

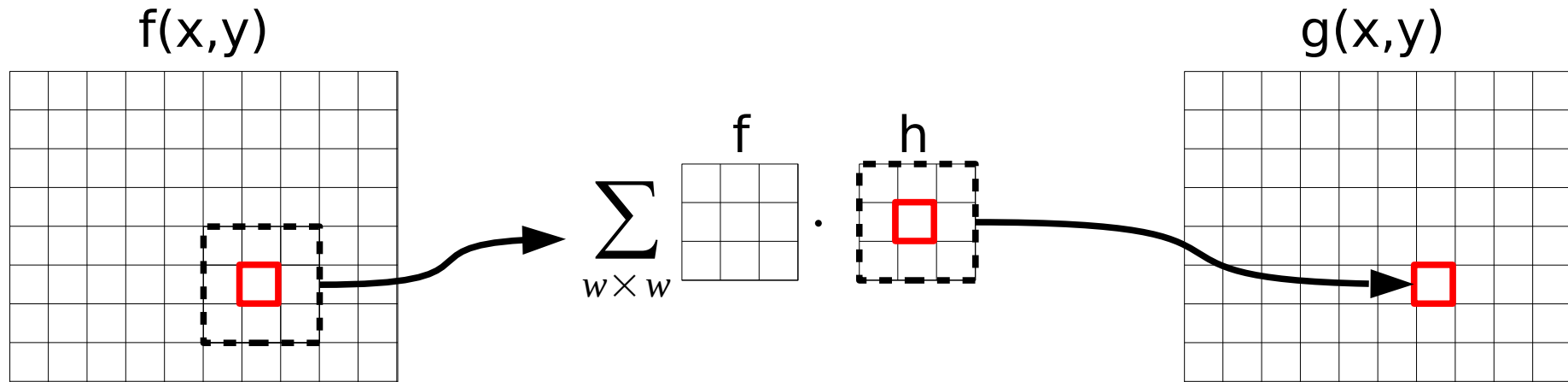
Digital Image Processing

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



Spatial Domain Convolution

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



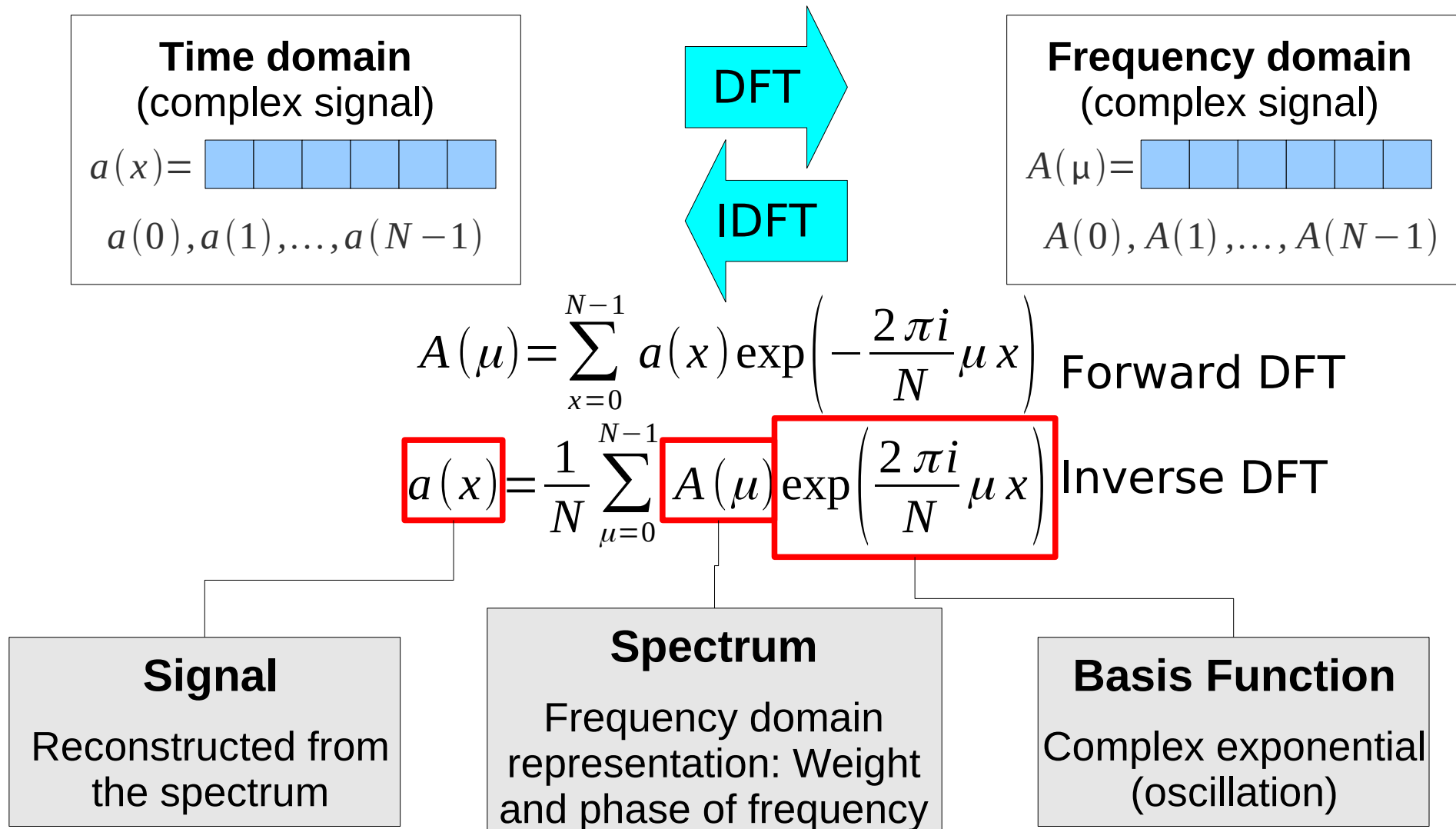
Runtime Complexity:

