Digital Image Processing

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

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

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

Signal

Reconstructed from the spectrum

Spectrum

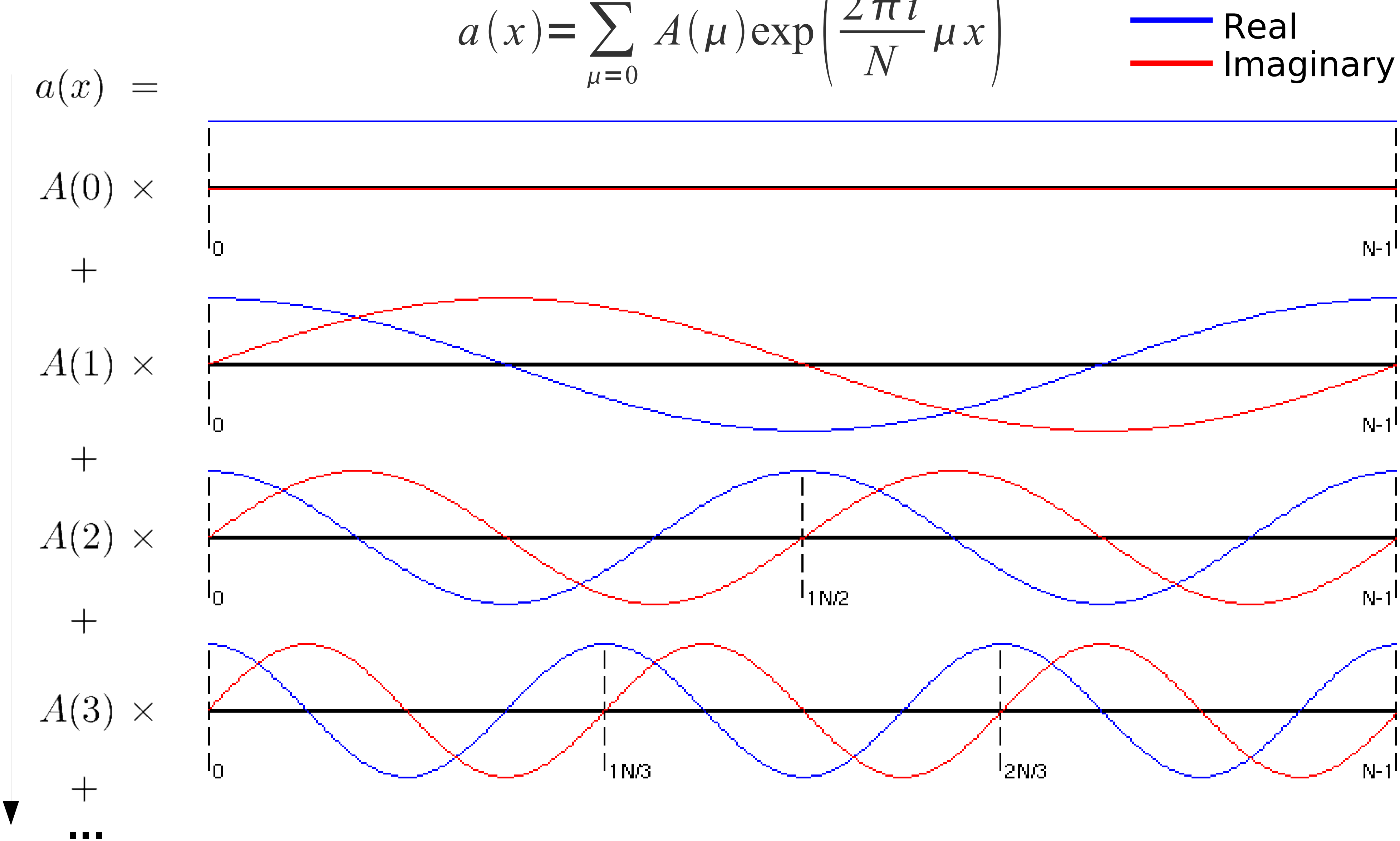
Frequency domain representation: Weight and phase of frequency

Basis Function

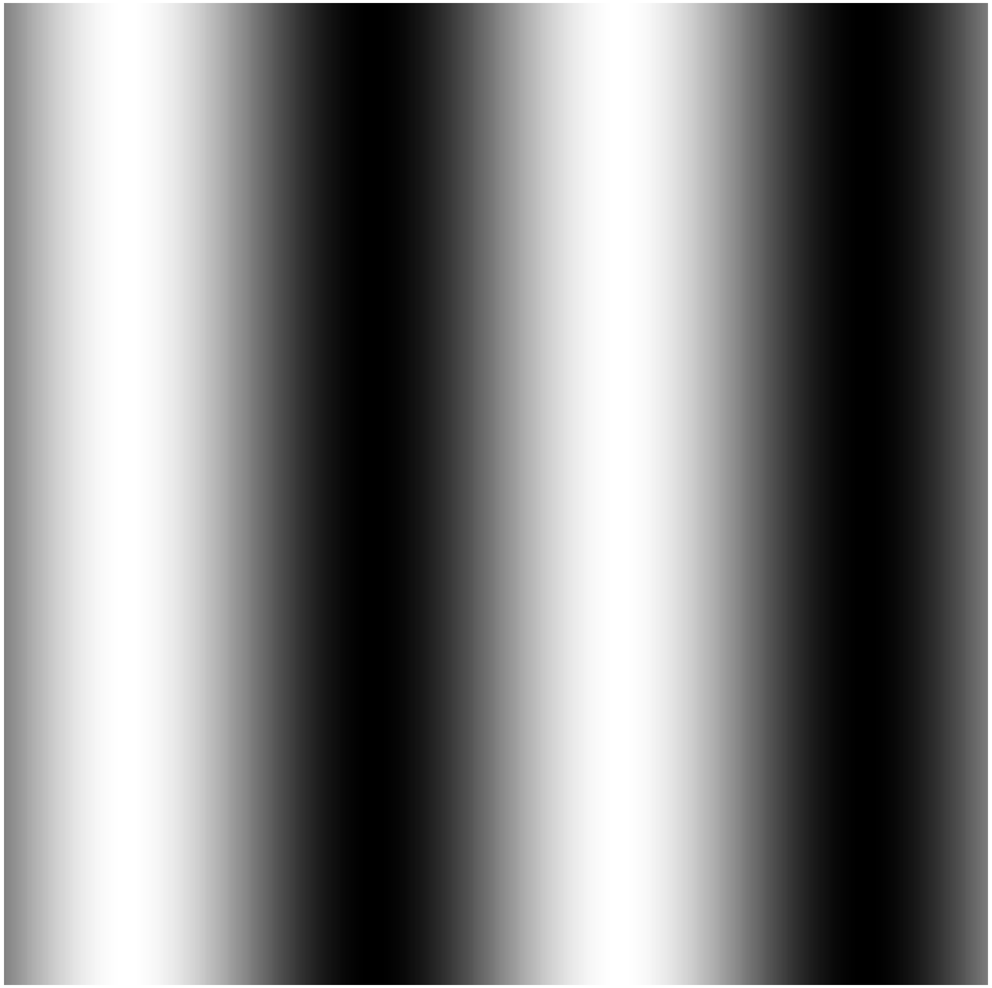
Complex exponential (oscillation)

