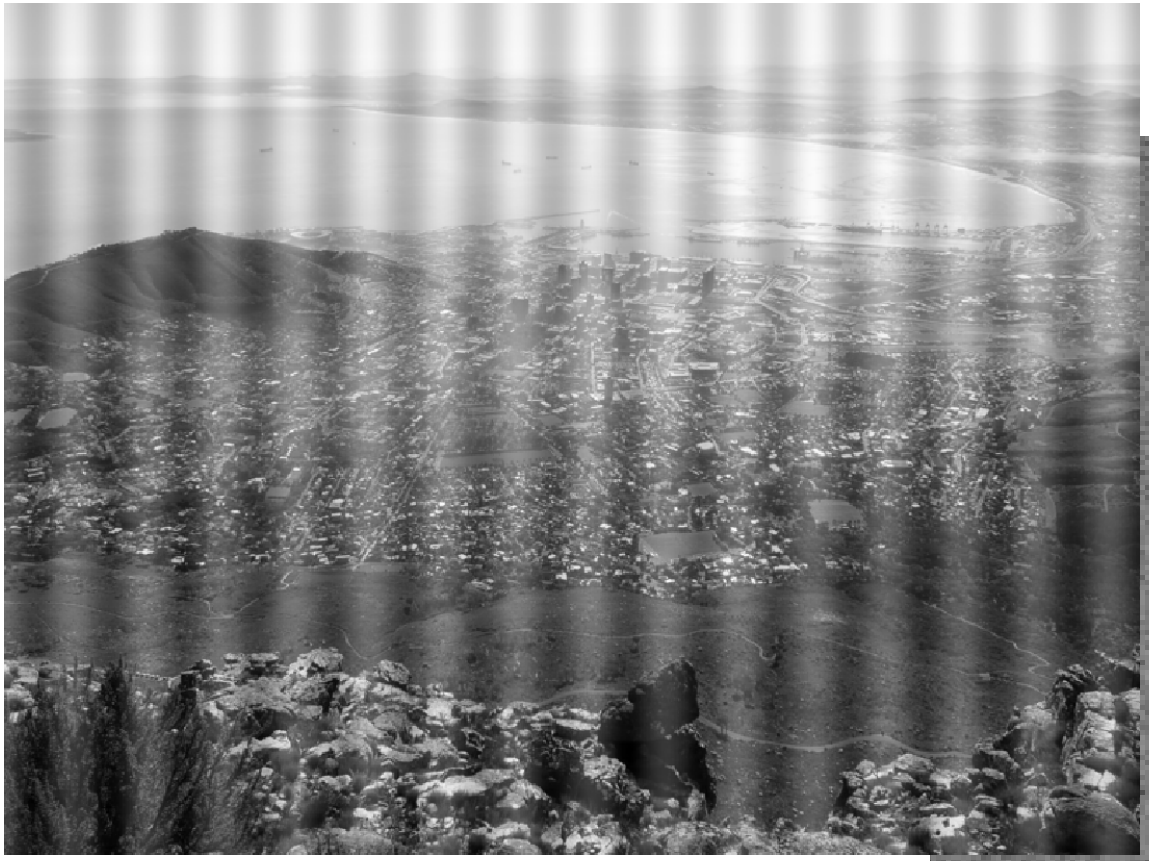


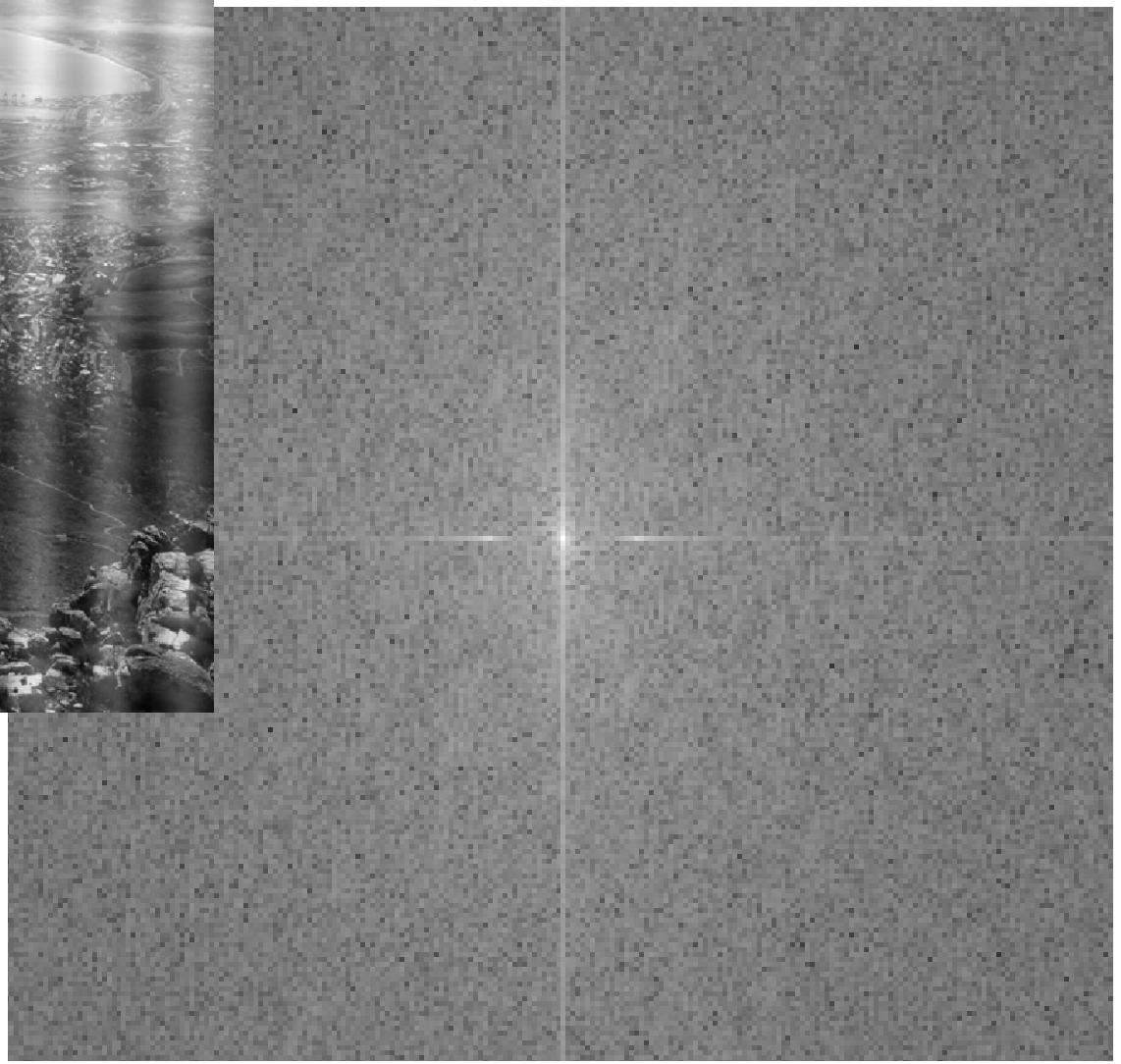
Frequency Domain



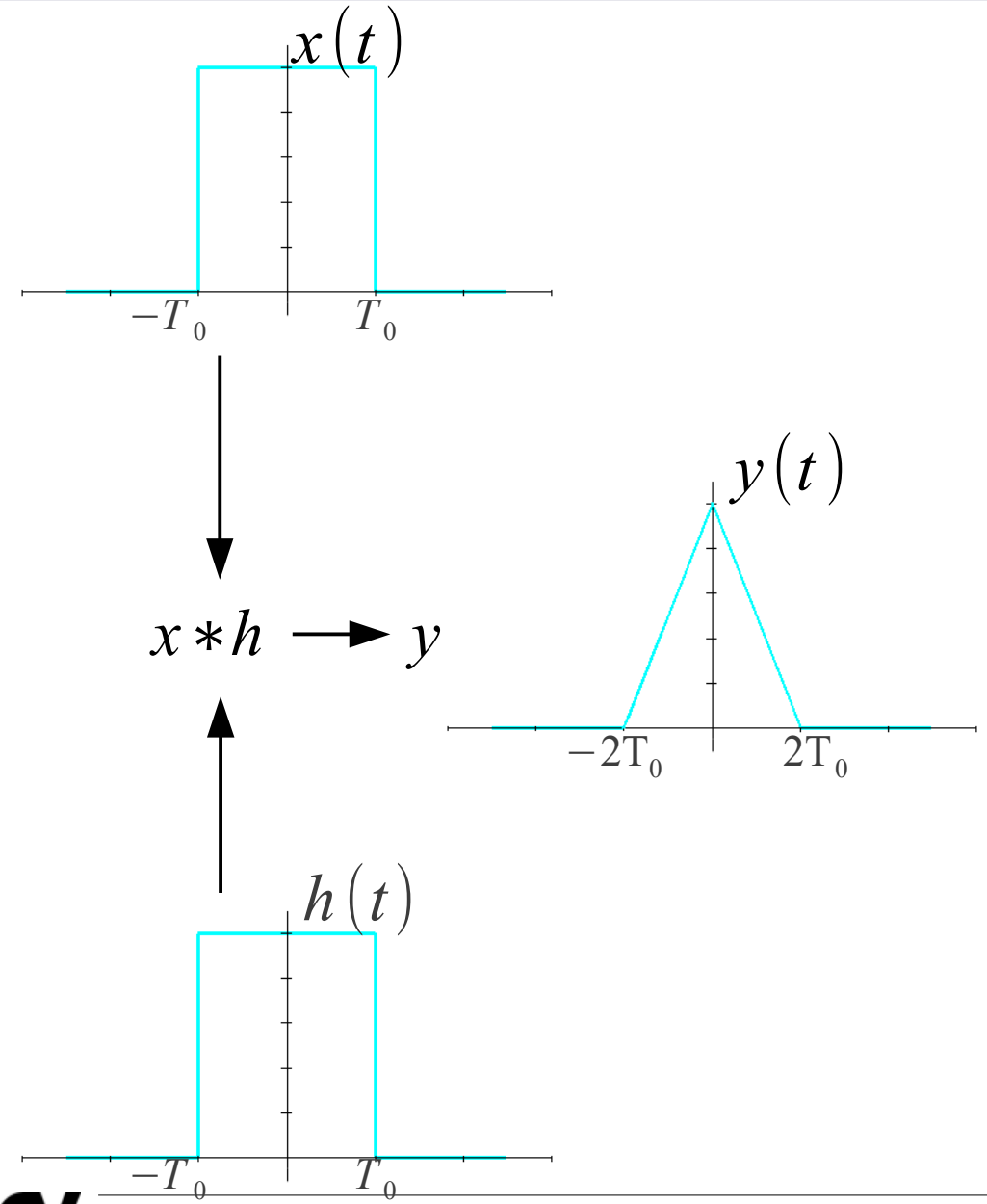


Frequency Domain

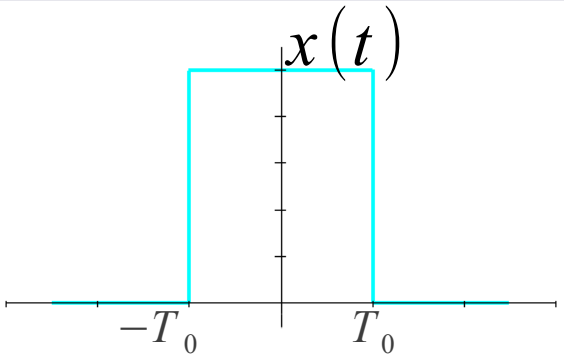


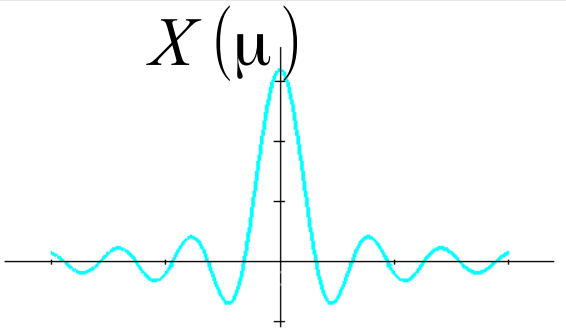


Example (Convolution Theorem)