→ Assuming kernel size == image size

→ $O(N^4)$

Frequency Domain

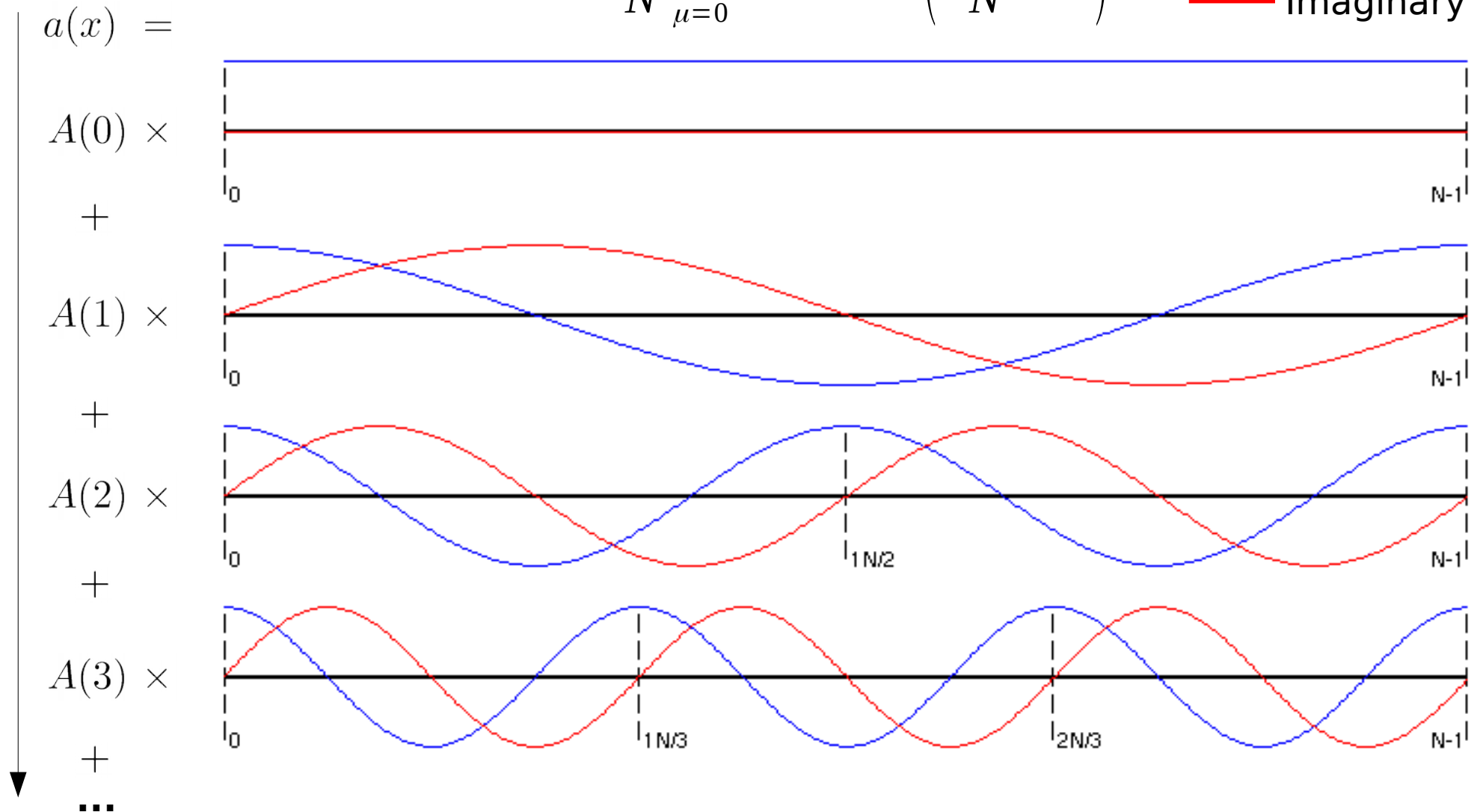
The Discrete Fourier Transform



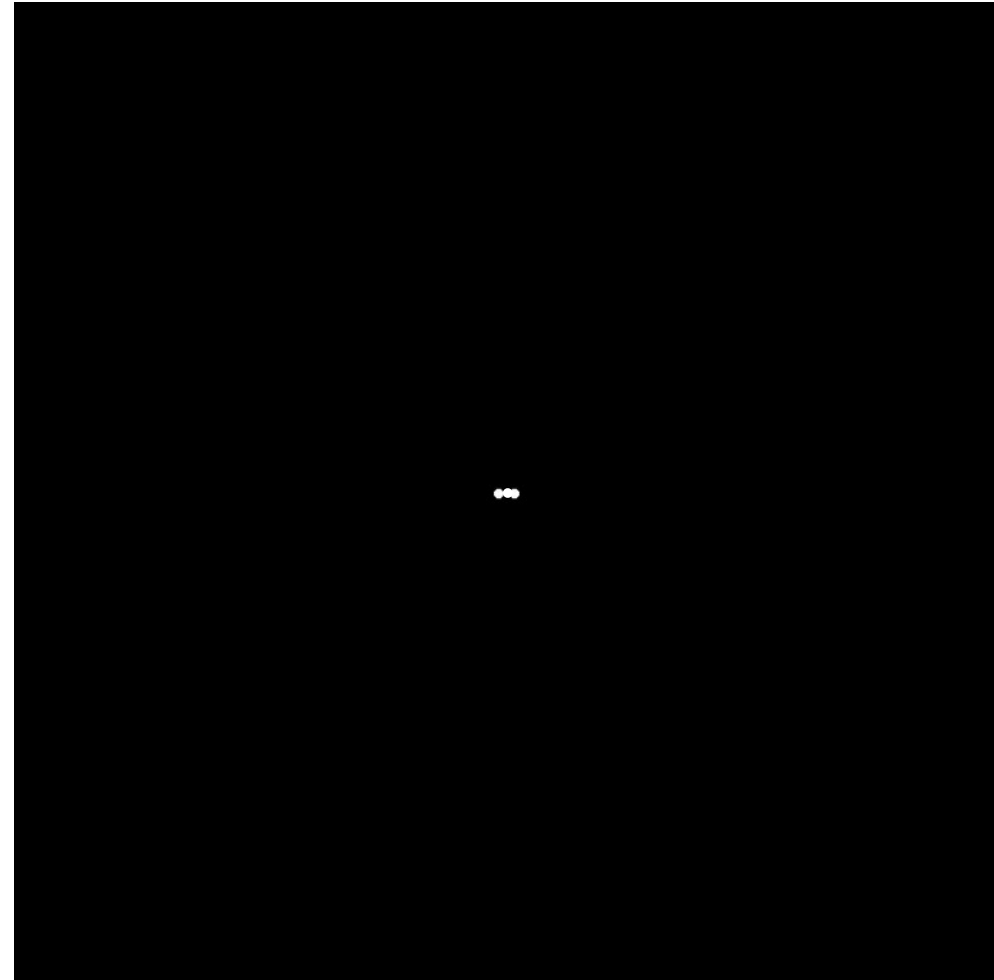
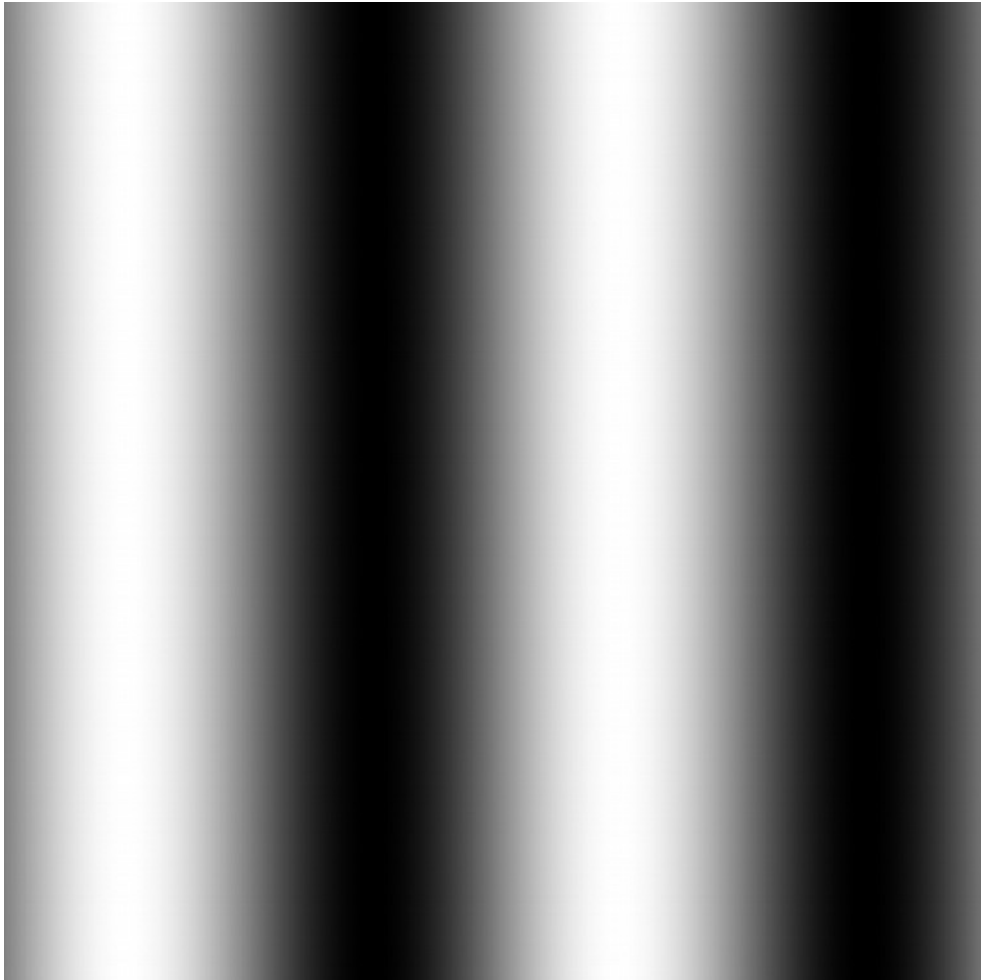
Frequency Domain

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

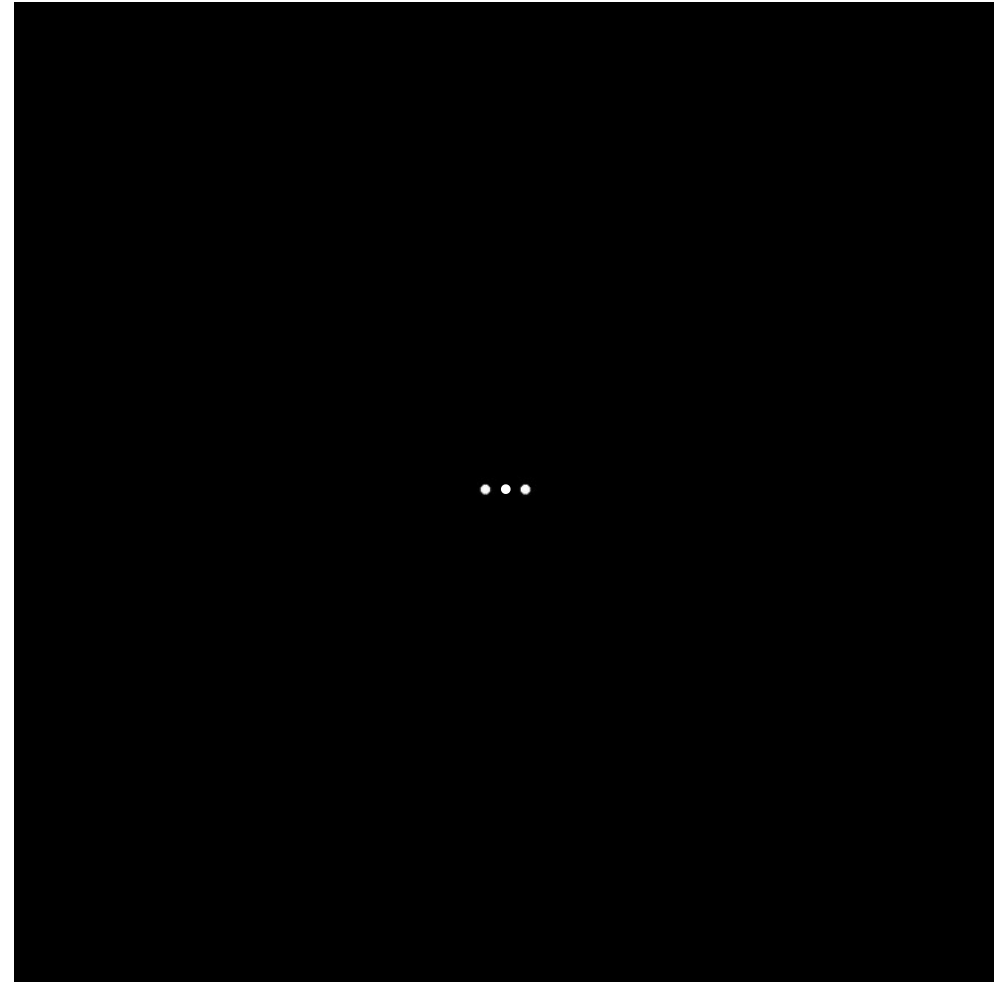
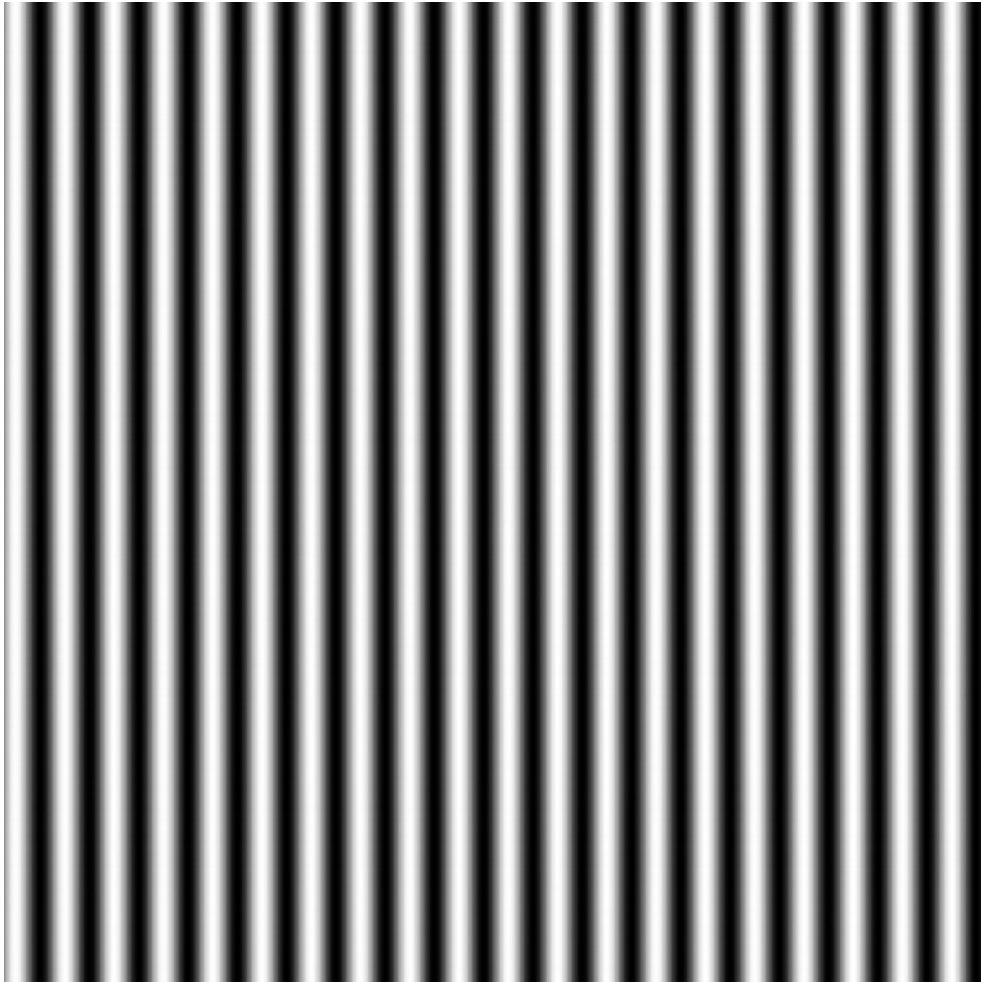
— Real
— Imaginary