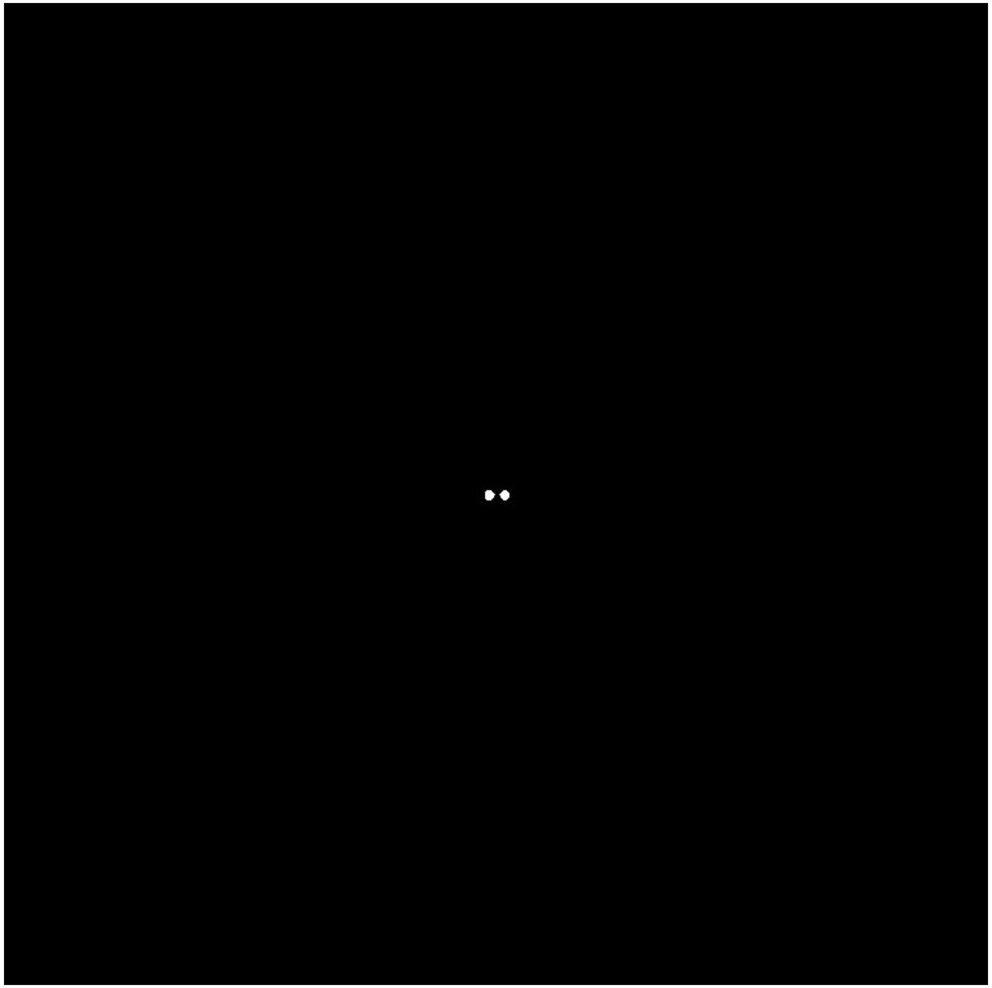
Frequency Domain

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