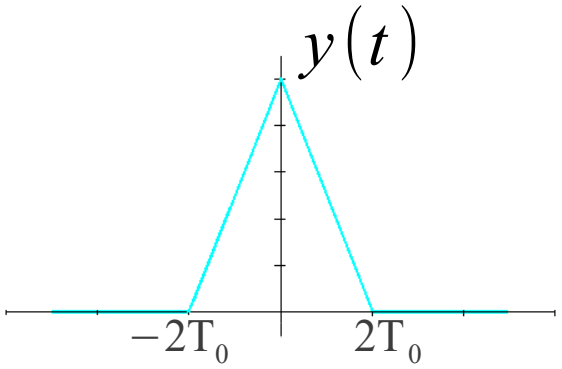


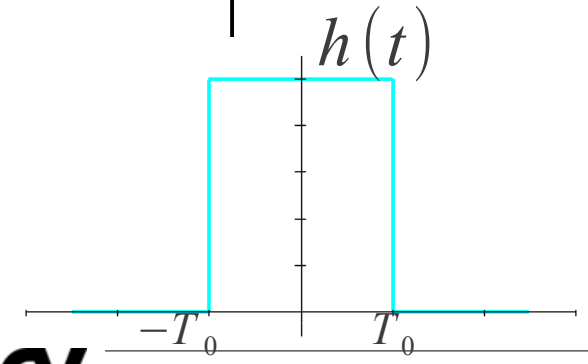
Example (Convolution Theorem)