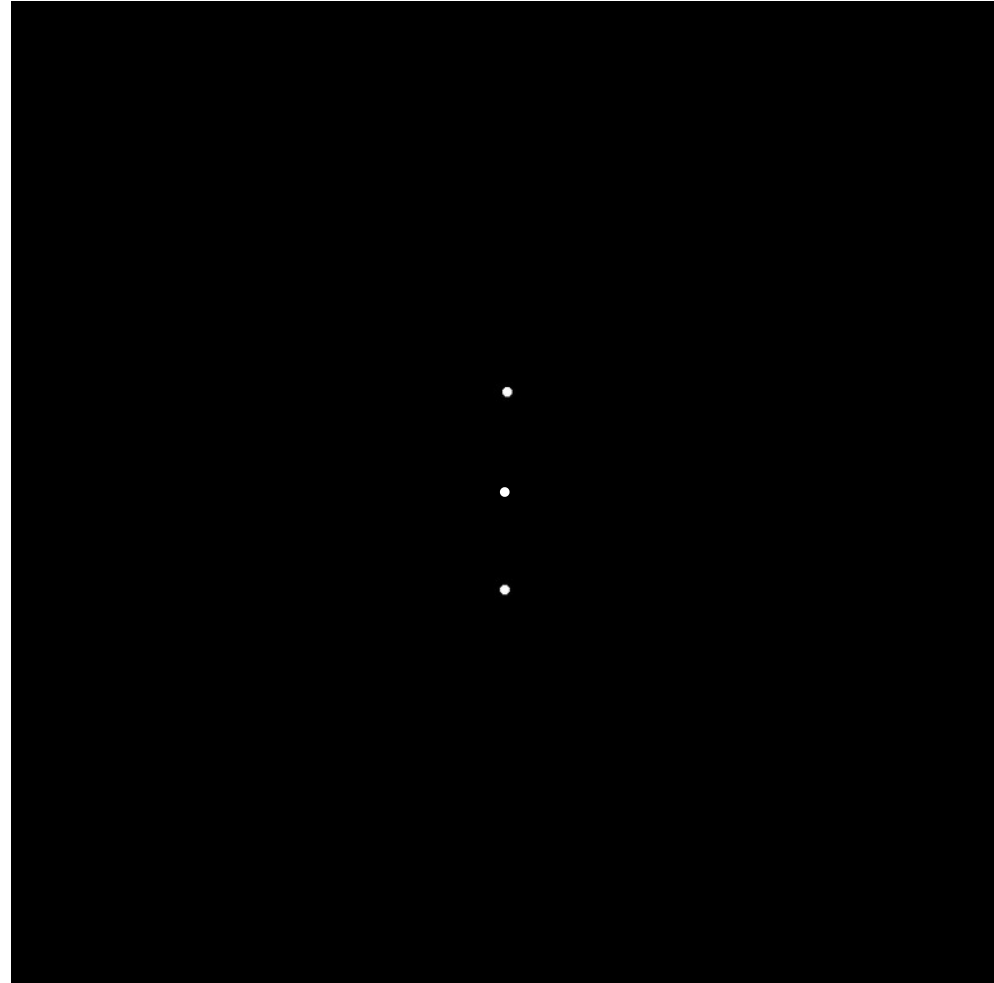
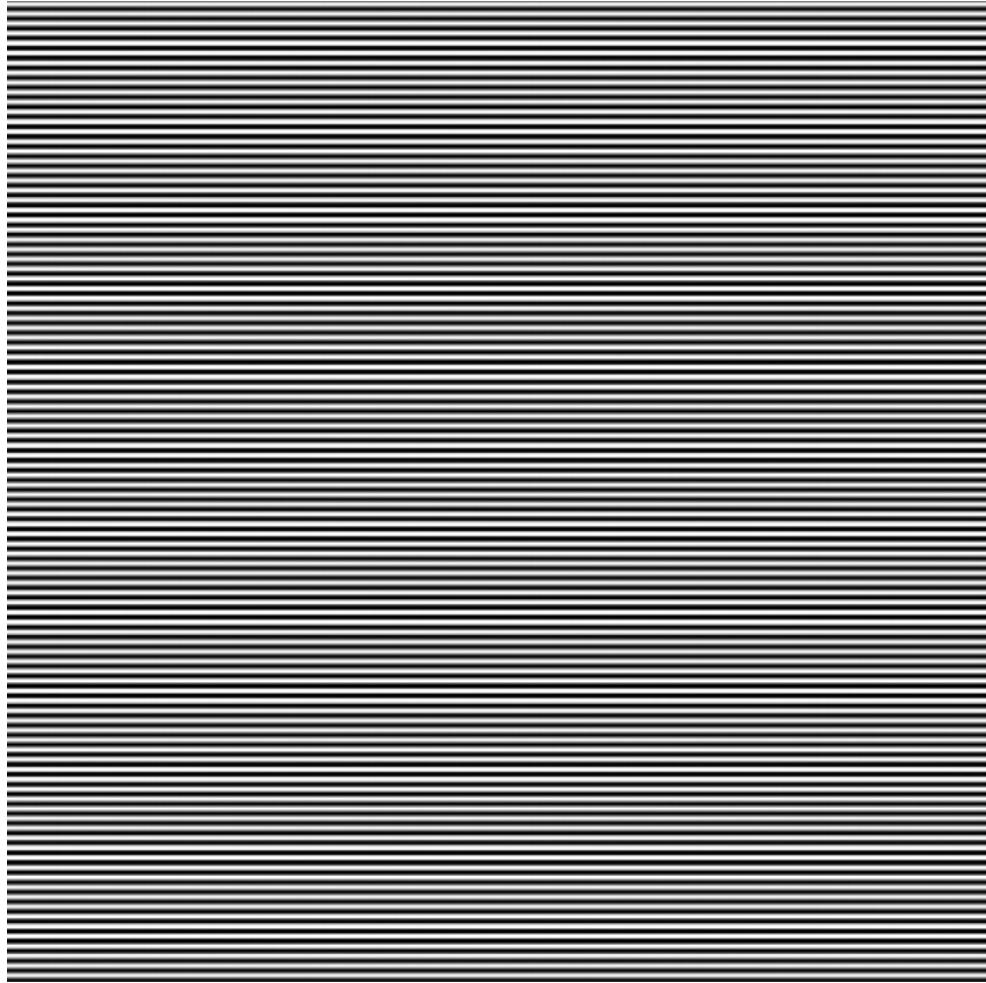
Frequency Domain



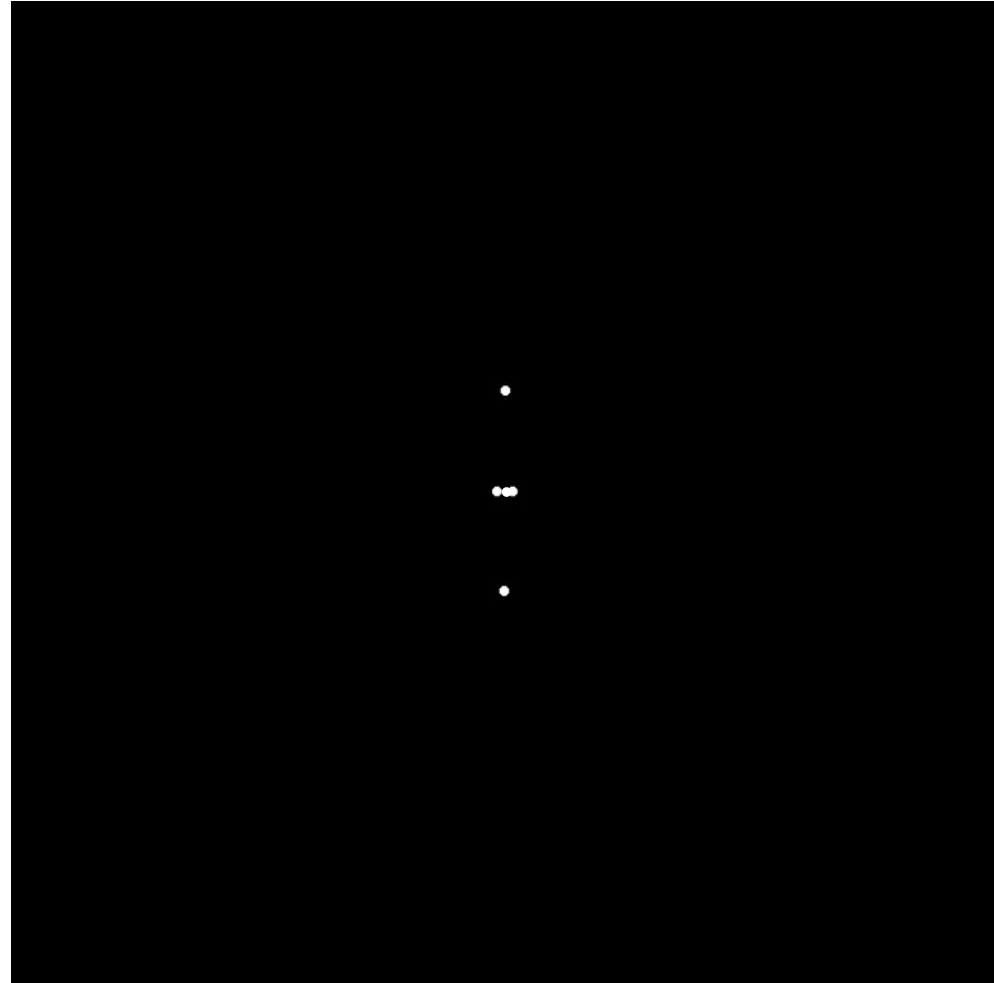
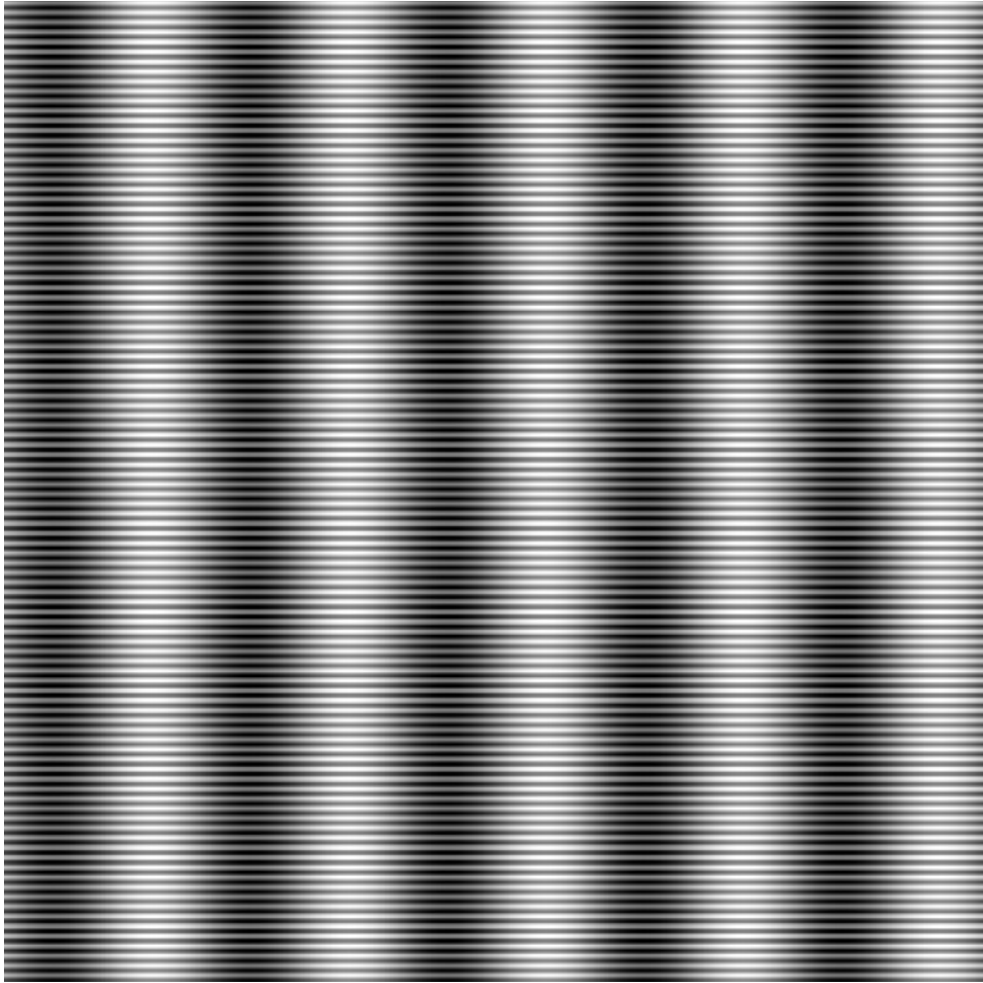
Frequency Domain