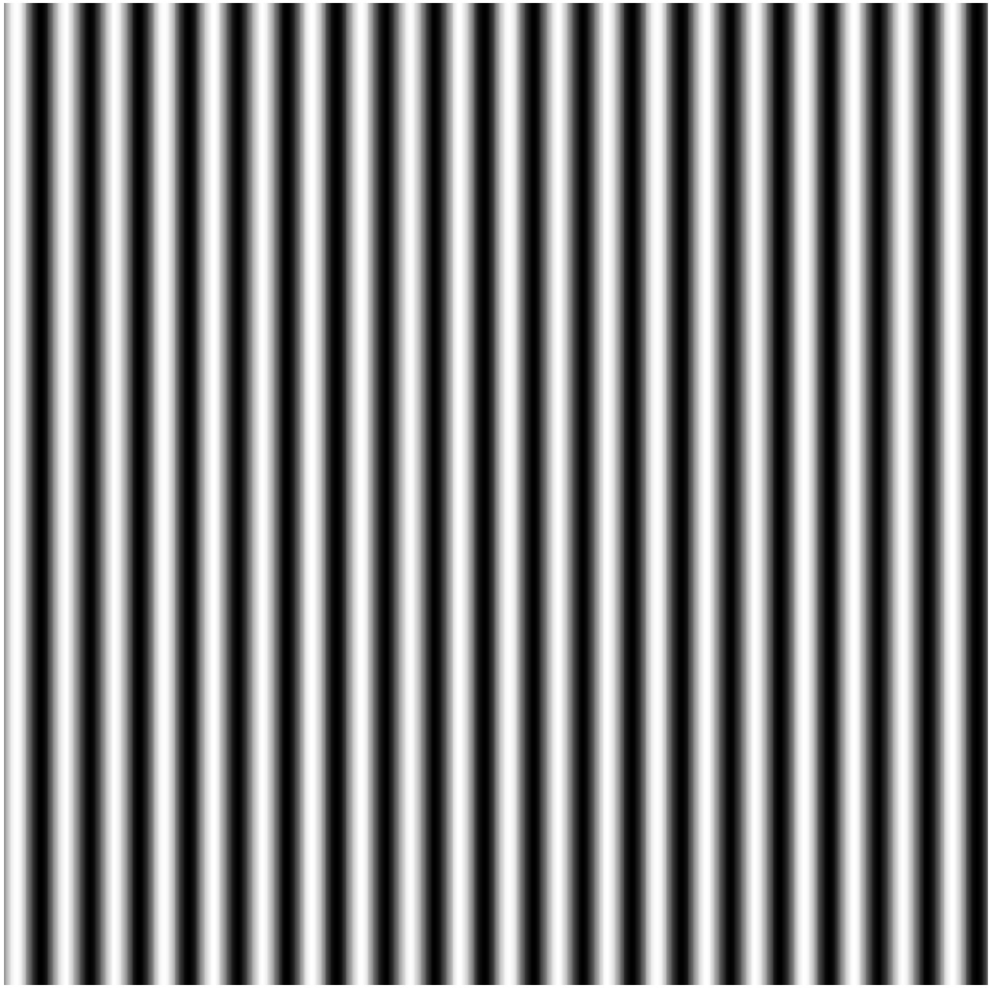


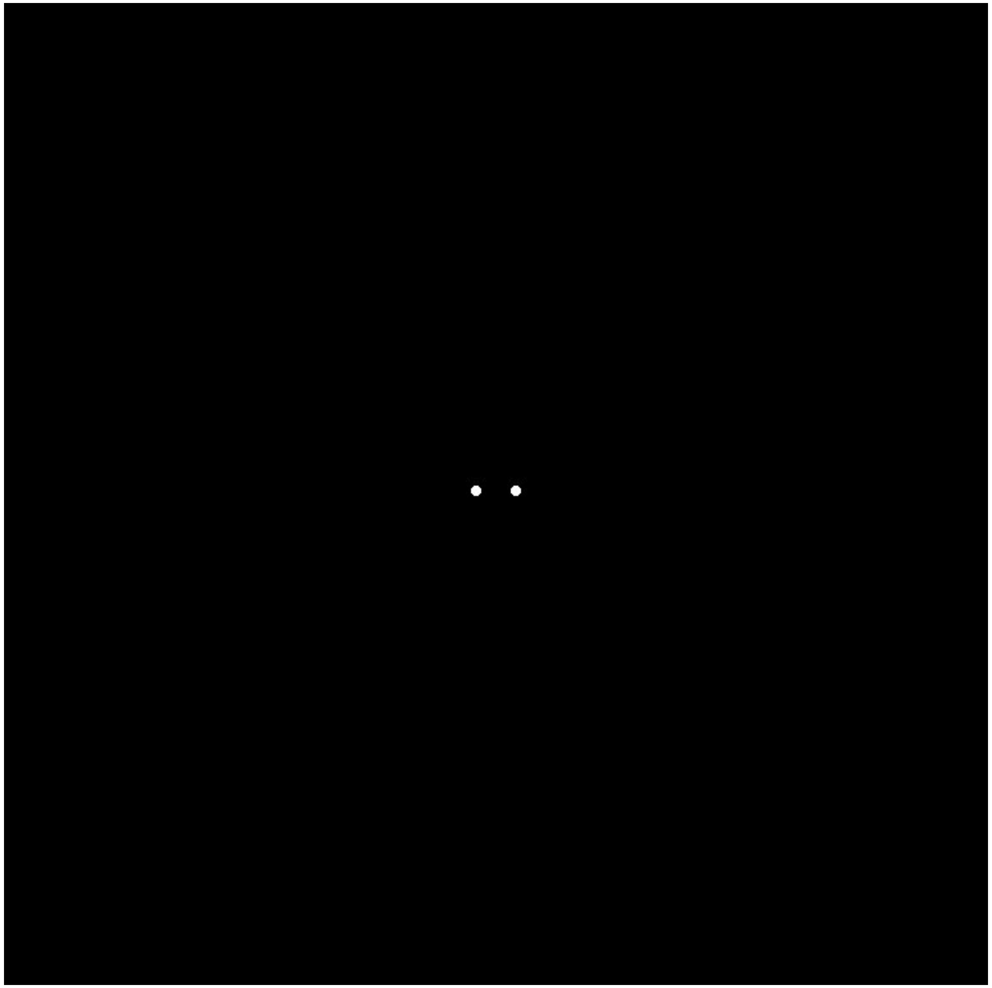
Frequency Domain



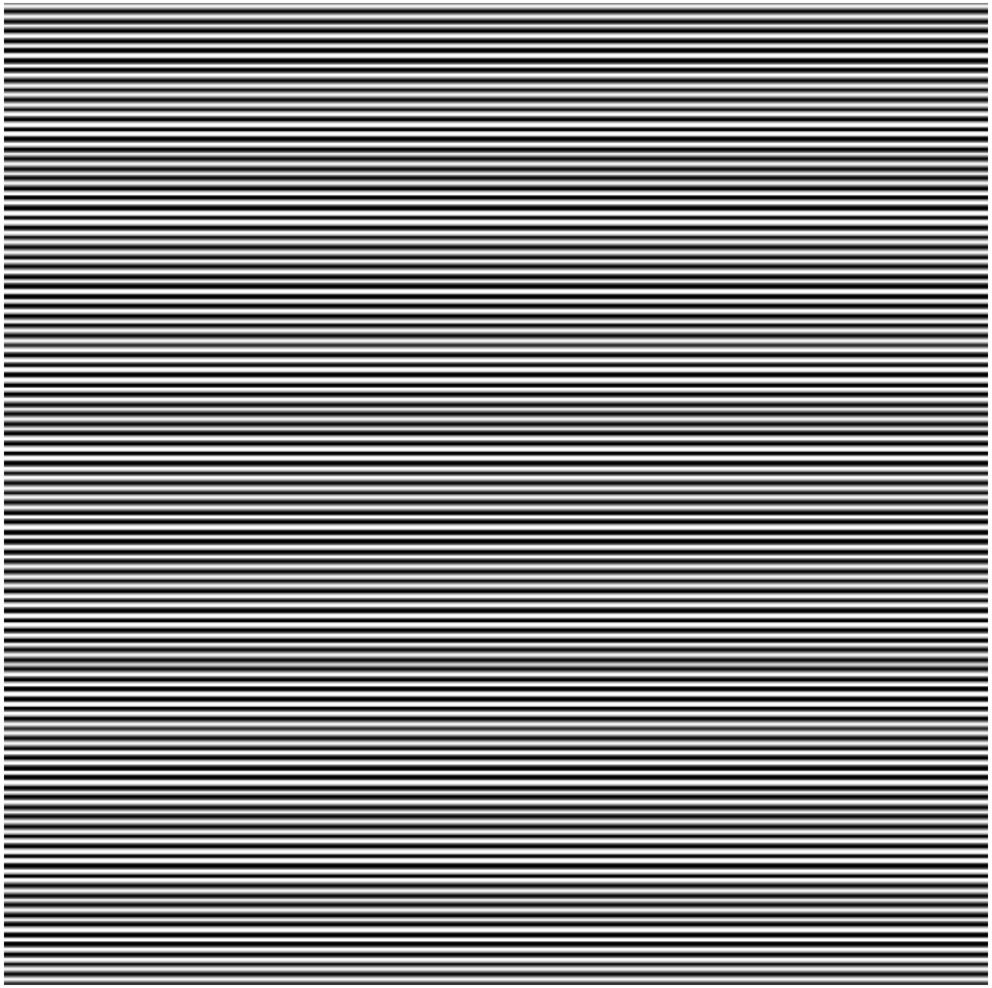