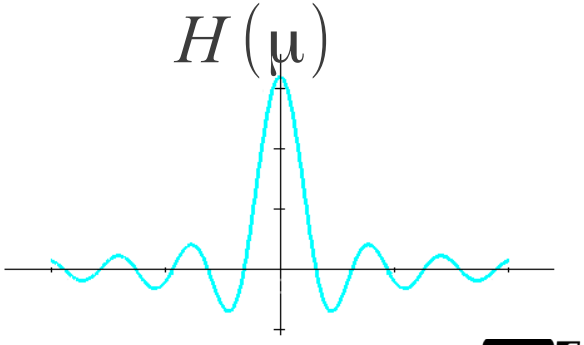




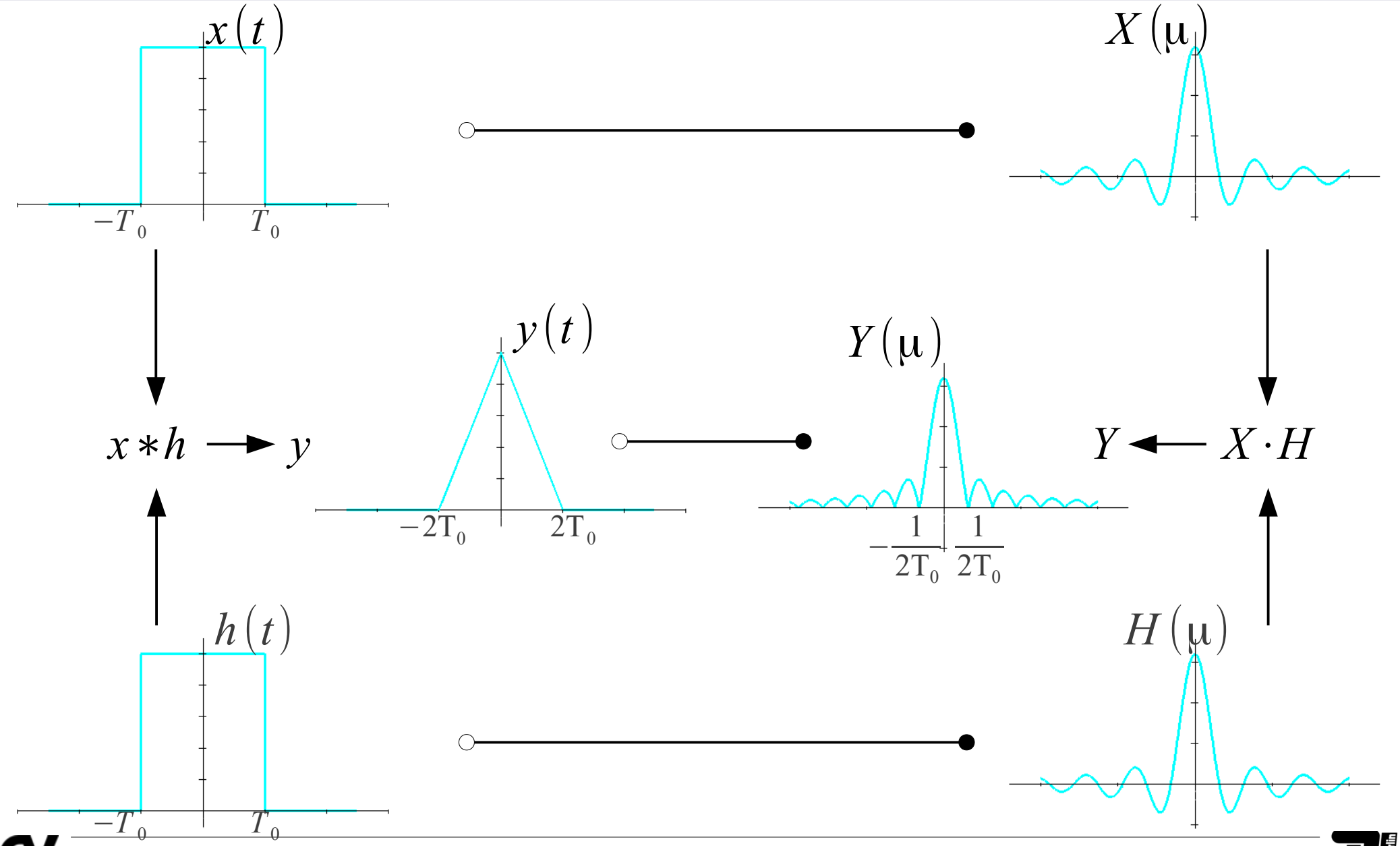
$x * h \rightarrow y$



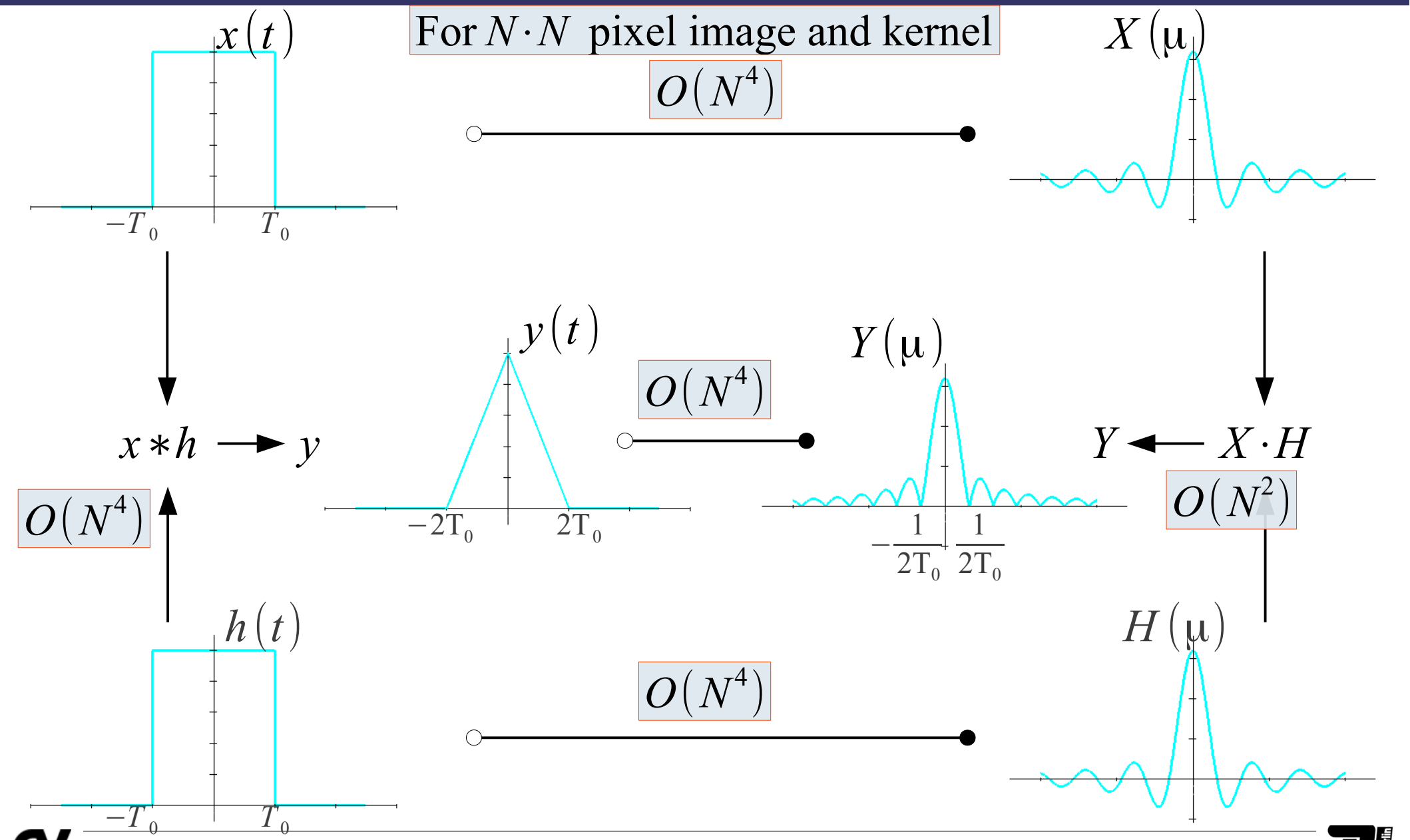




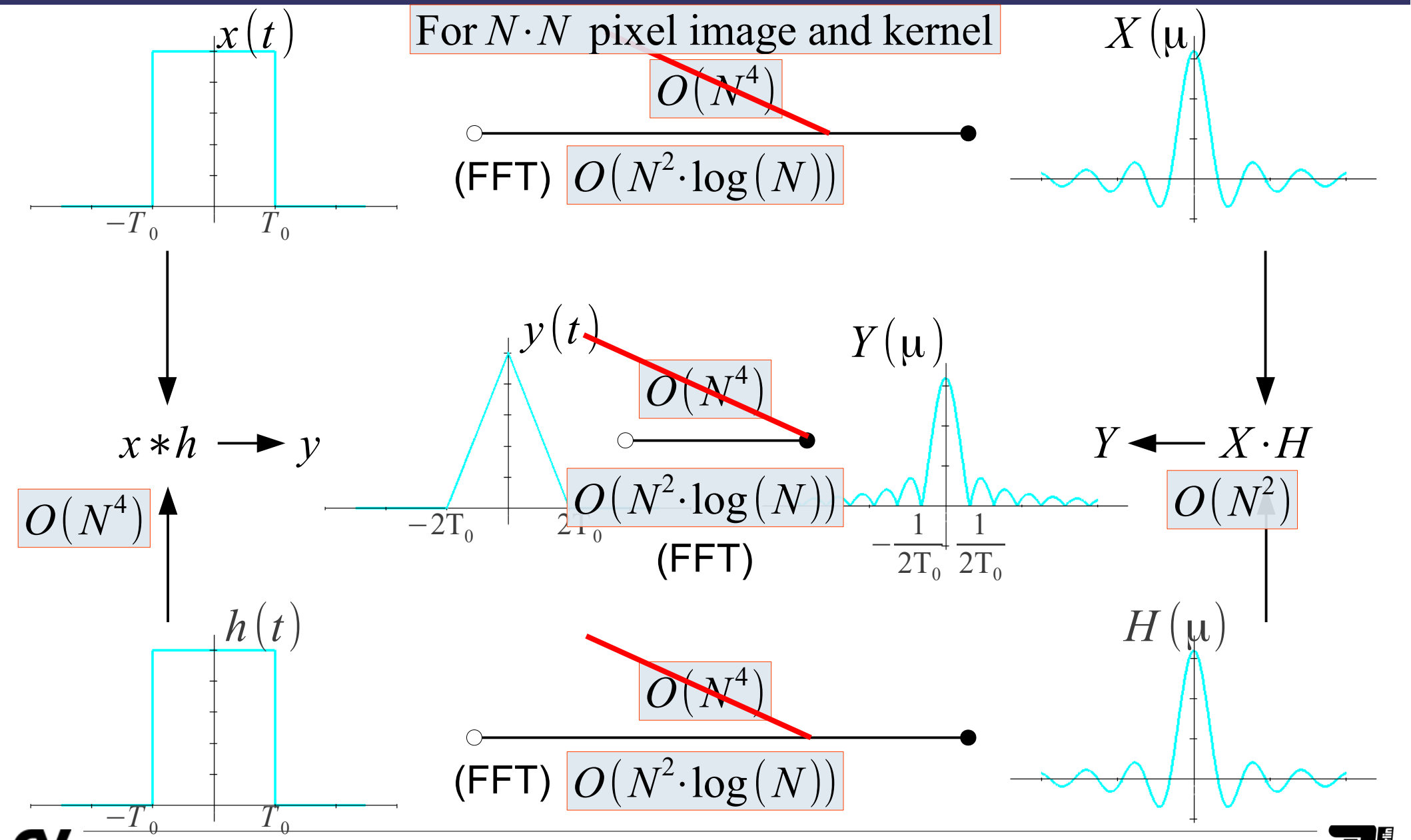
Example (Convolution Theorem)



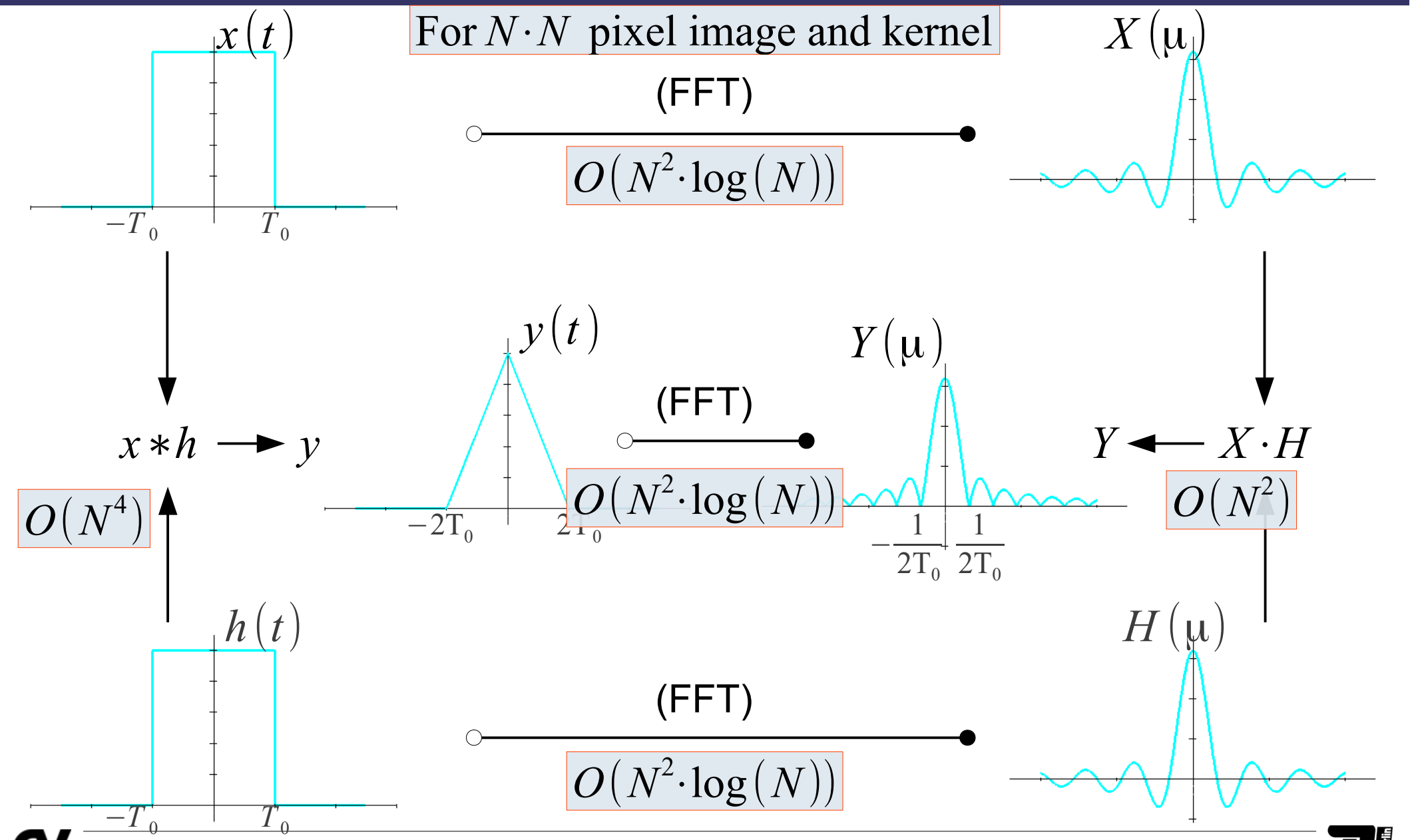
Example (Convolution Theorem)



Example (Convolution Theorem)



Example (Convolution Theorem)



Convolution Theorem

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y(t) \quad \exp(-\imath\mu t)\mathrm{d}t$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp(-\imath\mu t) \left[\int_{-\infty}^{\infty} x(\tau) h(t-\tau)\mathrm{d}\tau \right] \mathrm{d}t$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(\tau) \left[\int_{-\infty}^{\infty} h(t-\tau) \exp(-\imath\mu t)\mathrm{d}t \right] \mathrm{d}\tau$$

$$(s = t - \tau) \rightarrow \quad = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(\tau) \left[\int_{-\infty}^{\infty} h(s) \exp(-\imath\mu(s+\tau))\mathrm{d}s \right] \mathrm{d}\tau$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(\tau) H(\mu) \exp(-\imath\mu\tau)\mathrm{d}\tau$$

$$= H(\mu) \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(\tau) \exp(-\imath\mu\tau)\mathrm{d}\tau = \boxed{H(\mu)\ X(\mu)}$$

Convolution Theorem

1. Kernel and image must have the same size before transformation

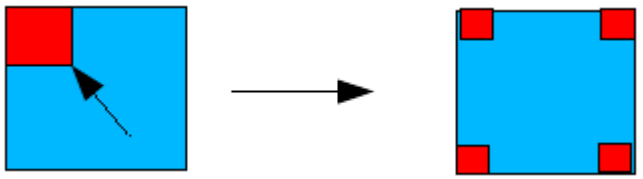
   → Copy the kernel into a larger matrix

Convolution Theorem

1. Kernel and image must have the same size before transformation

   ➔ Copy the kernel into a larger matrix

2. Centre the kernel (shift the central element to F'(1,1))

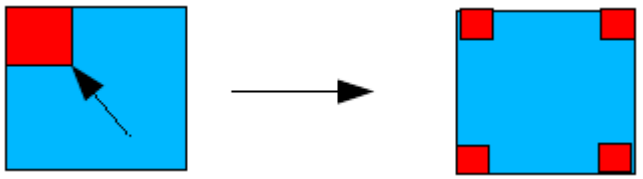


Convolution Theorem

1. Kernel and image must have the same size before transformation

   ➔ Copy the kernel into a larger matrix

2. Centre the kernel (shift the central element to F'(1,1))



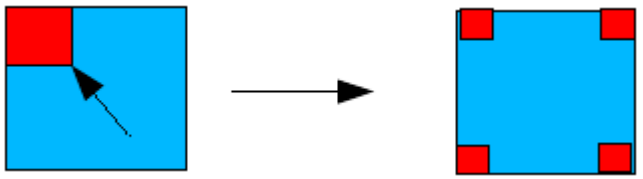
3. Fourier transform image and the centred kernel