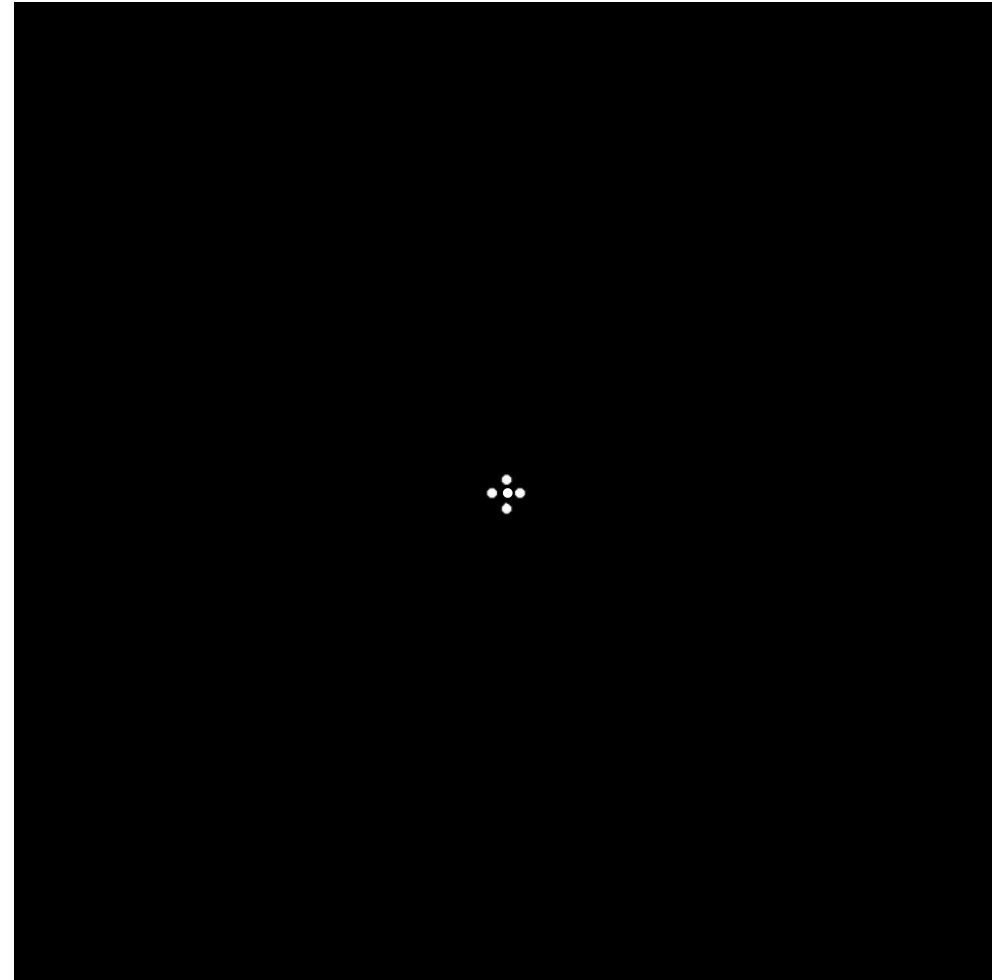
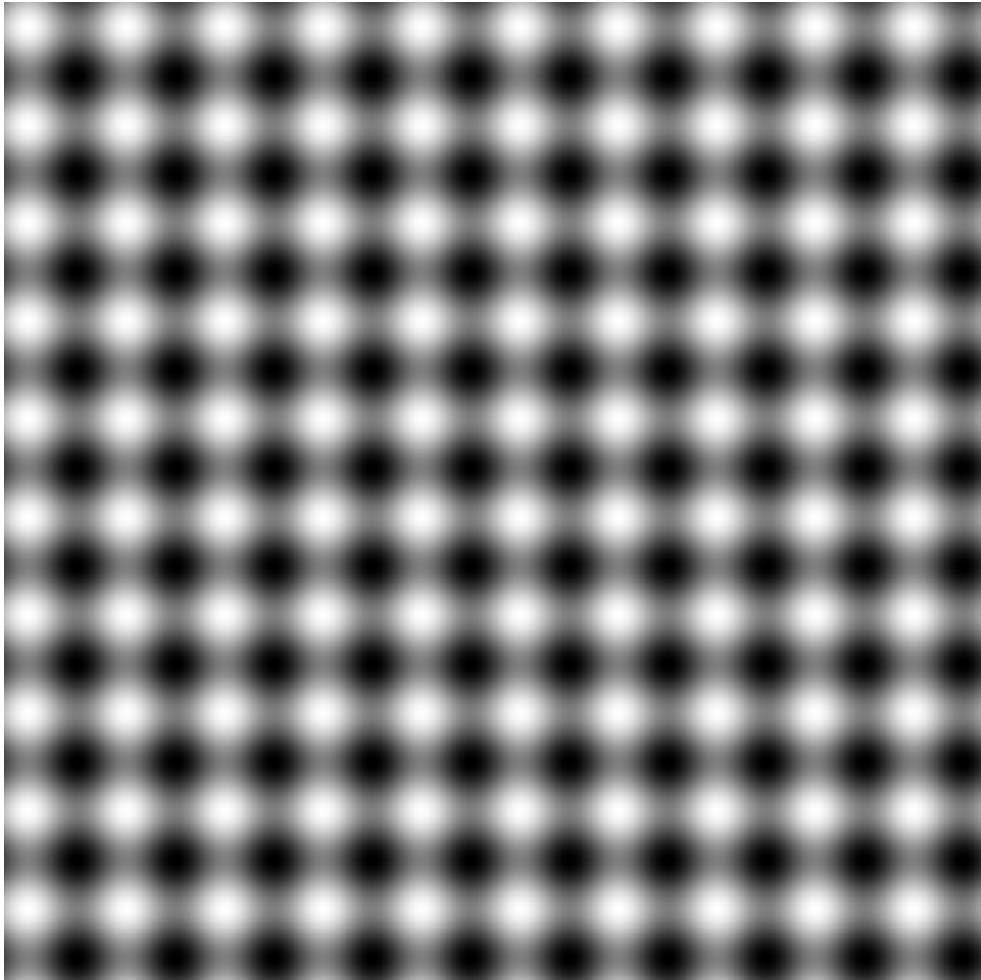
Frequency Domain