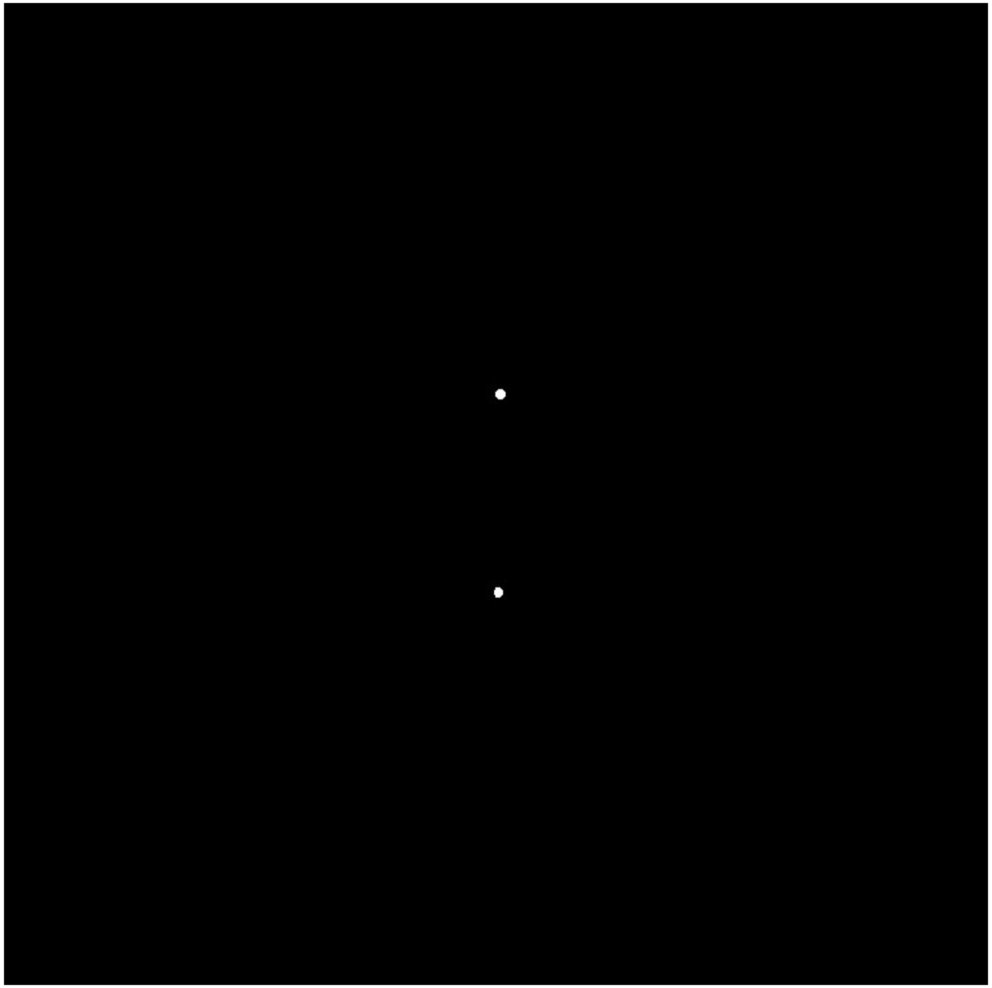
Frequency Domain



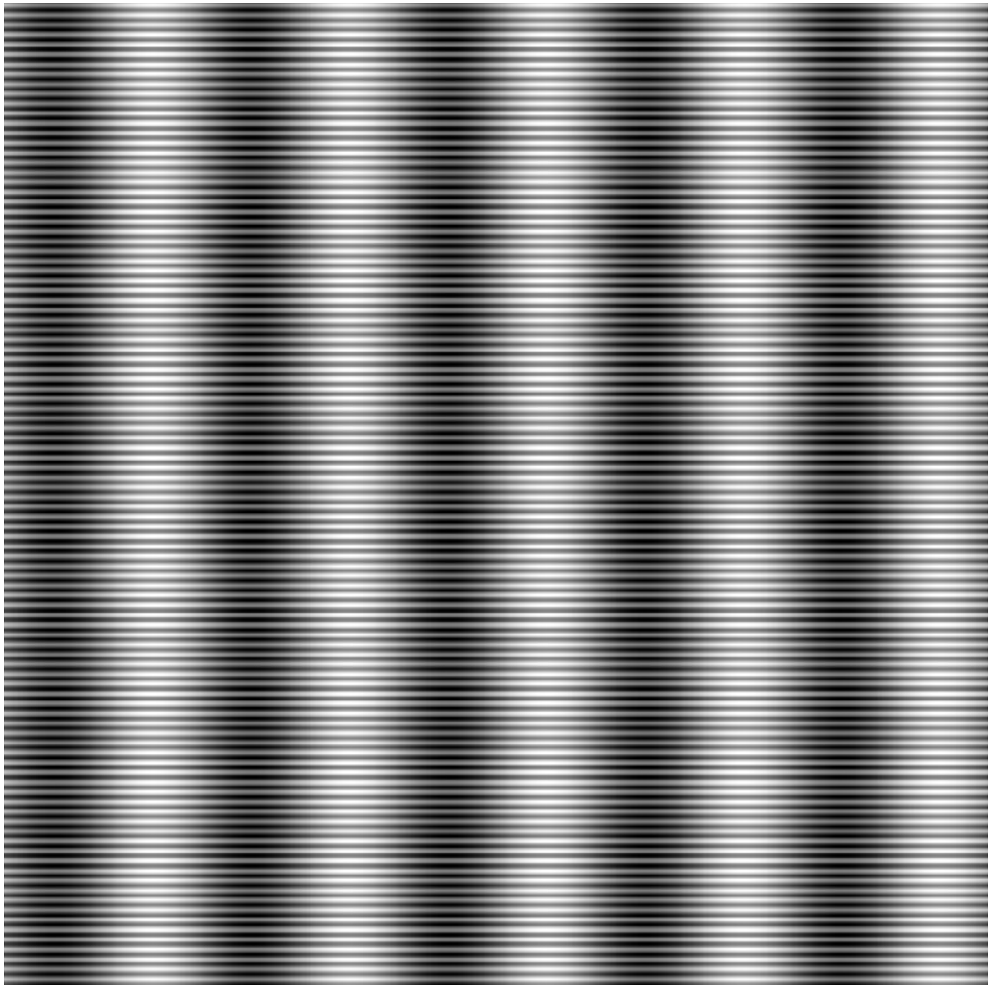


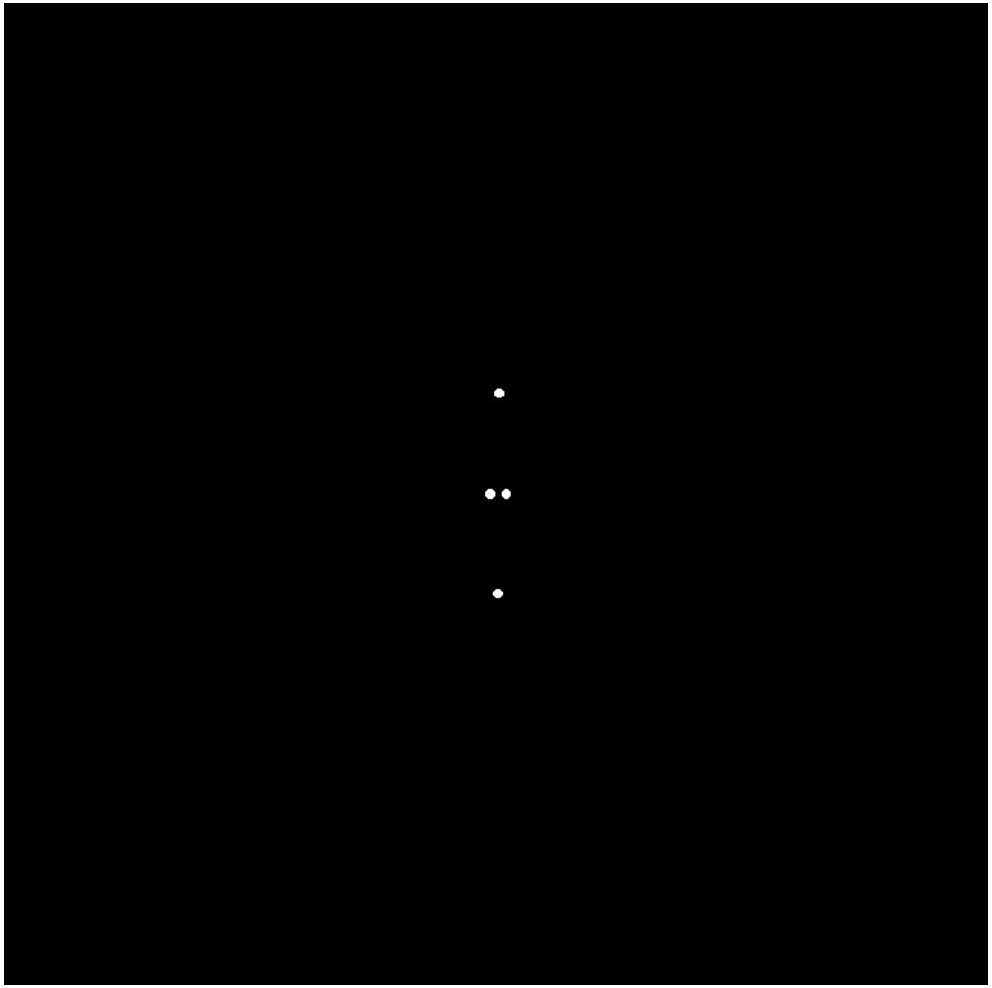