Convolution Theorem

1. Kernel and image must have the same size before transformation

   → Copy the kernel into a larger matrix

2. Centre the kernel (shift the central element to F'(1,1))

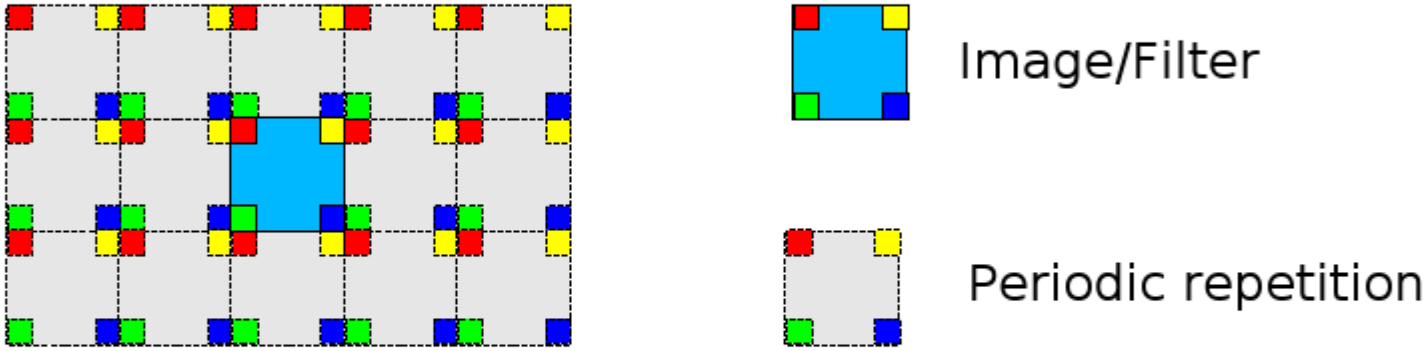


3. Fourier transform image and the centred kernel

4. The spectrum of the result E(x,y) is the element-wise product of the spectra obtained in step 3

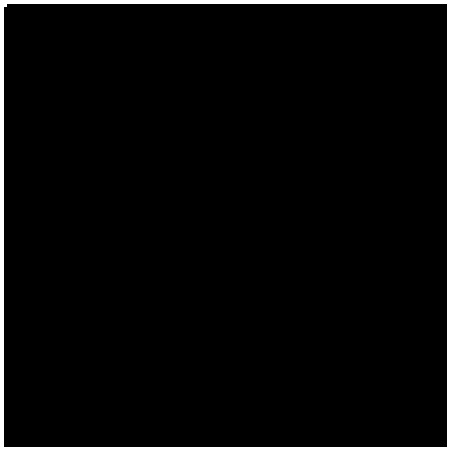
Convolution Theorem

- The DFT describes a periodic function
- When convolving in the frequency domain, it is as though image and filter are repeated to all sides indefinitely:



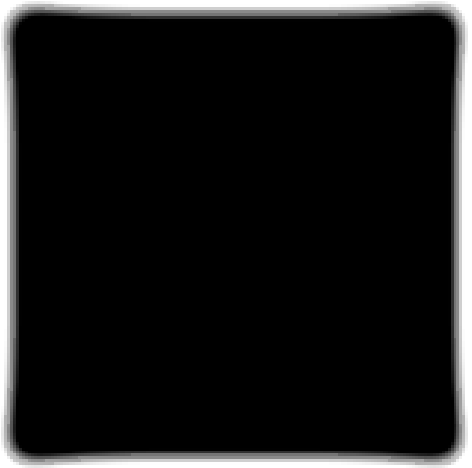
- For an image of resolution (M,N) in x and y, respectively:
  - → The point (-y,-x) is identical to the point (M-y, N-x)
- Elements that lie outside the kernel after centring simply re-appear on the opposite side of the kernel matrix!