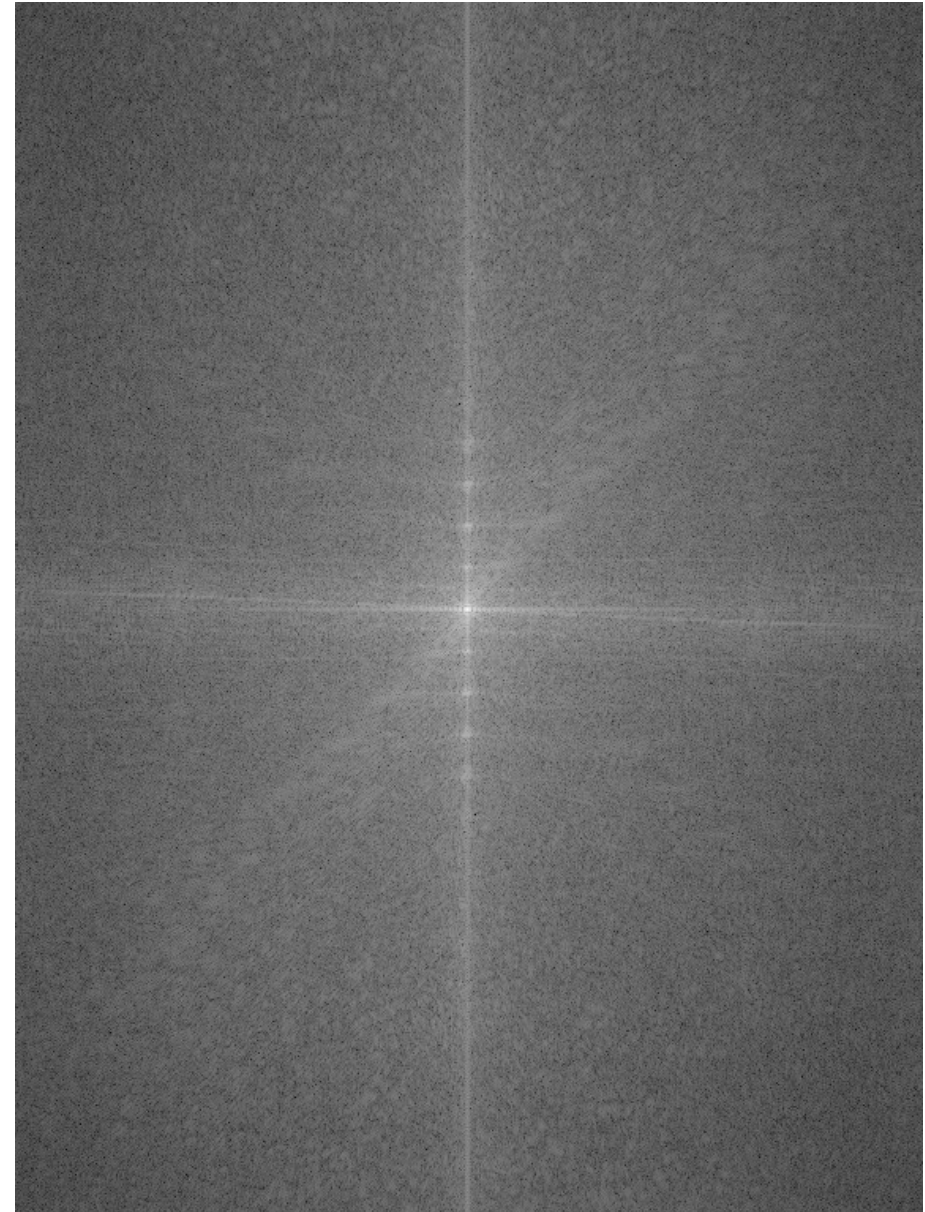
Frequency Domain



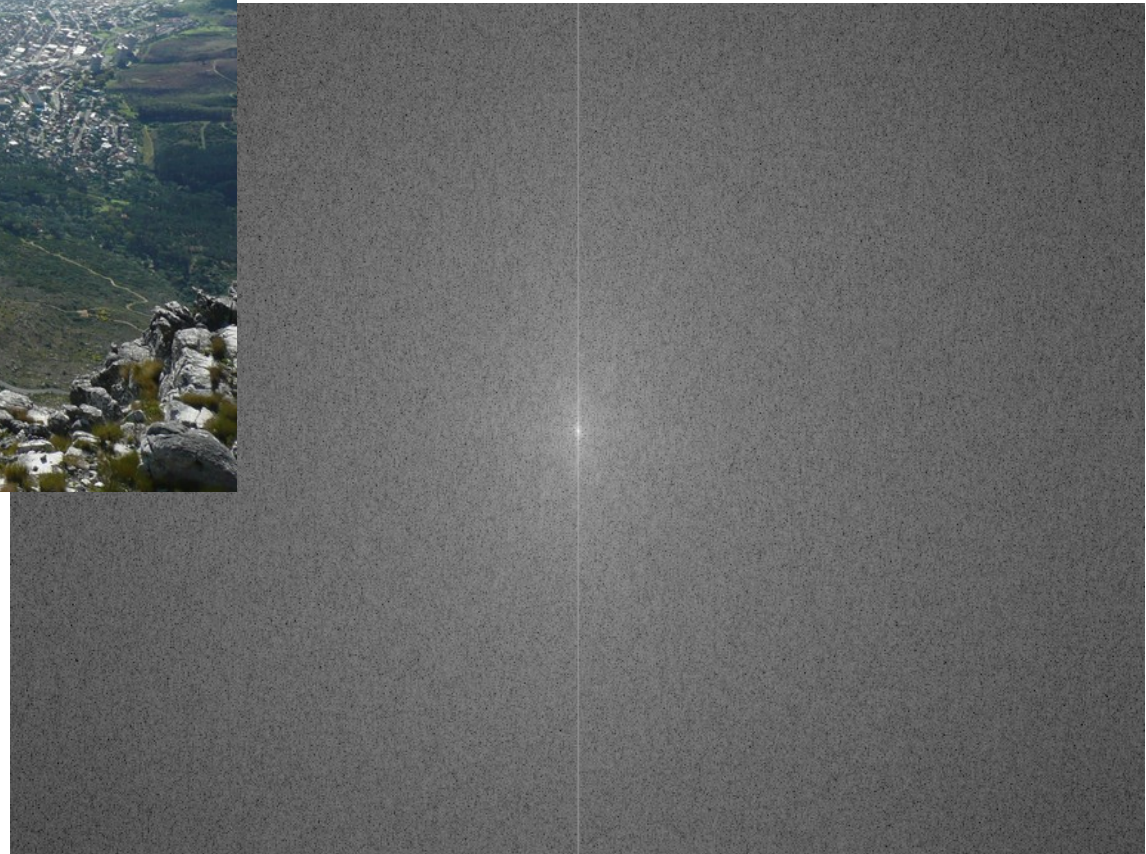
Frequency Domain



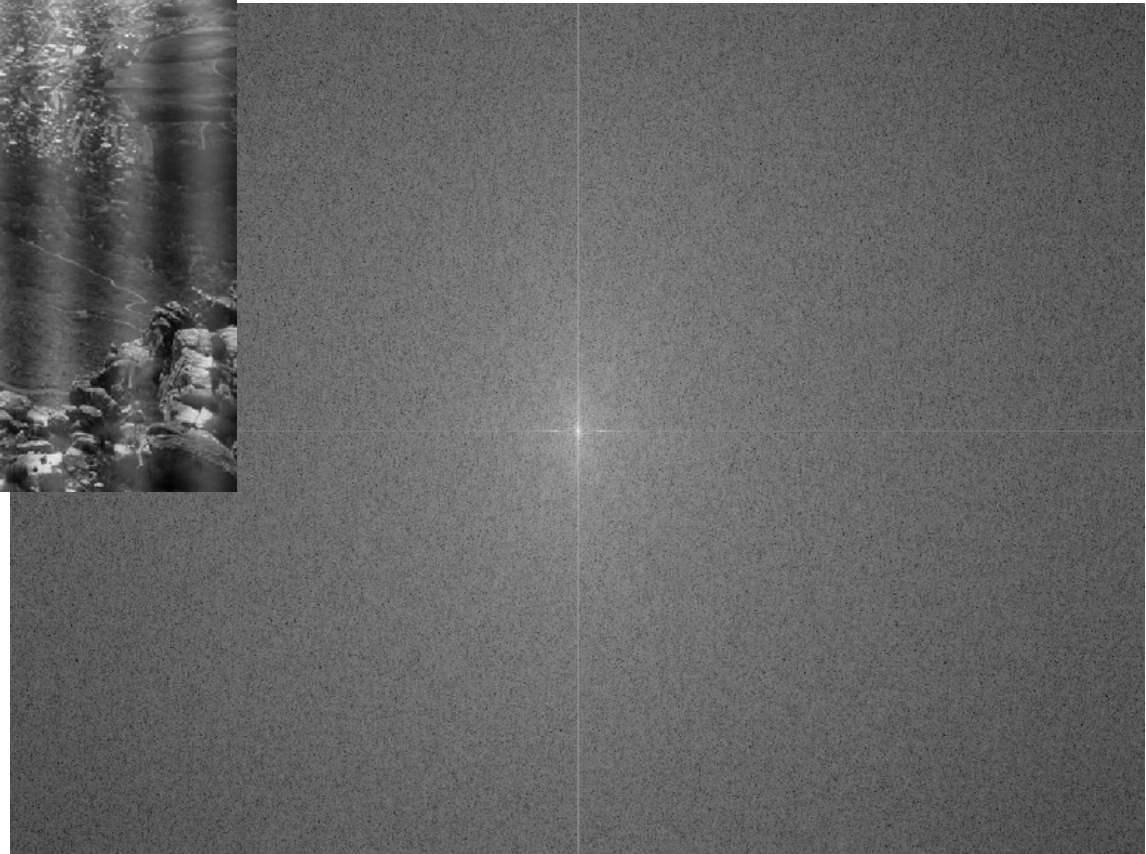
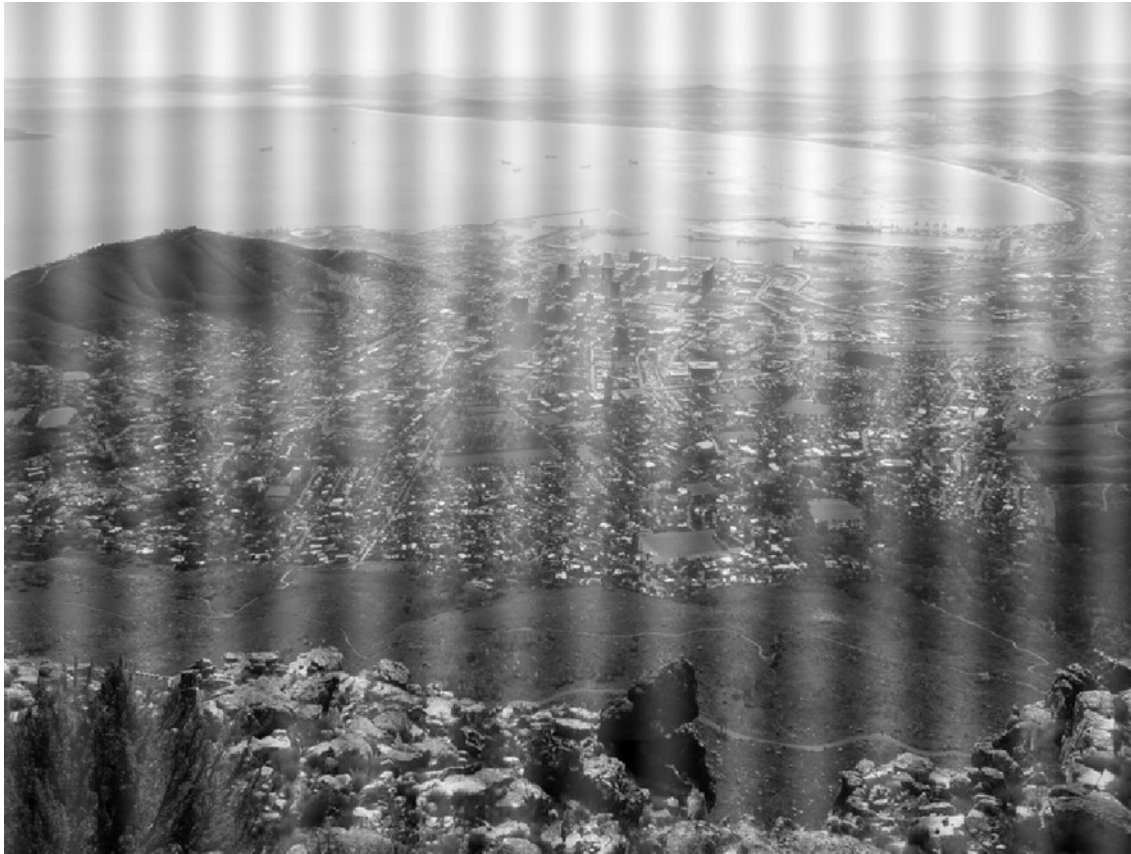
Frequency Domain



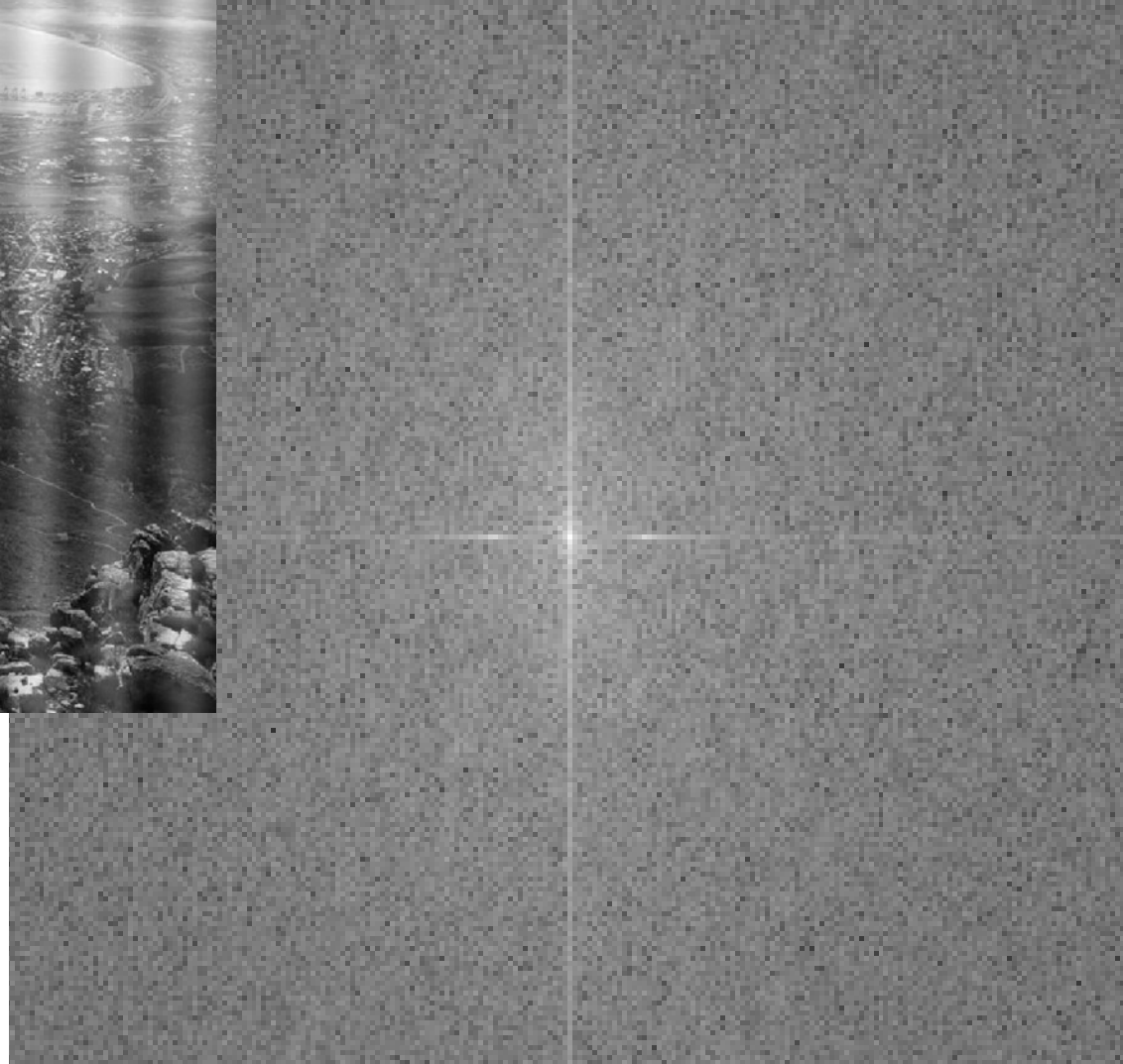
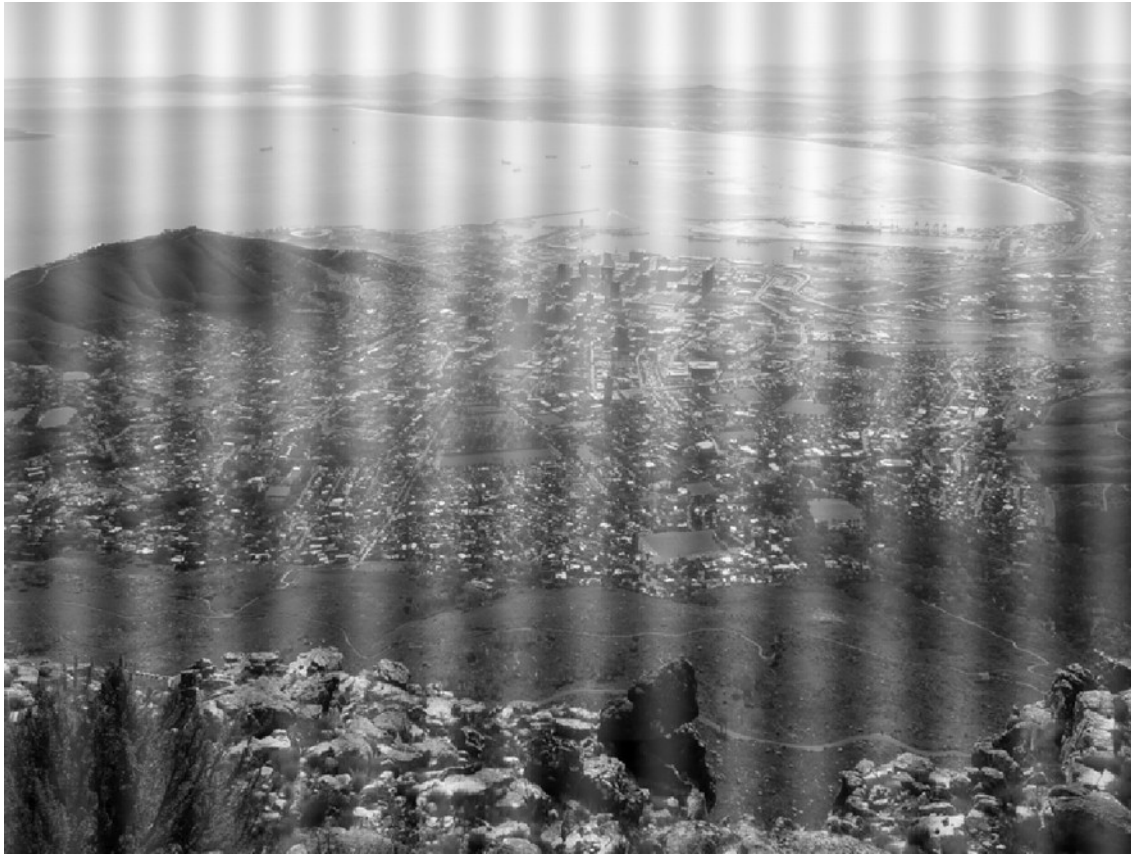
Frequency Domain