Frequency Domain



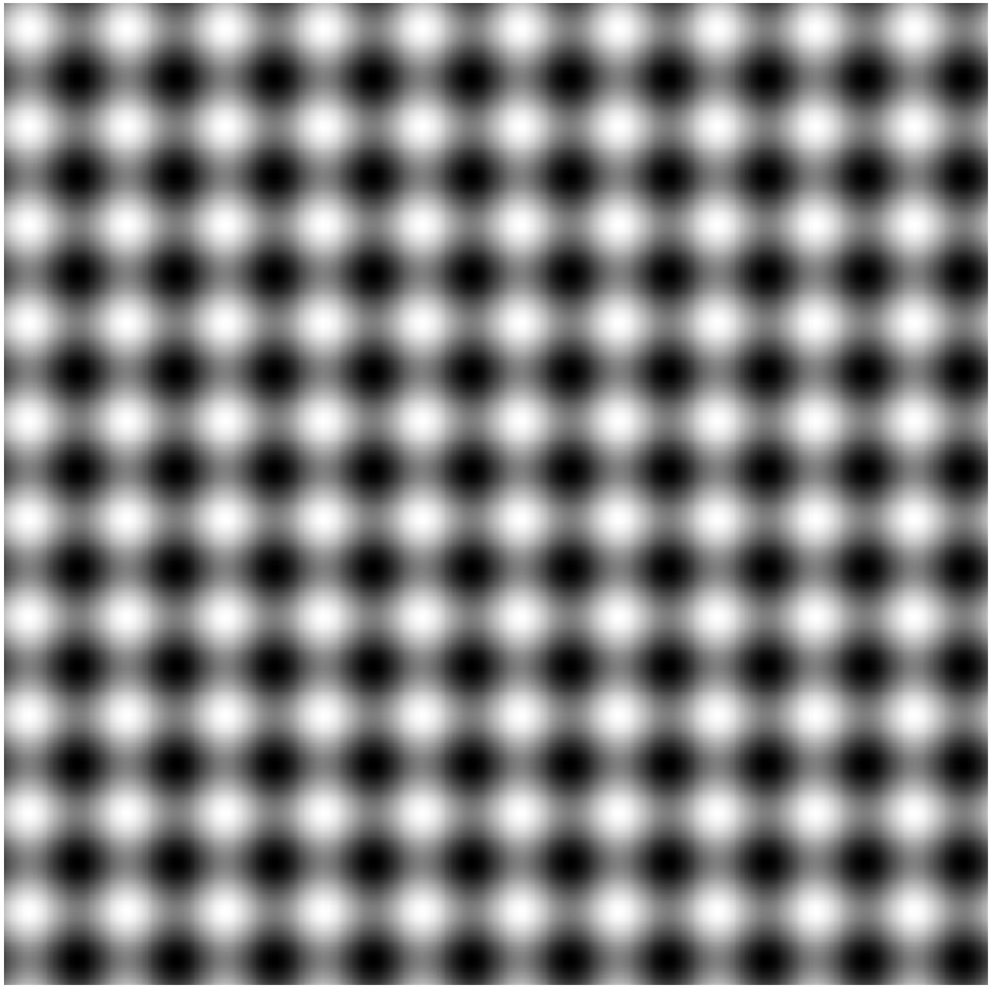


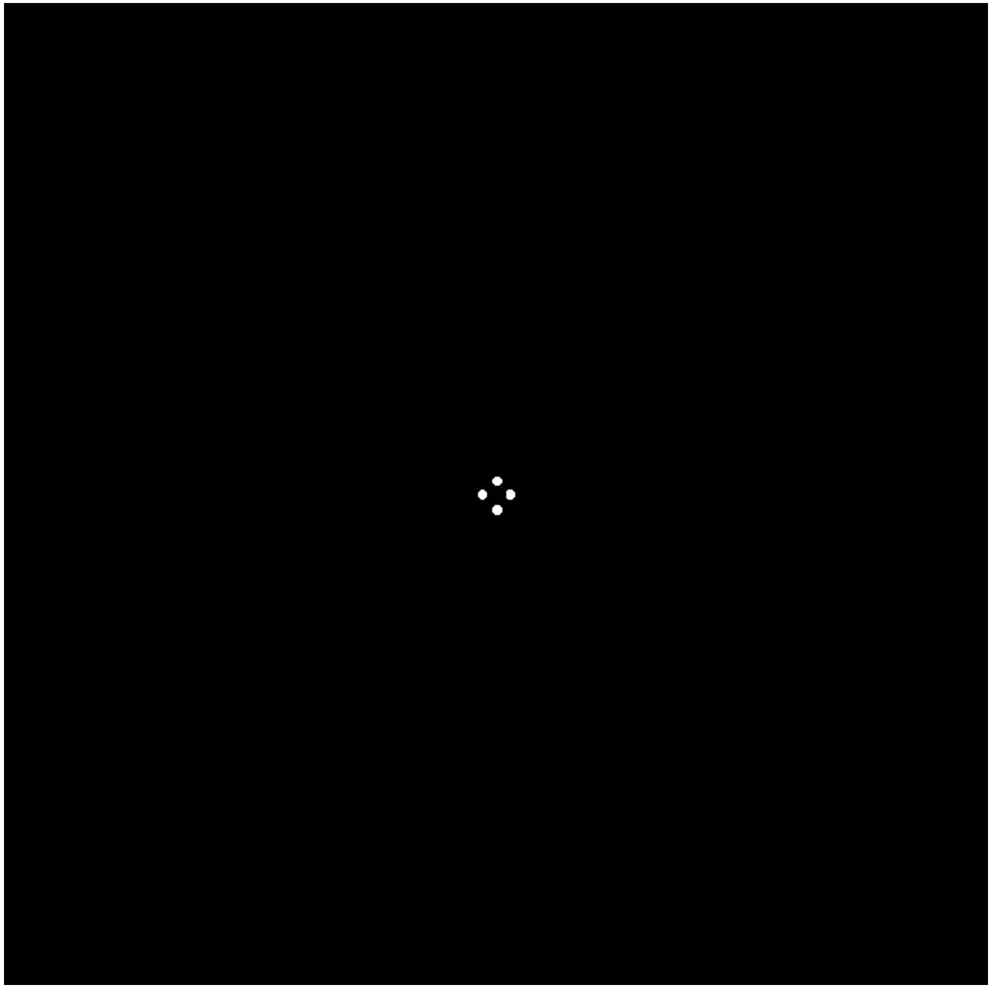
Frequency Domain