Unsharp Masking



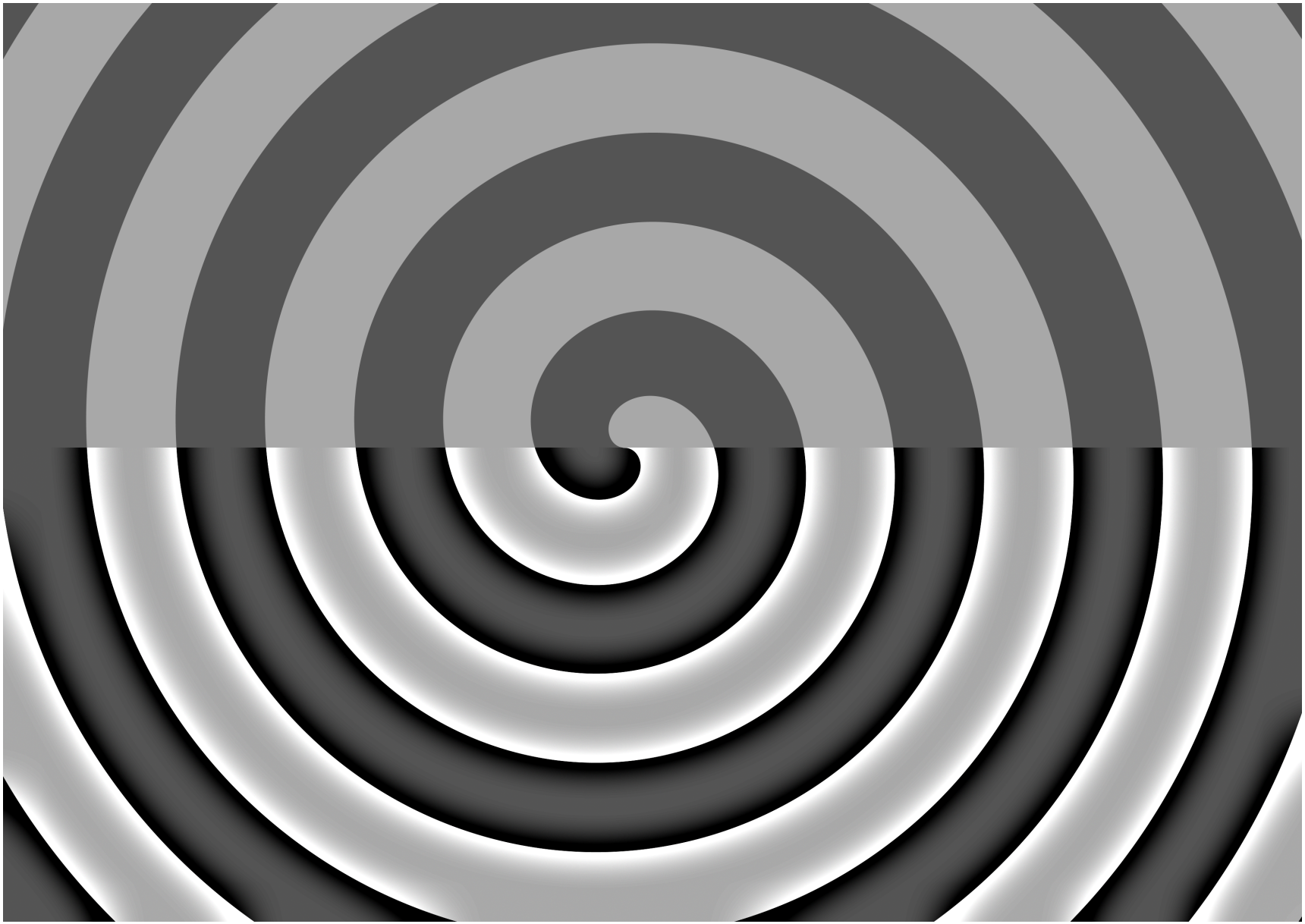
Original image

Distorted image



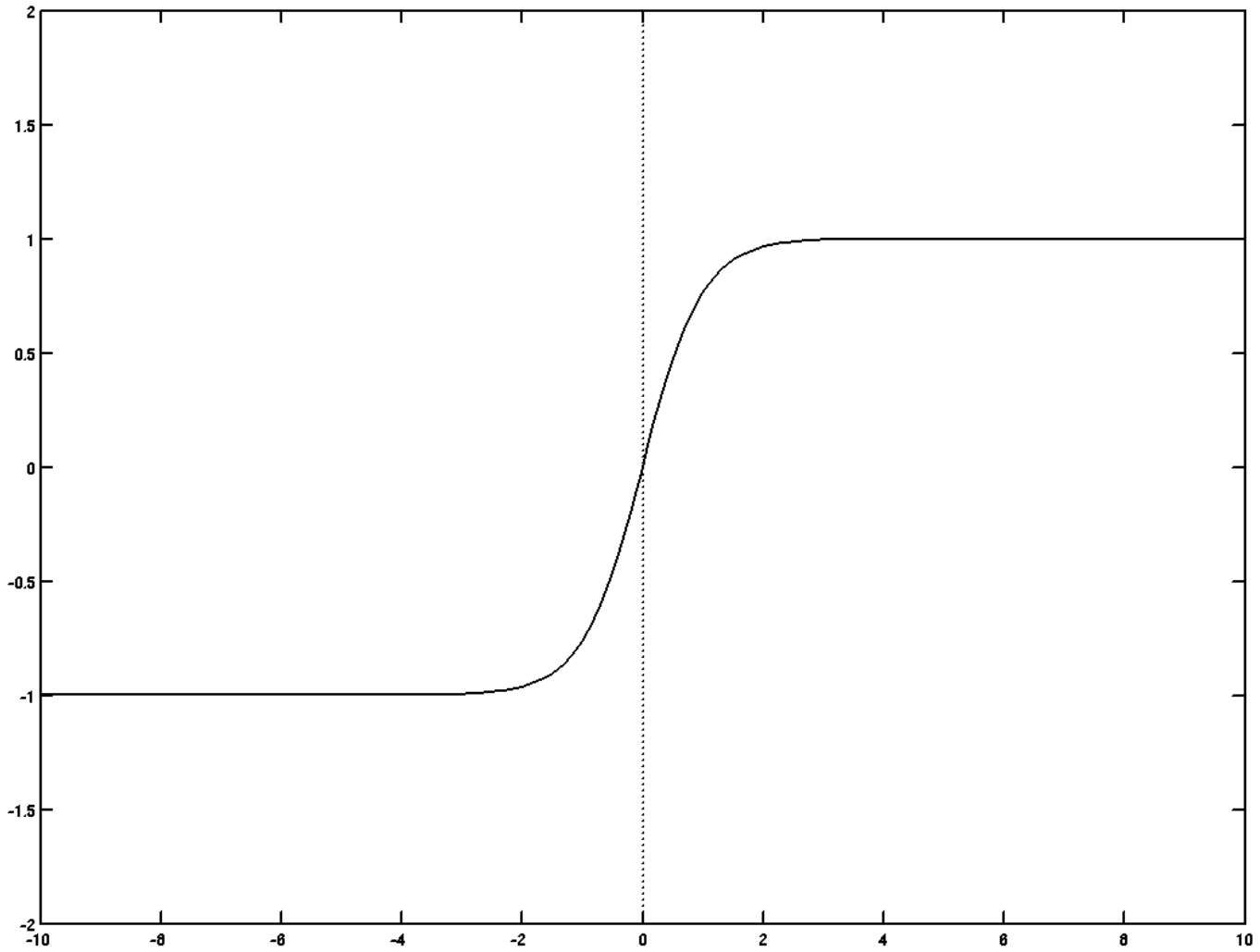
Enhanced image

Unsharp Masking



Unsharp Masking

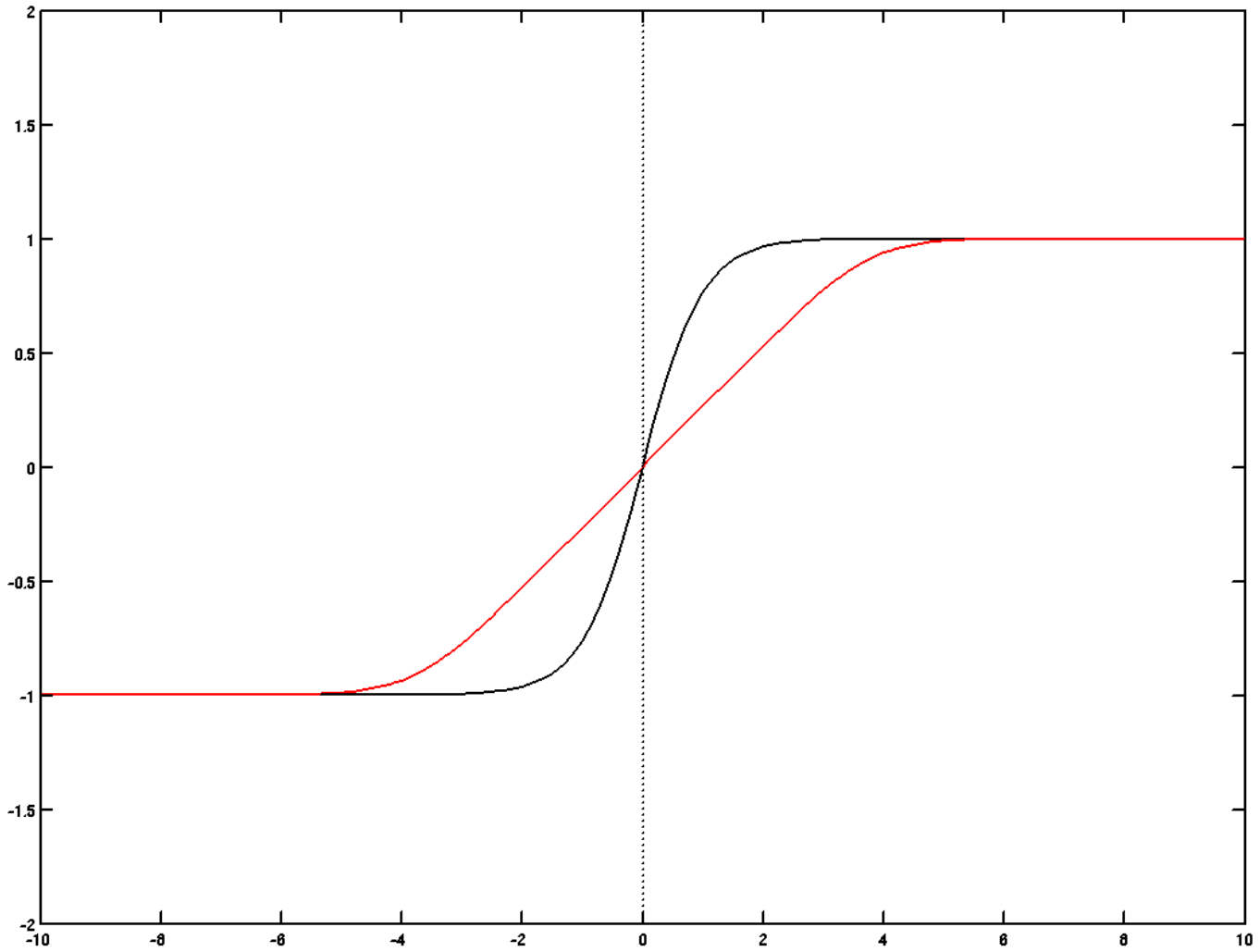
Black: Original edge y



Unsharp Masking

Black: Original edge y

Red: Degraded edge y_0



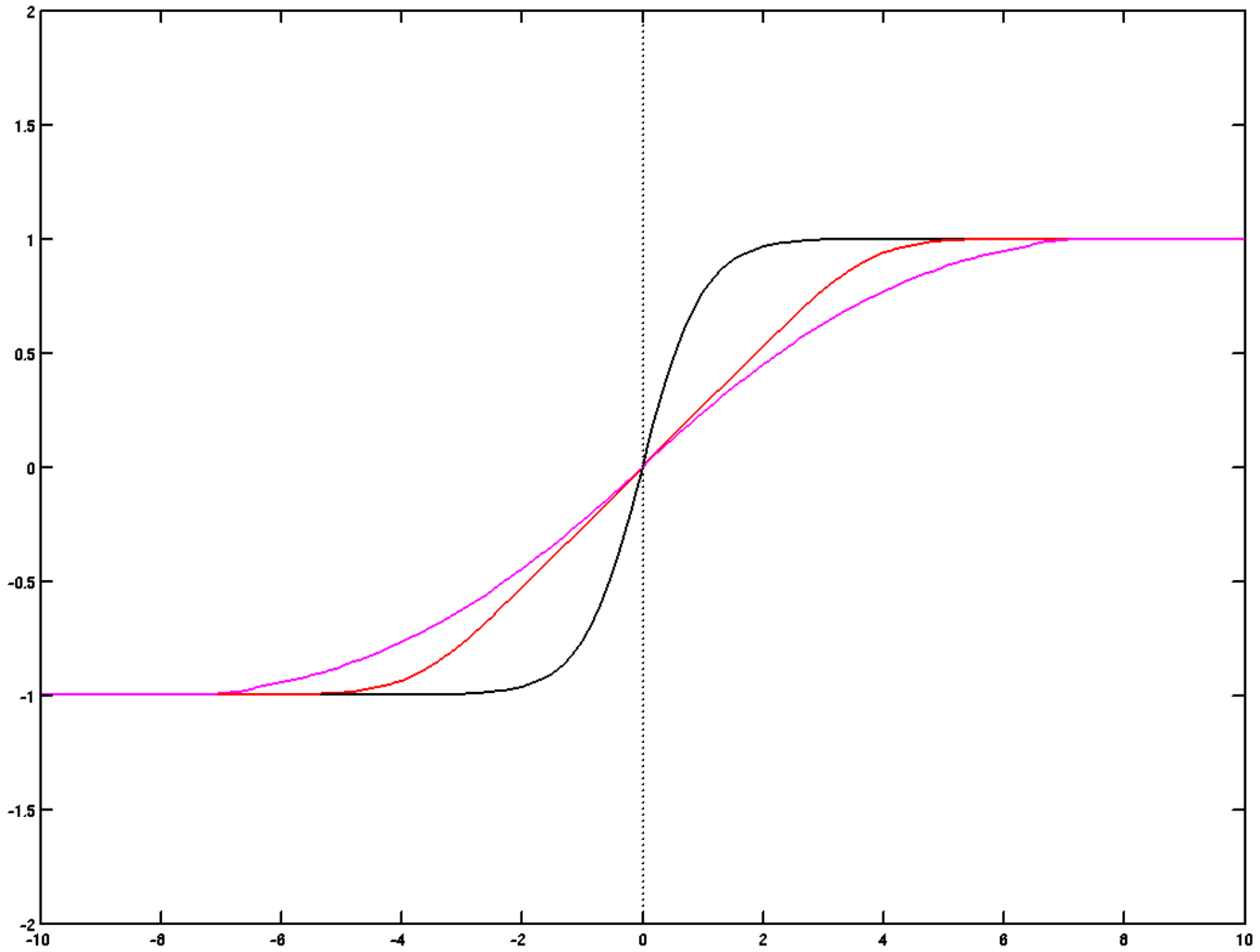
Unsharp Masking

Black: Original edge y

Red: Degraded edge y_0

USM (magenta):

1. Smooth: $y_0 \rightarrow y_1$



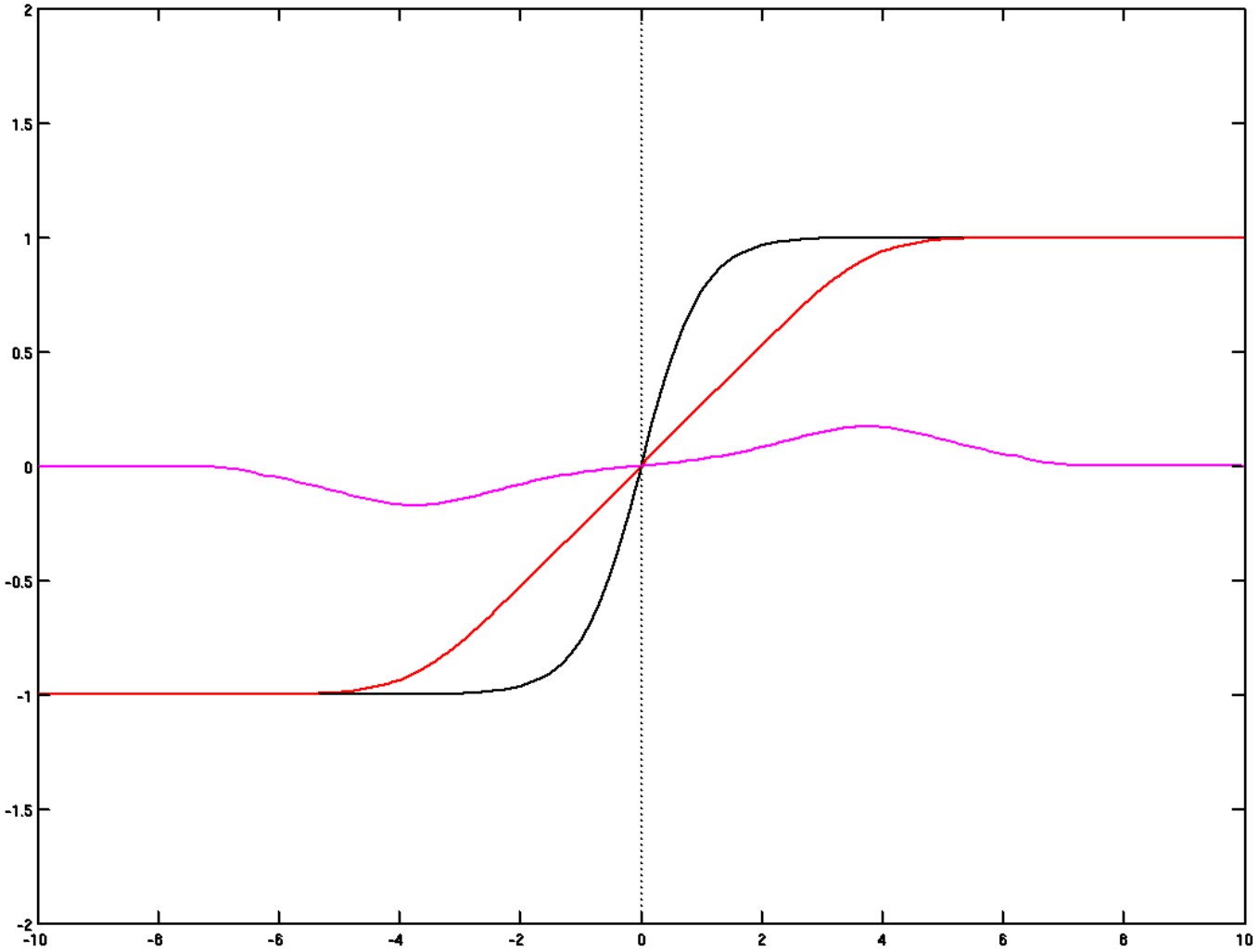
Unsharp Masking

Black: Original edge y

Red: Degraded edge y_0

USM (magenta):