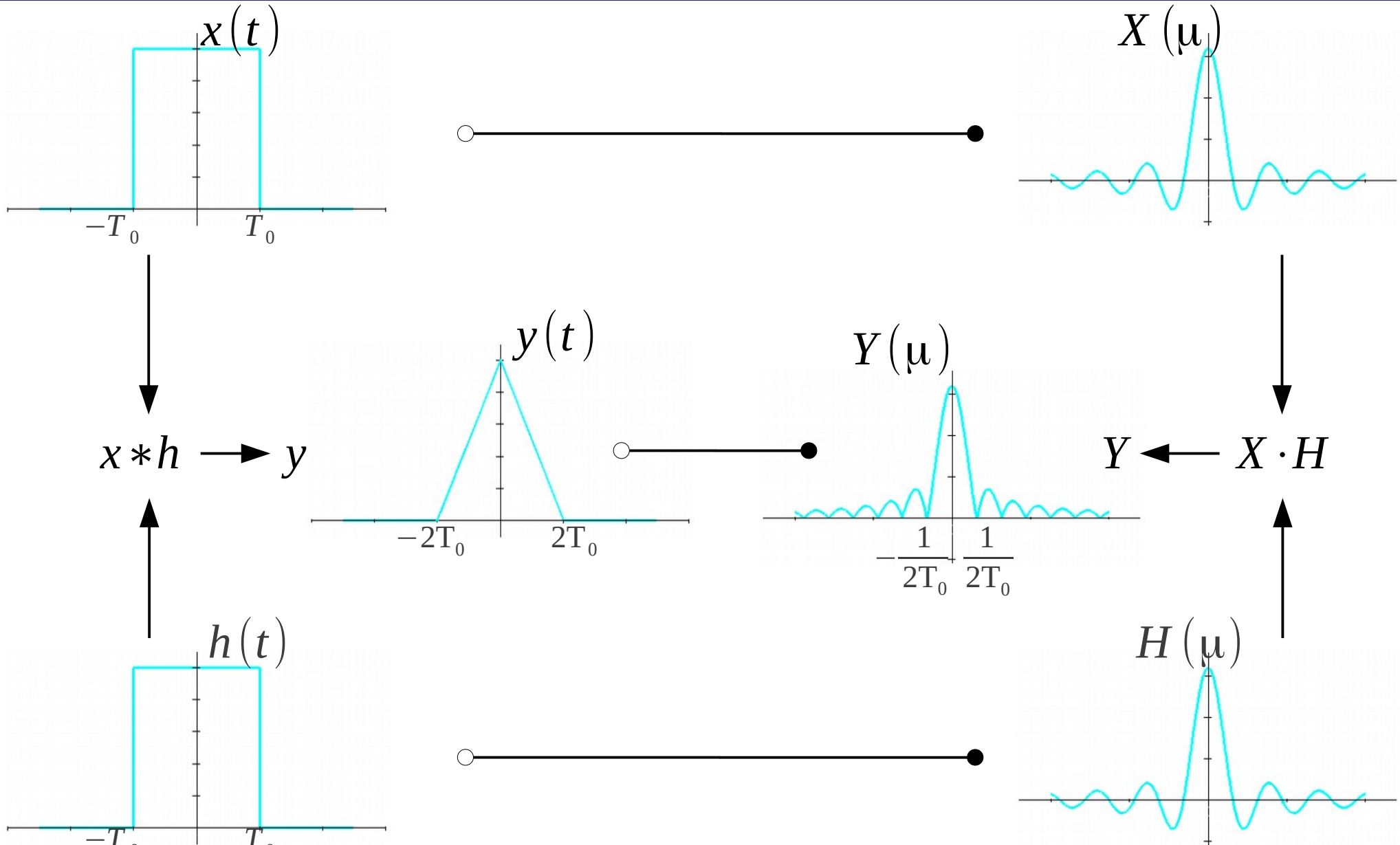
Frequency Domain



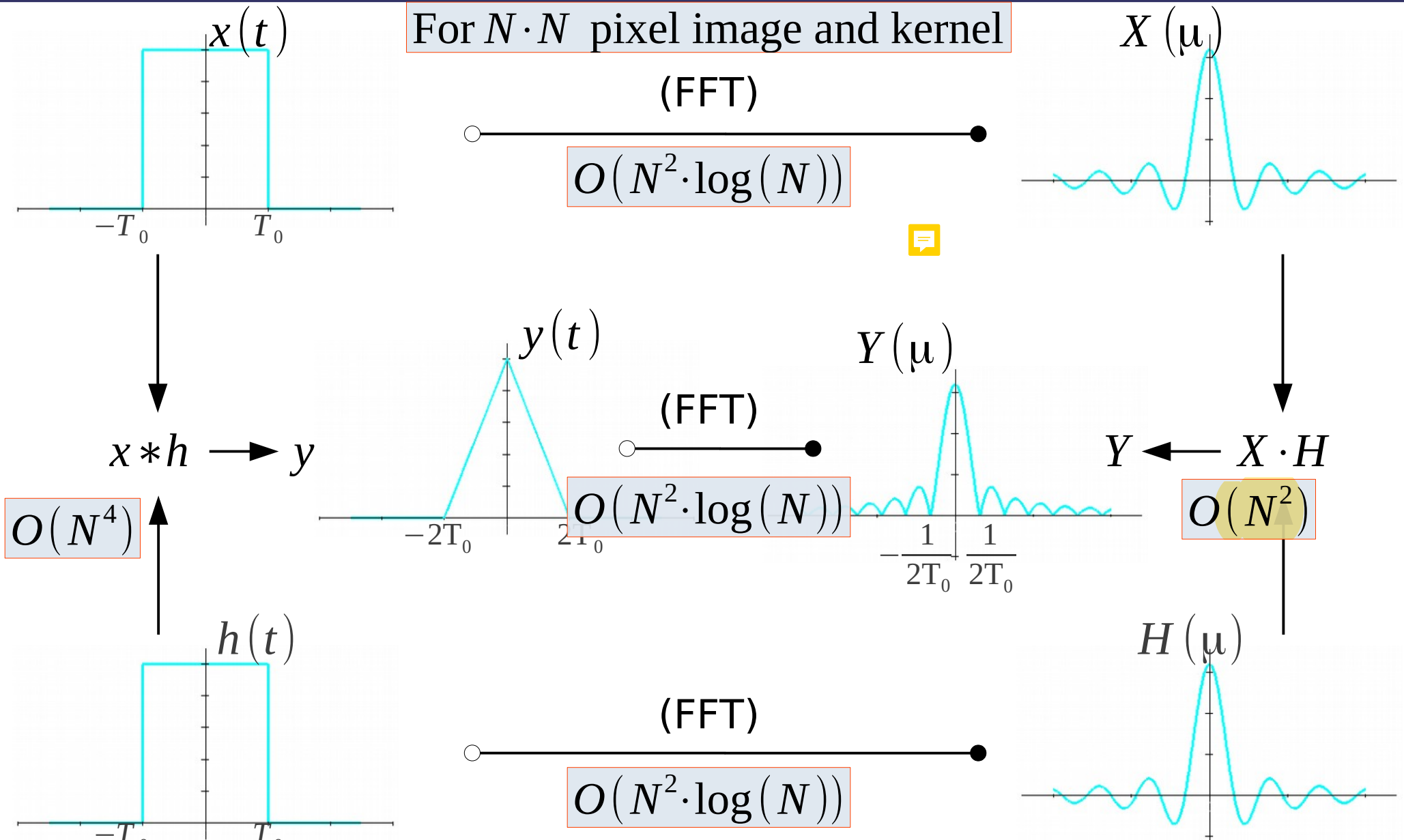
Frequency Domain



Example (Convolution Theorem)



Example (Convolution Theorem)

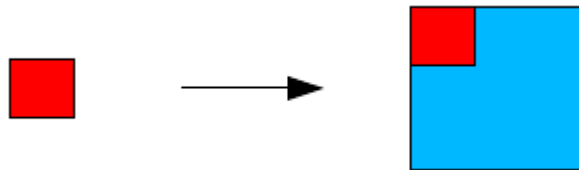


Convolution Theorem

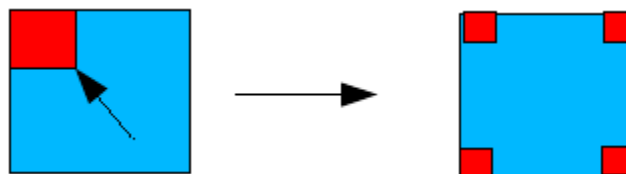
$$\begin{aligned} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} y(t) \exp(-i\mu t) dt \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp(-i\mu t) \left[\int_{-\infty}^{\infty} x(\tau) h(t - \tau) d\tau \right] dt \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(\tau) \left[\int_{-\infty}^{\infty} h(t - \tau) \exp(-i\mu t) dt \right] d\tau \\ (s = t - \tau) \rightarrow &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(\tau) \left[\int_{-\infty}^{\infty} h(s) \exp(-i\mu(s + \tau)) ds \right] d\tau \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(\tau) H(\mu) \exp(-i\mu\tau) d\tau \\ &= H(\mu) \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(\tau) \exp(-i\mu\tau) d\tau = \boxed{H(\mu) X(\mu)} \end{aligned}$$

Convolution Theorem

1. Kernel and image must have the same size before transformation
 - Copy the kernel into a larger matrix



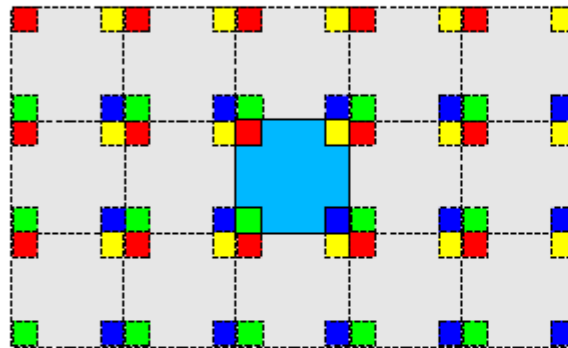
2. Centre the kernel (shift the central element to $F'(1,1)$)



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:



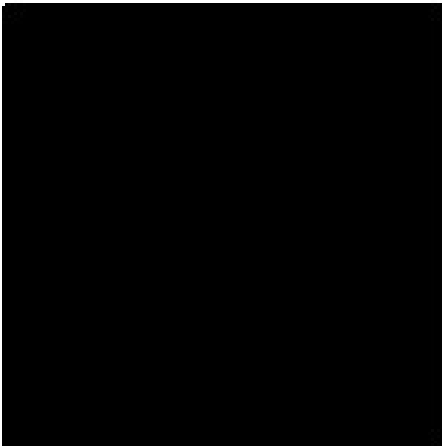
Image/Filter



Periodic repetition

- 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!