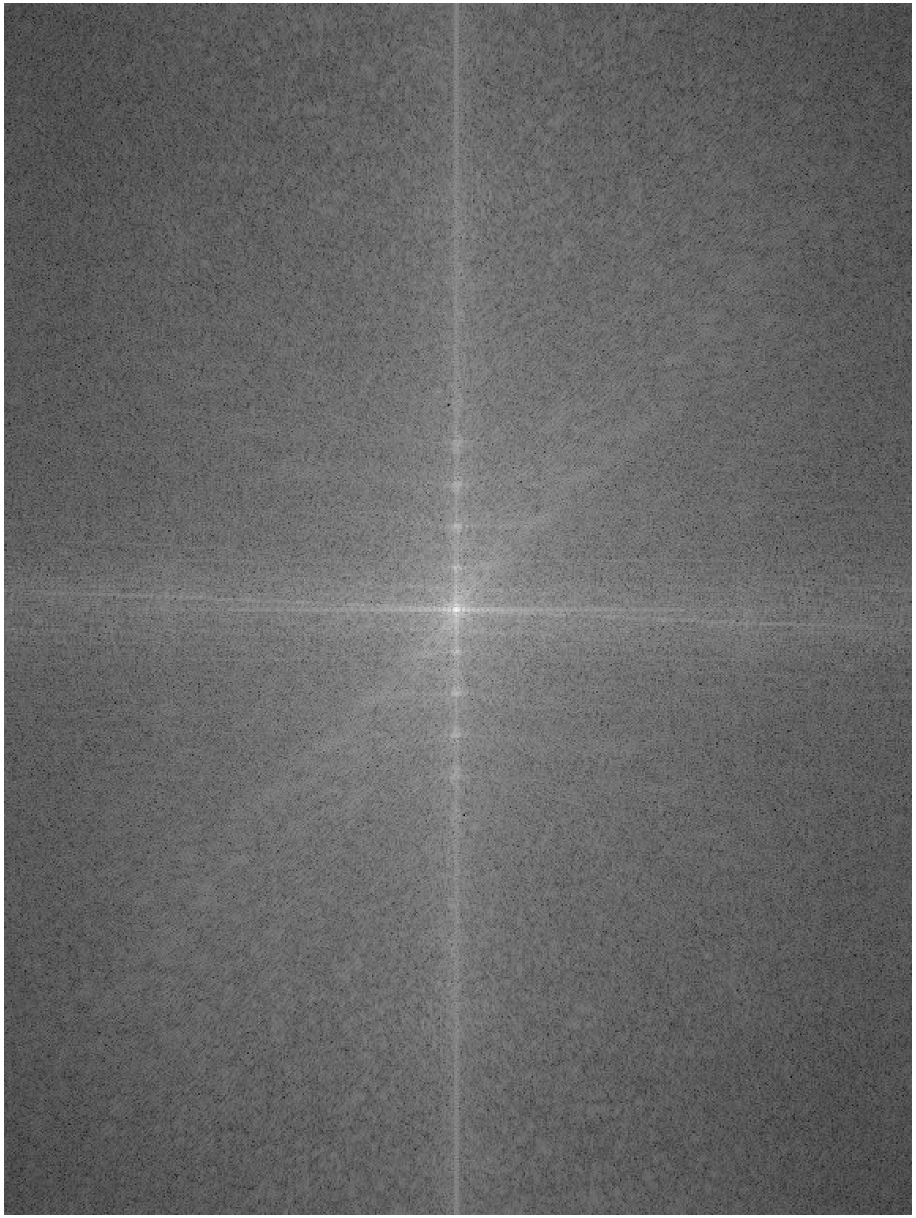


Frequency Domain

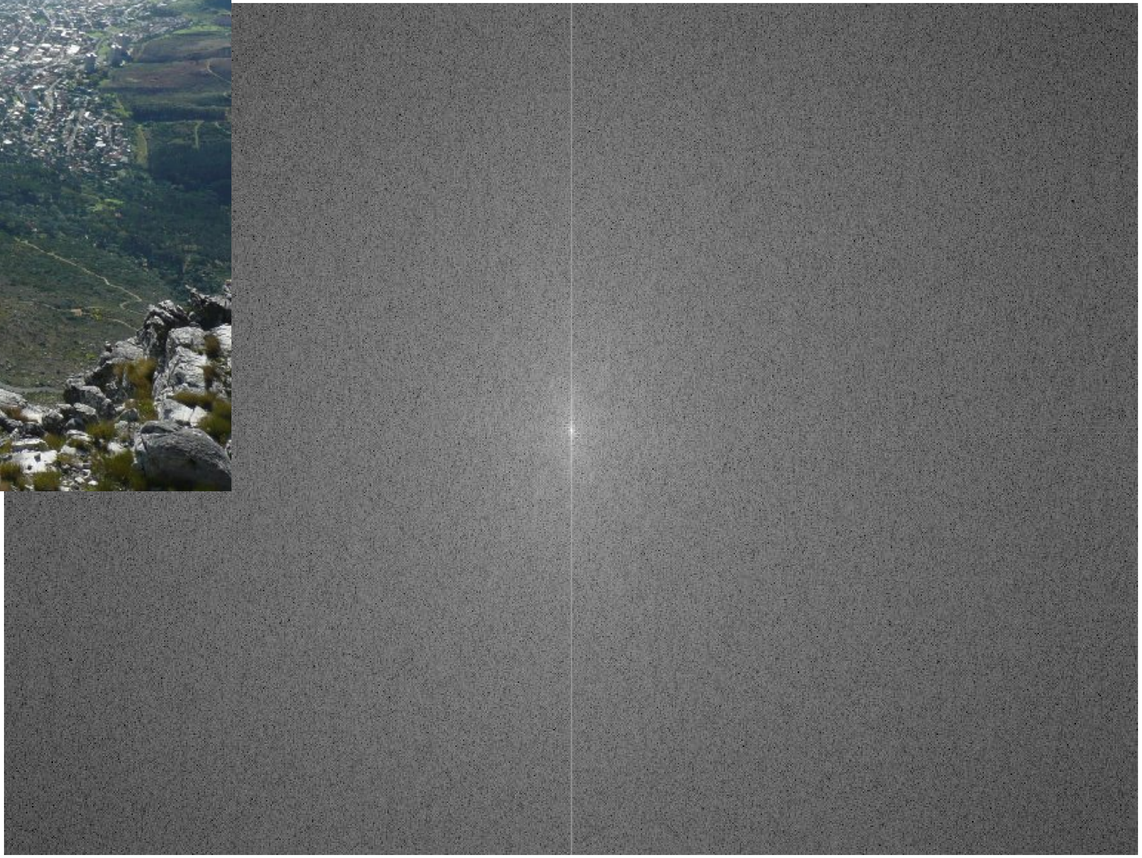