1. Smooth: $y_0 \rightarrow y_1$
2. Subtract: $y_2 = y_0 - y_1$



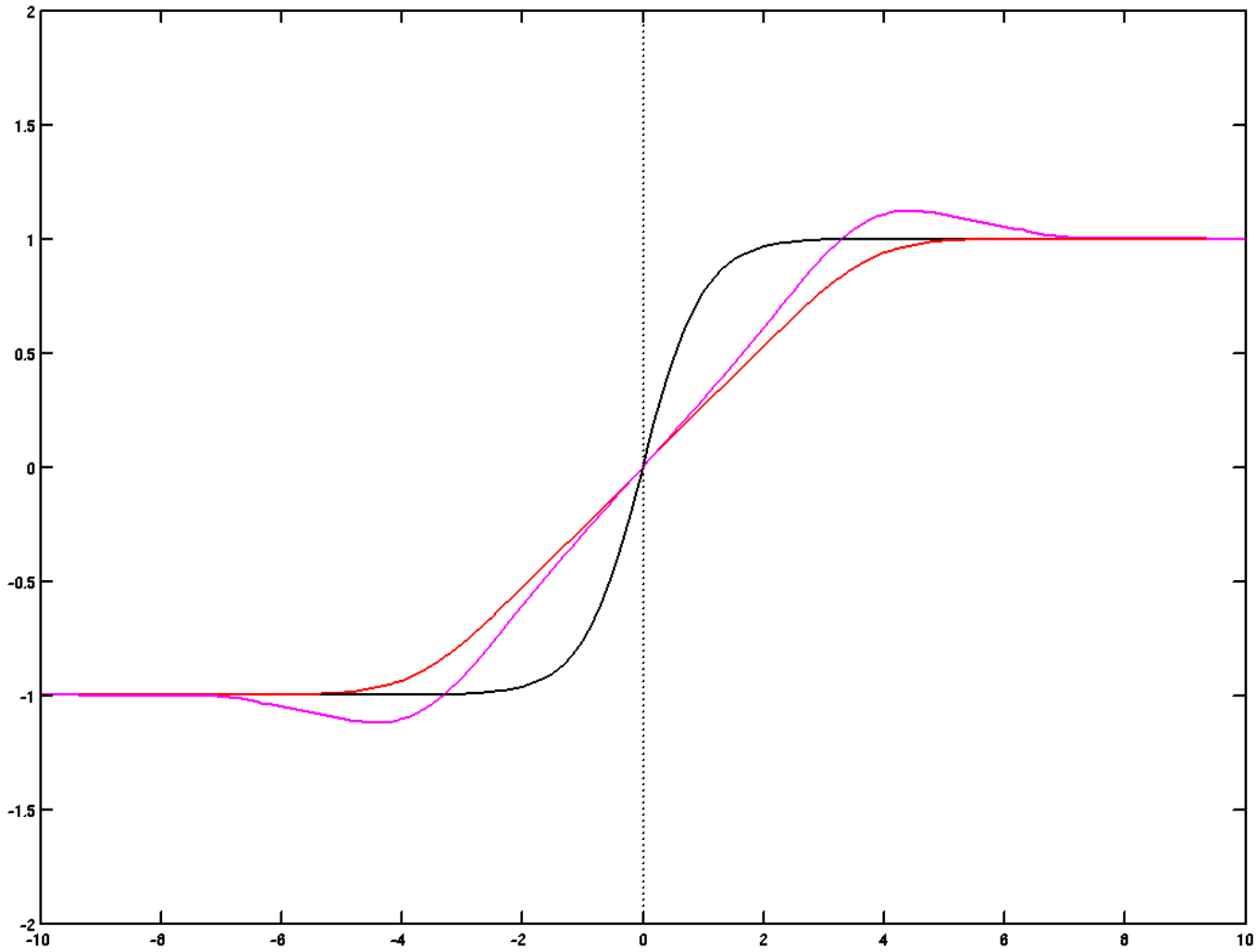
Unsharp Masking

Black: Original edge y

Red: Degraded edge y_0

USM (magenta):

1. Smooth: $y_0 \rightarrow y_1$
2. Subtract: $y_2 = y_0 - y_1$
3. Add: $y_3 = y_0 + y_2$



Unsharp Masking

Black: Original edge y

Red: Degraded edge y_0

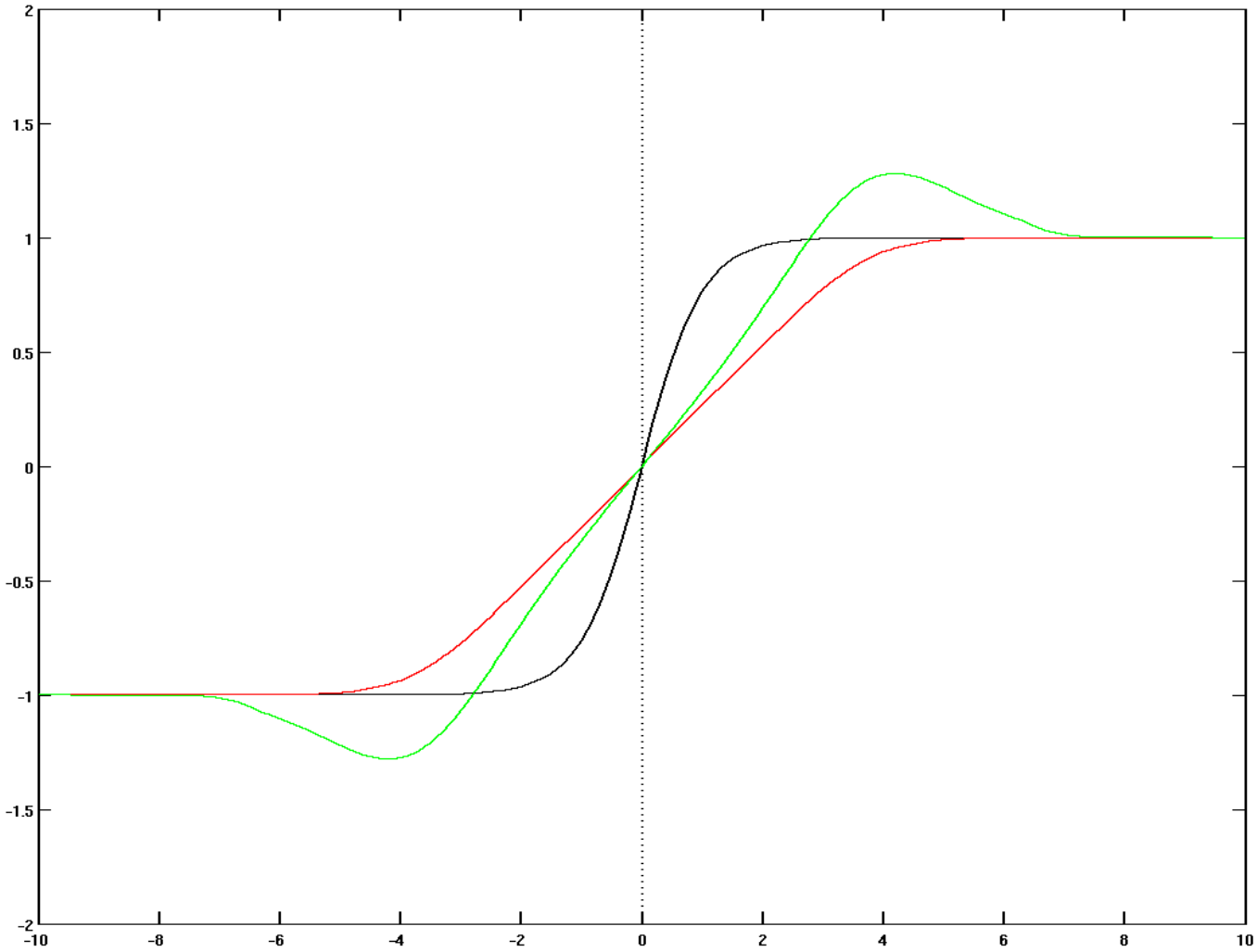
USM (magenta):

1. Smooth: $y_0 \rightarrow y_1$
2. Subtract: $y_2 = y_0 - y_1$

2.5. Scale: $\acute{y}_2 = s * y_2$

3. Add: $y_3 = y_0 + \acute{y}_2$

$\rightarrow$ Final (green)



Unsharp Masking

Original image

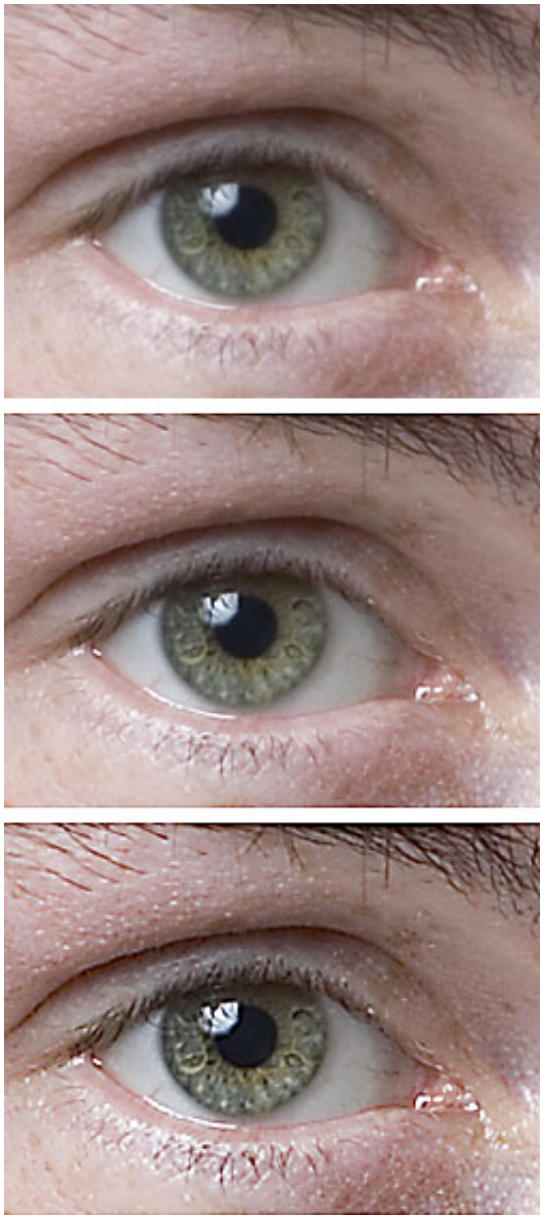
Enhanced image

Unsharp Masking

Original image

Enhanced image
with thresholding

Unsharp Masking



Original image

Sharpened image

Highly sharpened image

[GFDL - https://en.wikipedia.org/wiki/File:Unsharped_eye.jpg]

Unsharp Masking

0. Given: Image I_0, Task: Enhancement

Unsharp Masking

0. Given: Image I_0, Task: Enhancement

1. Smooth image I_0 to obtain I_1
   (1^{st} parameter: kernel size k)

Unsharp Masking

0. Given: Image I_0, Task: Enhancement

1. Smooth image I_0 to obtain I_1
   (1^{st} parameter: kernel size k)

2. Subtract smoothed image I_1 from original I_0
   $I_2 = I_0 - I_1$

Unsharp Masking

0. Given: Image I_0, Task: Enhancement

1. Smooth image I_0 to obtain I_1
   (1^{st} parameter: kernel size k)

2. Subtract smoothed image I_1 from original I_0
   $I_2 = I_0 - I_1$

3. Act only on pixel where difference is larger than threshold T
   If ($|I_2| > T$) then 4.
   (2^{nd} parameter: threshold T)