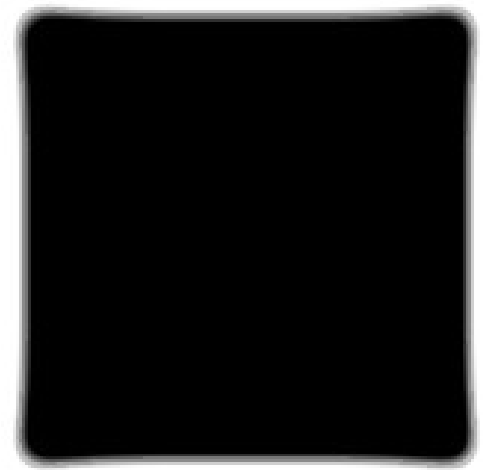
Unsharp Masking



Original image



Distorted image



Enhanced image

Unsharp Masking



Unsharp Masking

Black: Original edge y

Red: Degraded edge y_0

USM (magenta):

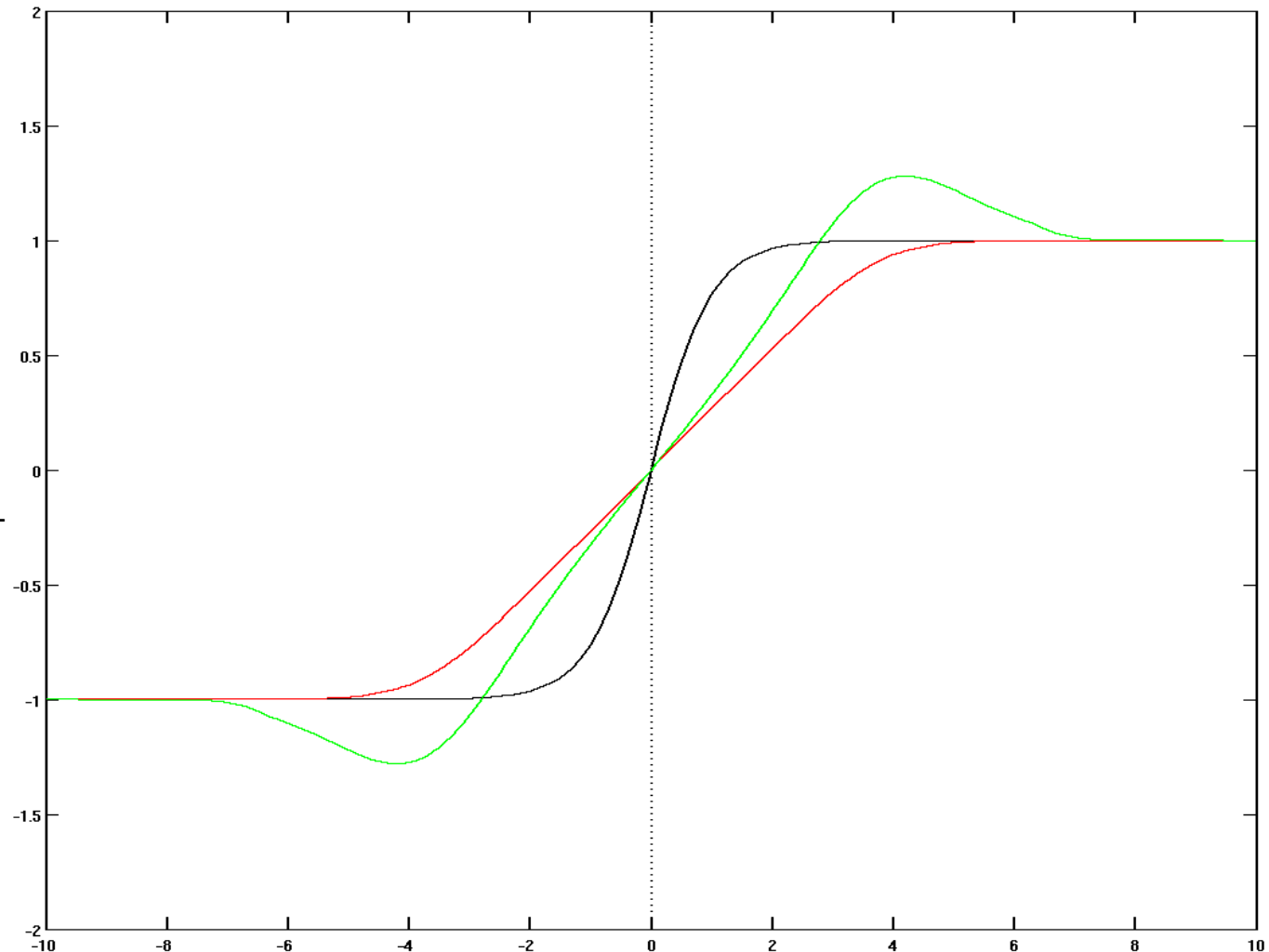
1. Smooth: $y_0 \rightarrow y_1$

2. Subtract: $y_2 = y_0 - y_1$

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

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

→ 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
1. Smooth image I_0 to obtain I_1
(1st 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.
(2nd parameter: threshold T)
4. Scale difference to further enhance edges
 $I_3 = s * I_2$
(3rd 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:

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

Parameter:

k, T, s

Unsharp Masking

Slightly improved (in my opinion):

Difference image:

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

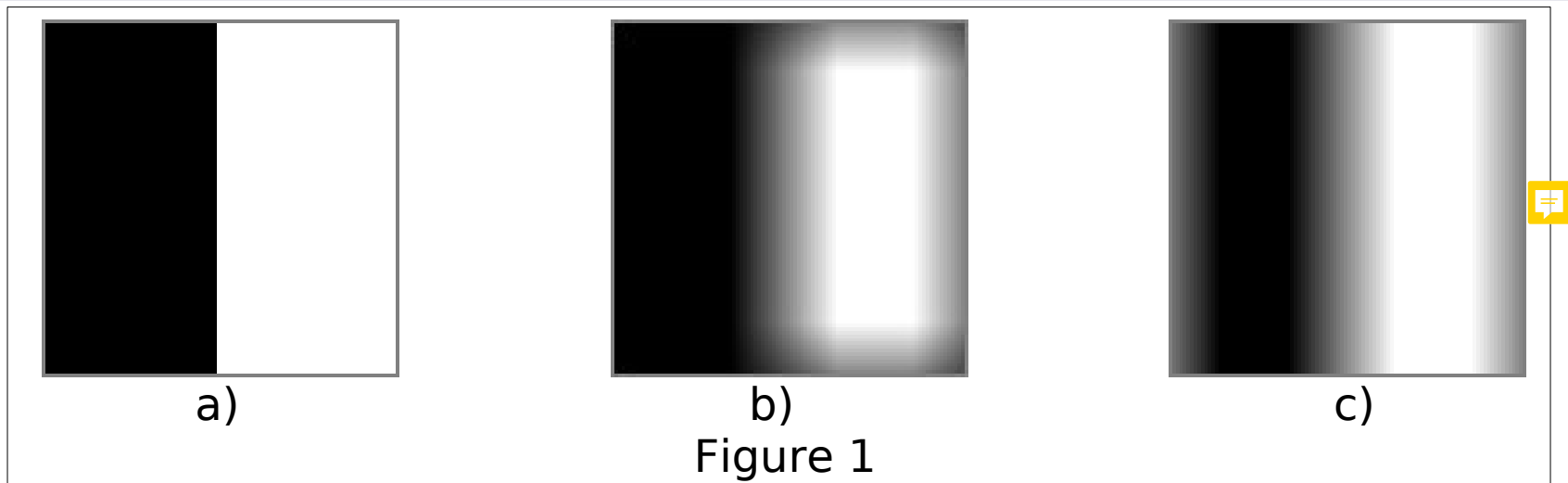
Enhanced image:

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

Parameter:

k, T, s

3. Exercise - Theory



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.

- Explain which assumptions lead to the “unexpected” border values in each image and why they are different for both methods.
- What steps are necessary for the convolution in spatial domain to produce the result in Fig. 1c)?
- 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)