Unsharp Masking

0. Given: Image I_0, Task: Enhancement

1. Smooth image I_0 to obtain I_1
   (1^{st} parameter: kernel size k)

2. Subtract smoothed image I_1 from original I_0
   $I_2 = I_0 - I_1$

3. Act only on pixel where difference is larger than threshold T
   If ($|I_2| > T$) then 4.
   (2^{nd} parameter: threshold T)

4. Scale difference to further enhance edges
   $I_3 = s*I_2$
   (3^{rd} parameter: scale s)

Unsharp Masking

0. Given: Image I_0, Task: Enhancement

1. Smooth image I_0 to obtain I_1
   (1^{st} parameter: kernel size k)

2. Subtract smoothed image I_1 from original I_0
   $I_2 = I_0 - I_1$

3. Act only on pixel where difference is larger than threshold T
   If ($|I_2| > T$) then 4.
   (2^{nd} parameter: threshold T)

4. Scale difference to further enhance edges
   $I_3 = s*I_2$
   (3^{rd} parameter: scale s)

5. Add to original image
   $I_4 = I_0+I_3$

Unsharp Masking

In short:

Difference image:

$$d(x,y) = i(x,y) - (i(x,y) \otimes m_k)$$

Enhanced image:

$$o(x,y) = \begin{cases} i(x,y) & \text{if} \quad |d(x,y)| < T \\ i(x,y) + s \cdot d(x,y) & \text{else} \end{cases}$$

Parameter:

k, T, s

Unsharp Masking

Slightly improved (in my opinion):

Difference image:

$$d(x,y) = i(x,y) - (i(x,y) \otimes m_k)$$

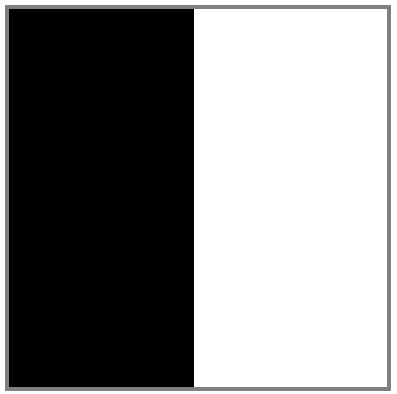
Enhanced image:

$$o(x,y) = \begin{cases} i(x,y) & \text{if} \quad |d(x,y)| < T \\ i(x,y) + s \cdot (d(x,y) - T) & \text{if} \quad d(x,y) > 0 \\ i(x,y) + s \cdot (d(x,y) + T) & \text{if} \quad d(x,y) < 0 \end{cases}$$

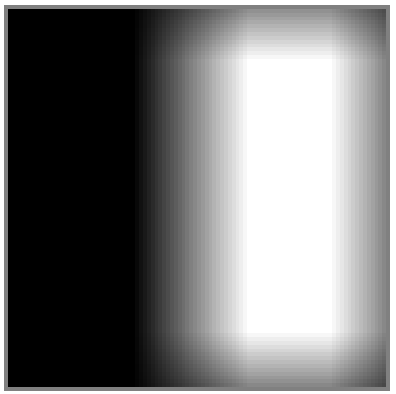
Parameter:

k, T, s

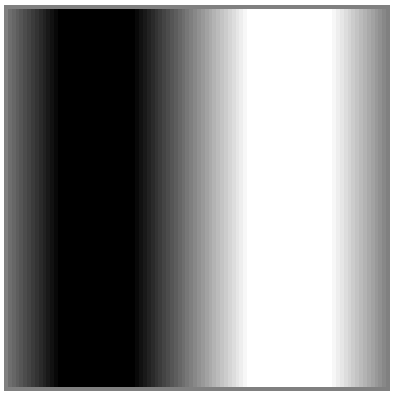
3. Exercise – Theory



a)



b)



c)

Figure 1

A moving average filter was applied to the image in Figure 1a).

Figure 1b) shows the result, if the convolution is carried out in spatial domain,
Figure 1c) if the convolution is carried out as multiplication in frequency domain.

i) Explain which assumptions lead to the “unexpected” border values in each image and why they are different for both methods.

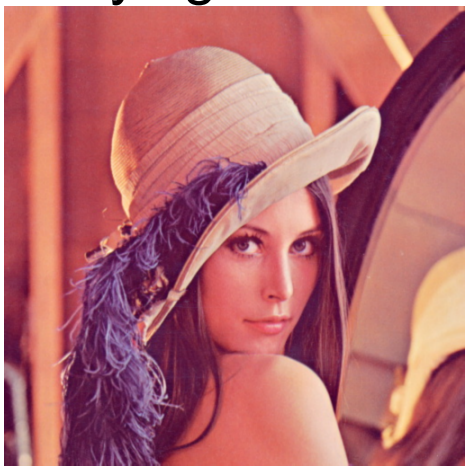
ii) What steps are necessary for the convolution in spatial domain to produce the result in Fig. 1c)?

iii) What steps are necessary for the convolution by multiplication in frequency domain to produce the result in Fig. 1b)?

3. Exercise – Given

FILE: main.cpp
int main(int argc, char argv)**

- Declares variables
- Displays and saves images
- Runs unsharp masking, using various smoothing functions:
  - convolution in spatial domain
  - separable convolution in spatial domain
  - convolution by multiplication in frequency domain
- ➔ Measures and saves time to csv files
  - ➔ For varying image sizes
  - ➔ And for varying kernel sizes



3. Exercise – Given

`Mat Dip3::smoothImage(Mat& in, int size, FilterMode filterMode)`

`in`: input image

`size`: size of filter kernel

`type`: whether or not working in spatial domain

`return`: smoothed image

- Applies Gaussian(-ish) blurring to image
- Switches between the different implementations

`FILE: unit_test.cpp`

→ Simple test function to check for basic correctness

3. Exercise – Reuse!

`Mat Dip3::spatialConvolution(Mat& src, Mat& kernel)`

- Parameter:
  - **`src`** : source image
  - **`kernel`** : kernel of the convolution
  - **`return`** : output image
- Applies convolution in spatial domain
- One method of border handling: size(src) == size(return)
- Do **NOT** use convolution functions of OpenCV

3. Exercise – Reuse!

```
Mat Dip3::separableFilter(Mat& src, Mat &kernel)
```

- Parameter:
  - **`src`** : source image
  - **`kernel`** : 1D kernel of the convolution
  - **`return`** : output image

- Applies convolution in spatial domain
- Performs two consecutive 1D smoothing operations:
  - Blur input image horizontally with kernel, write transposed to temp image
  - Blur temp image horizontally with kernel, write transposed to output image

- One method of border handling: size(src) == size(return)

- Do **NOT** use convolution functions of OpenCV

3. Exercise – To Do

`Mat Dip3::createGaussianKernel1D(int kSize)`

`kSize`: the kernel size

`return`: the generated filter kernel

- Calculates Gaussian filter kernels

$$f(x) = \frac{1}{2\pi\sigma_x} \exp\left(-\frac{1}{2}\left[\frac{(x-\mu_x)^2}{\sigma_x^2}\right]\right)$$

  - Mean and variance are determined by kernel size (e.g. $\sigma = kSize/5$)
  - Maximal value at centre
  - Must integrate to one!

3. Exercise – To Do

`Mat Dip3::createGaussianKernel2D(int kSize)`

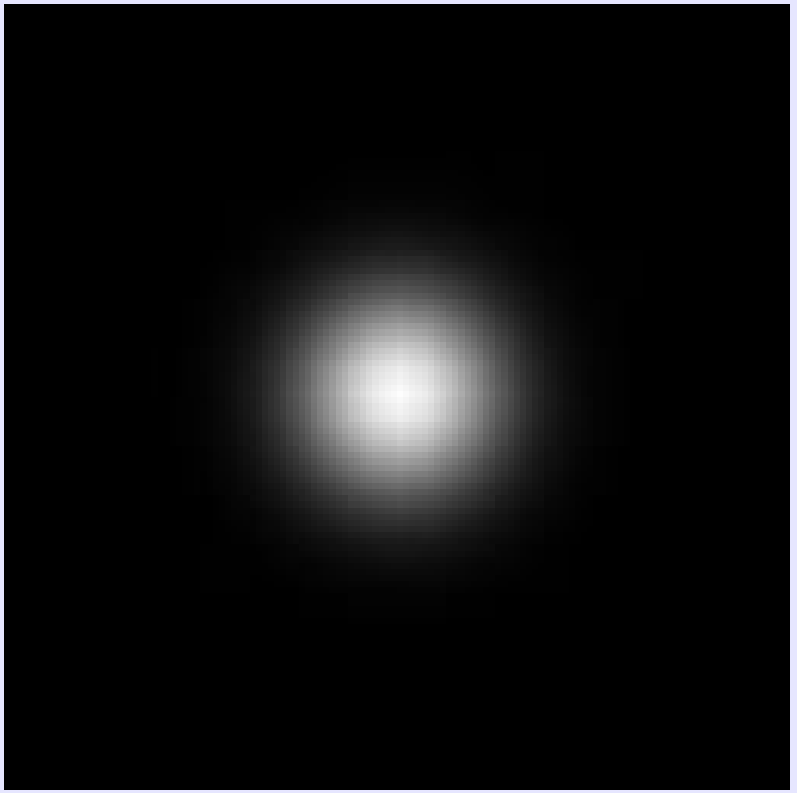
`kSize`: the kernel size

`return`: the generated filter kernel

- Calculates Gaussian filter kernels

$$f(x,y)=\frac{1}{2\pi\sigma_x\sigma_y}\exp\left(-\frac{1}{2}\left[\frac{(x-\mu_x)^2}{\sigma_x^2}+\frac{(y-\mu_y)^2}{\sigma_y^2}\right]\right)$$

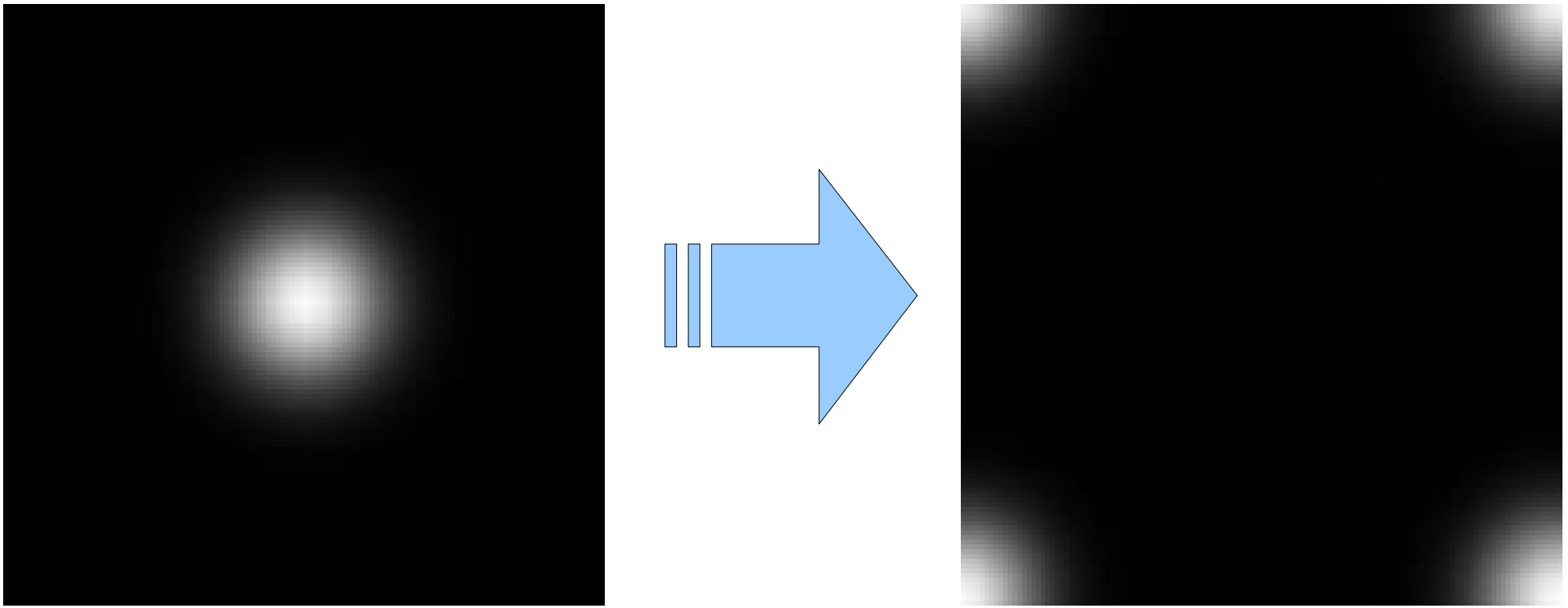
  - Mean and variance are determined by kernel size (e.g. $\sigma = kSize/5$)
  - Maximal value at centre
  - Must integrate to one!



3. Exercise – To Do

```
Mat Dip3::circShift(Mat& in, int dx, int dy)
```

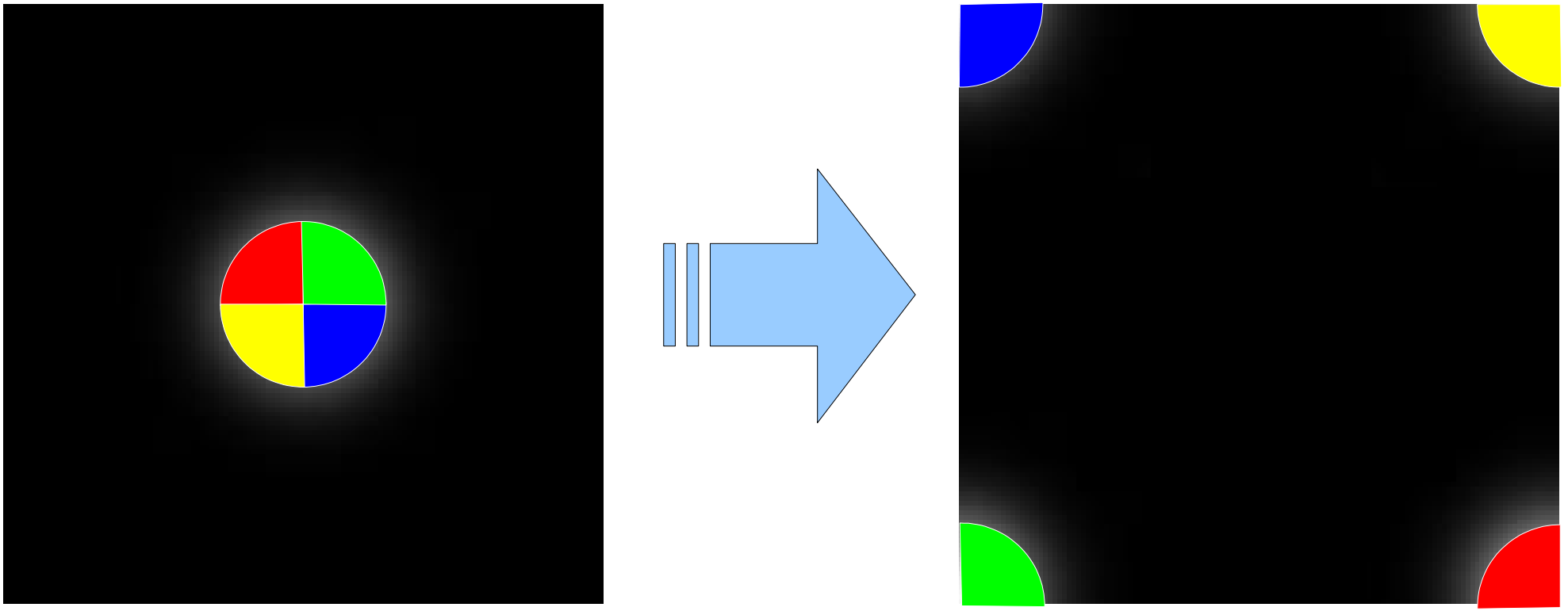
- Performes circular shift in (dx,dy) direction



3. Exercise – To Do

```
Mat Dip3::circShift(Mat& in, int dx, int dy)
```

- Performes circular shift in (dx,dy) direction



3. Exercise – To Do

`Mat Dip3::frequencyConvolution(Mat& in, Mat& kernel)`

- Calculates convolution by exploiting convolution theorem
- Forward transform:
  dft(Mat, Mat, 0);
- Inverse transform
  dft(Mat, Mat, DFT_INVERSE + DFT_SCALE);
- Spectrum multiplication
  mulSpectrums(Mat, Mat, Mat, 0);

Re Y_{00}	Re Y_{01}	Im Y_{01}	Re Y_{02}	Im Y_{02}	. . .
Re Y_{10}	Re Y_{11}	Im Y_{11}	Re Y_{12}	Im Y_{12}	. . .
Re Y_{20}	Re Y_{21}	Im Y_{21}	Re Y_{22}	Im Y_{22}	. . .
. . .	. . .	. . .	. . .	. . .	. . .

3. Exercise – To Do

Mat Dip3::usm(Mat& in, FilterMode filterMode, int size, double thresh, double scale)

`in`: input image

`filterMode`: smoothing mode (just call smoothImage(…) to delegate)

`size`: kernel size k

`thresh`: threshold T

`scale`: scale s

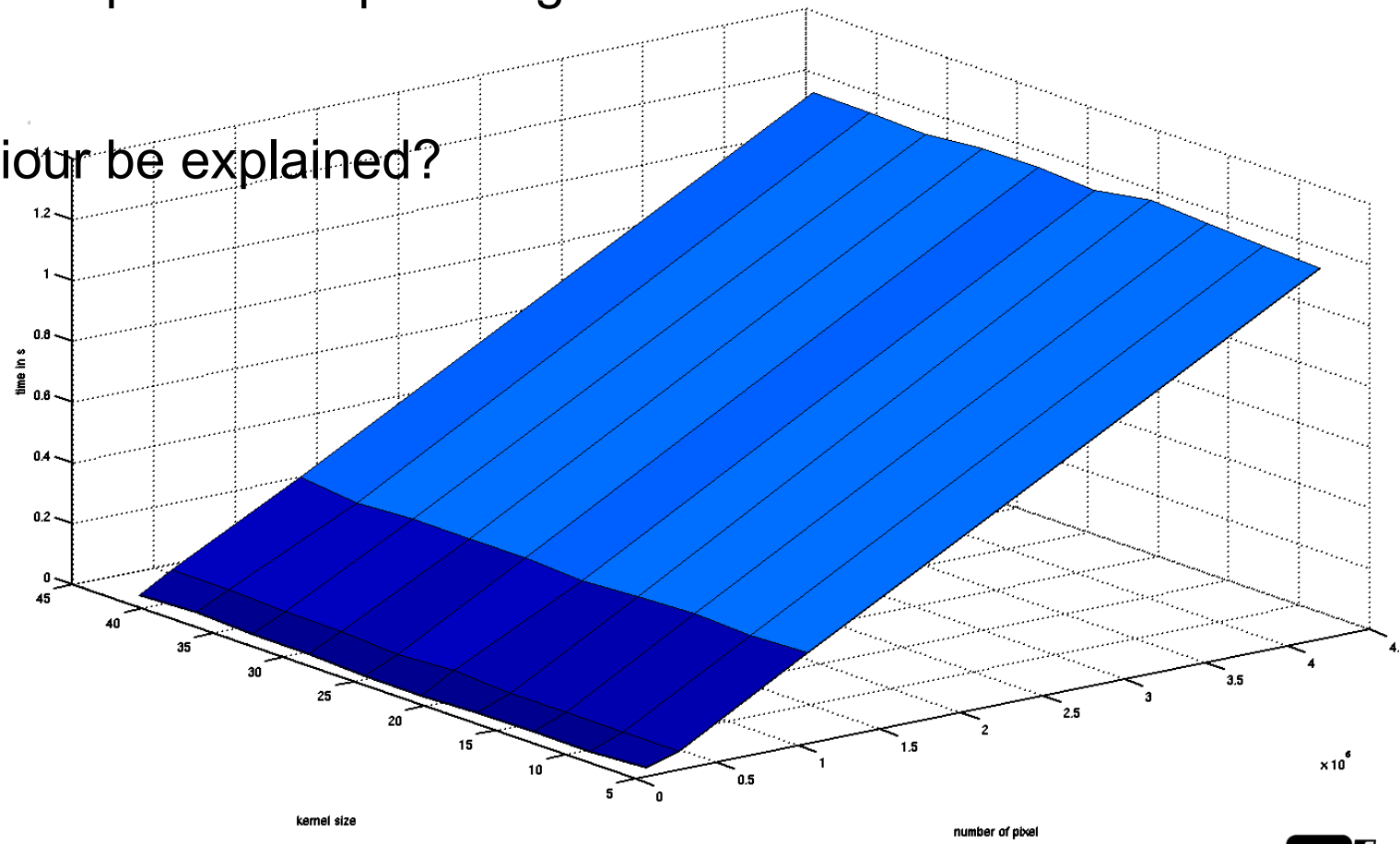
`return`: result

- Unsharp masking using smoothed image
- Thresholding to ensure valid image ranges [0,255]
  - Useful functions: threshold(..)

3. Exercise – To Do

- Prepare three graphs that describe time behaviour of convolution by usage of spatial, separable spatial, and frequency domain
  - Software is your choice (eg. Matlab, Excel, OpenOffice-Calc, ...)
  - x-direction: filter size
  - y-direction: number of pixels in input image
  - z-direction: time

- How can this behaviour be explained?



3. Exercise – Optional

Integral images

→ Integral image computation available in OpenCV
→ Approximate smoothing by box filter
→ Measure time
  → How does time performance change?

→ What are the effects on UnsharpMasking?
→ Improve Gaussian approximation:
  → use multiple box filters of different size

Next Meeting

Deadline: 01.12.2021