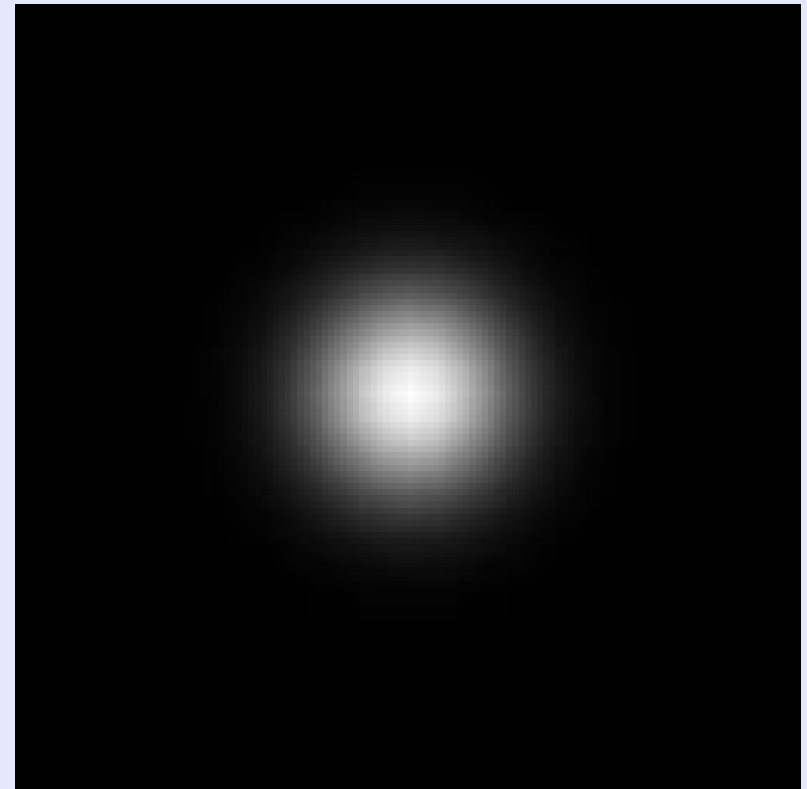
kSize: the kernel size

return: the generated filter kernel

- Calculates Gaussian filter kernels

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

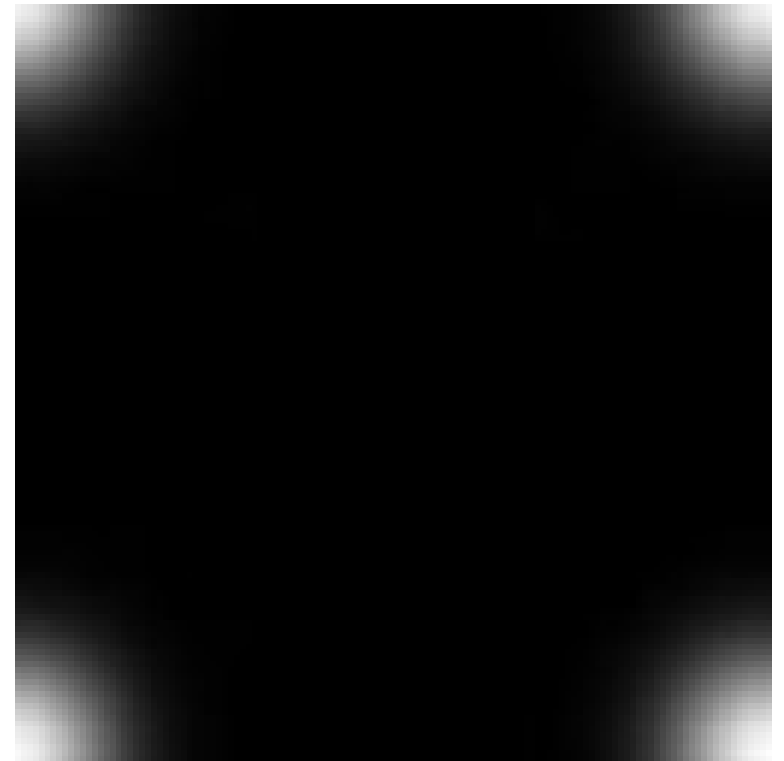
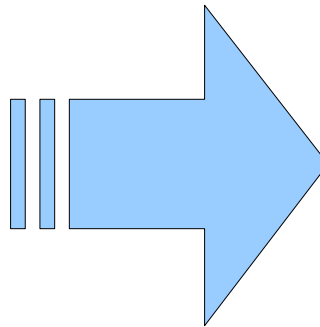
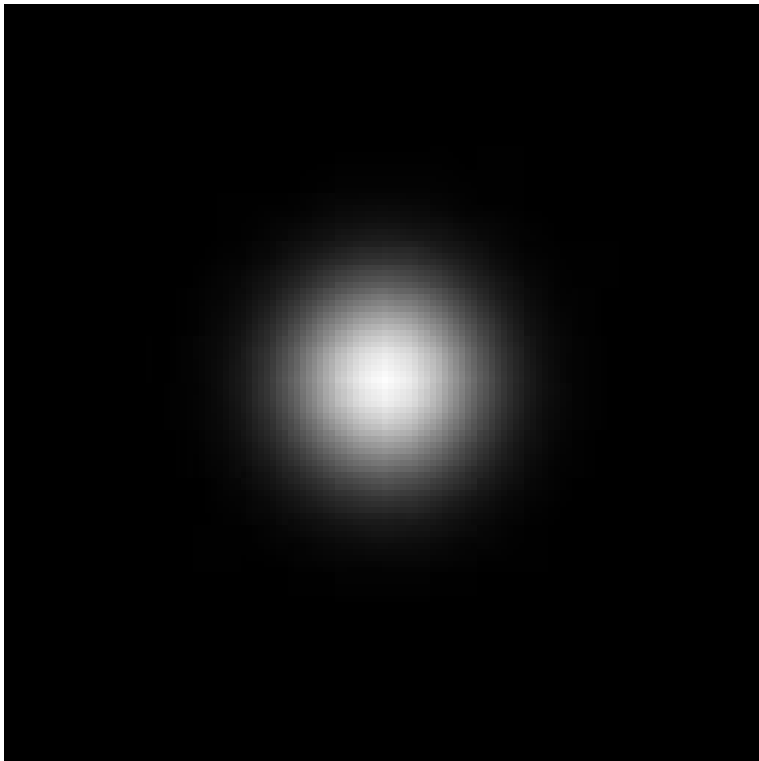
- Mean and variance are determined by kernel size (e.g. $\sigma = kSize/5$)
- Maximal value at centre
- Must integrate to one!



3. Exercise - To Do

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

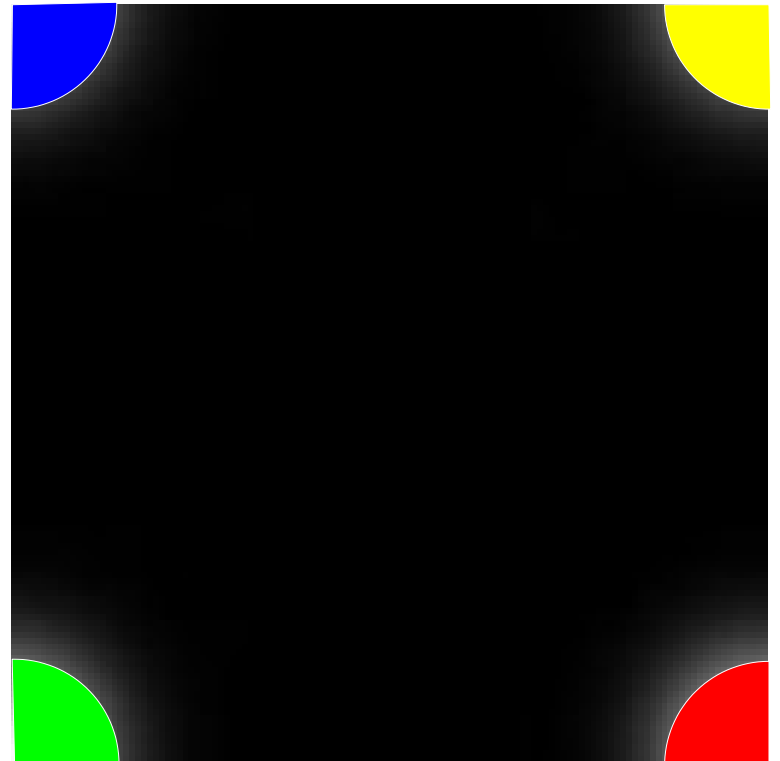
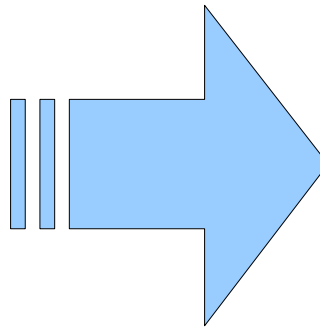
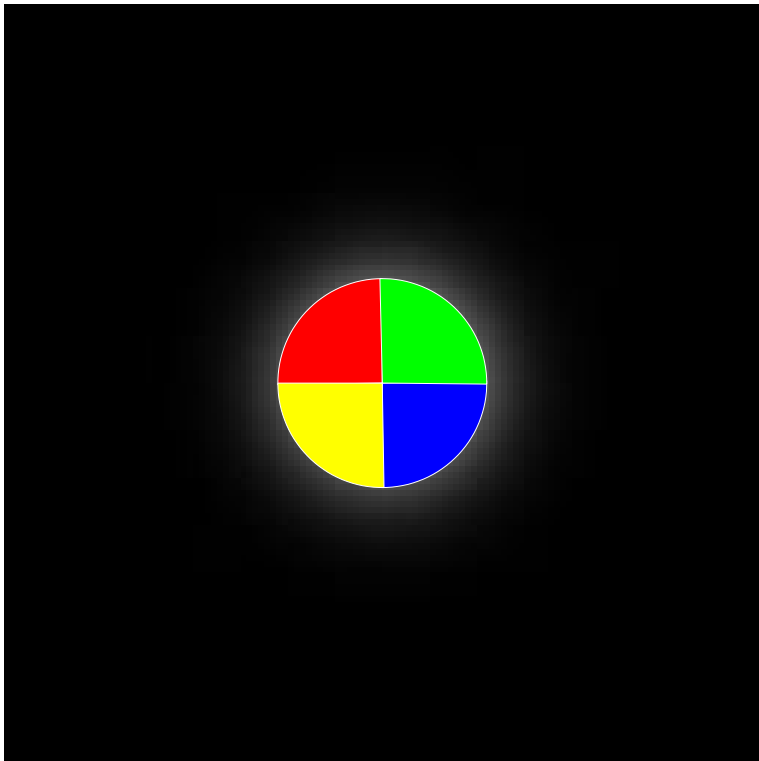
- Performes circular shift in (dx,dy) direction



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

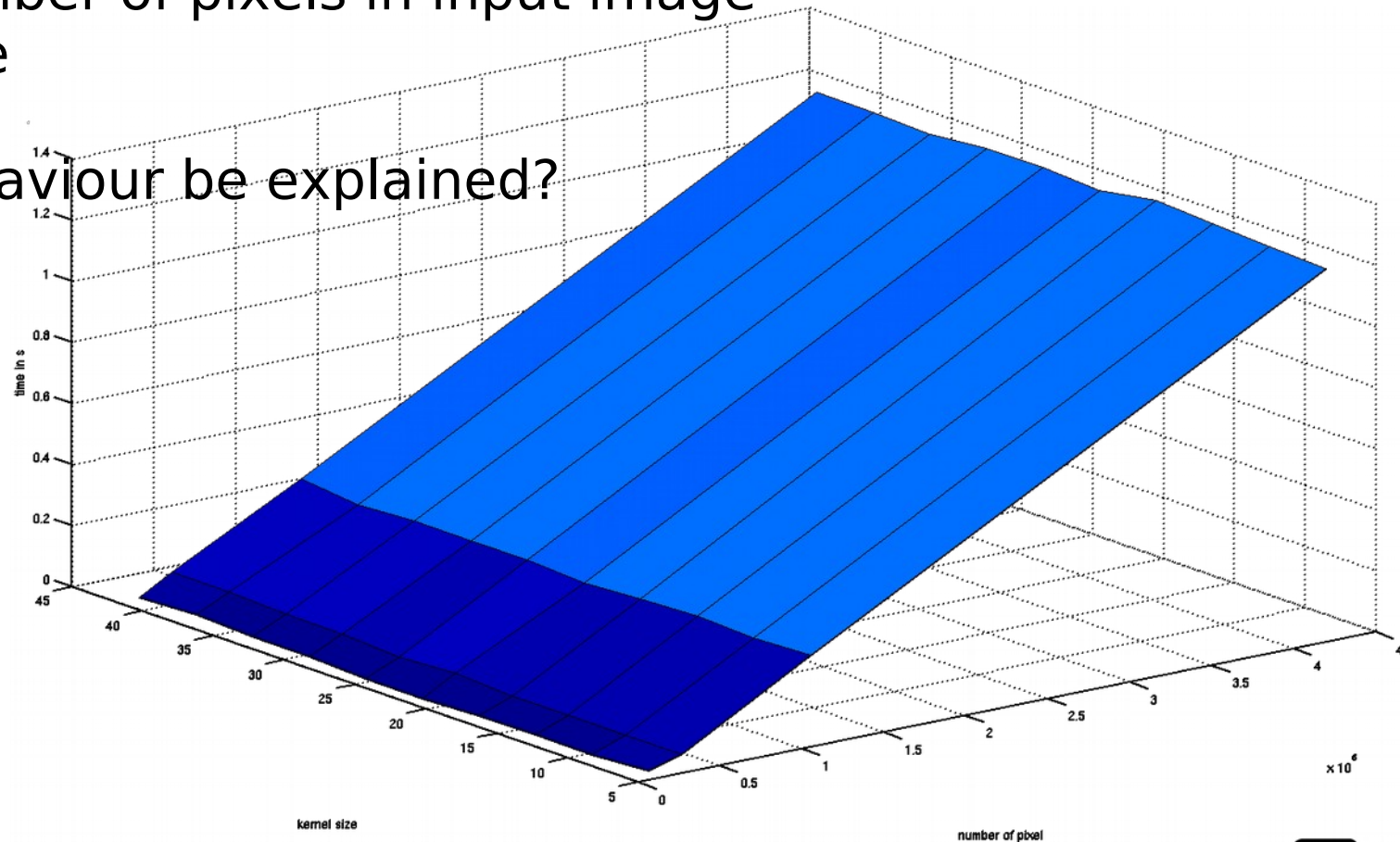
Mat Dip3::usm(Mat& in, FilterMode filterMode, int size, double thresh, double scale)

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: 26.11.2019

Next Meeting: 26.11.2019

No consultation times on 13th and 20th of November!
→ I'll try to respond to emails though