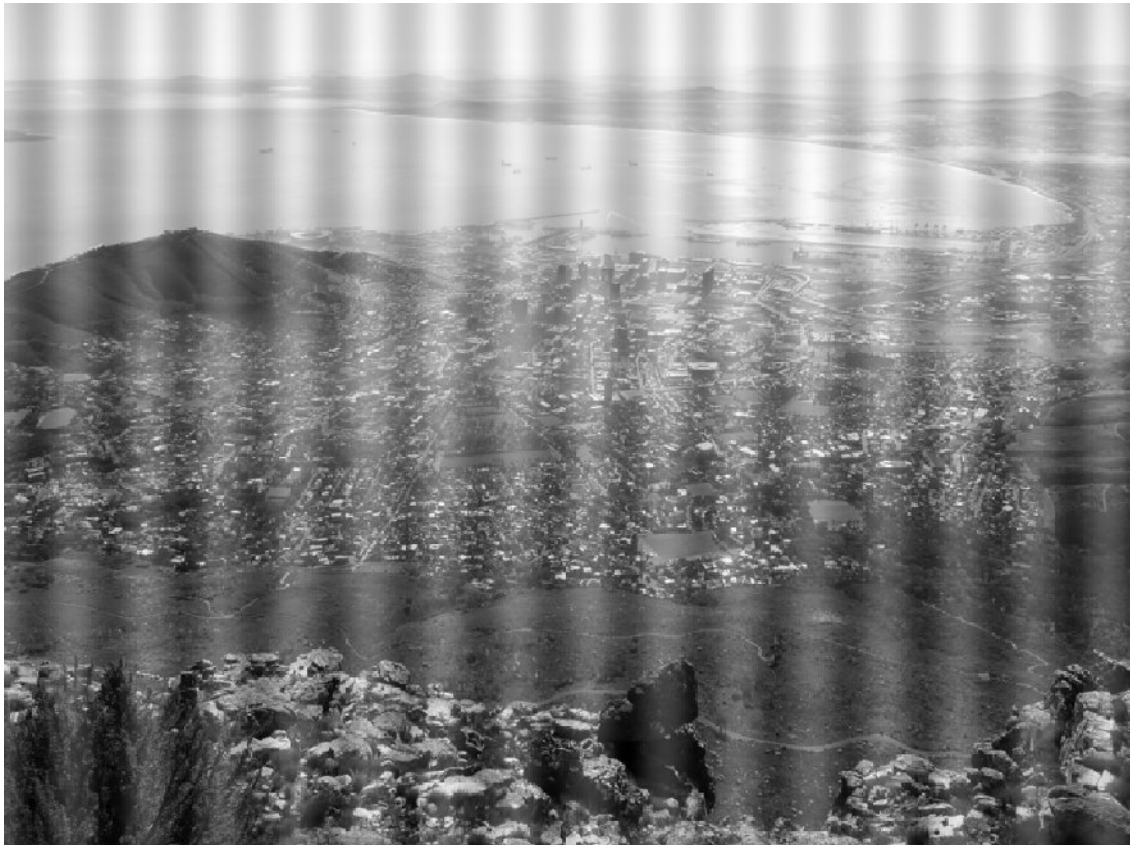
Frequency Domain

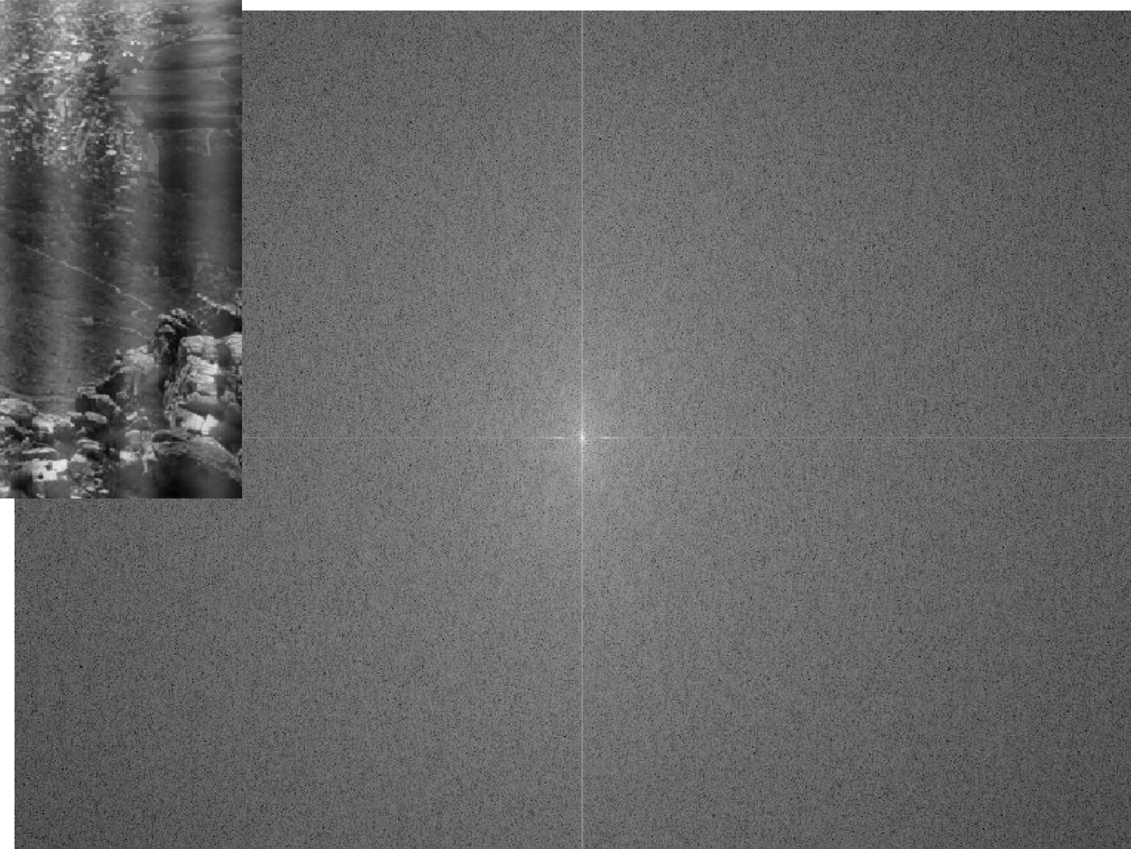


Frequency Domain



Frequency Domain





Convolution Theorem

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

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

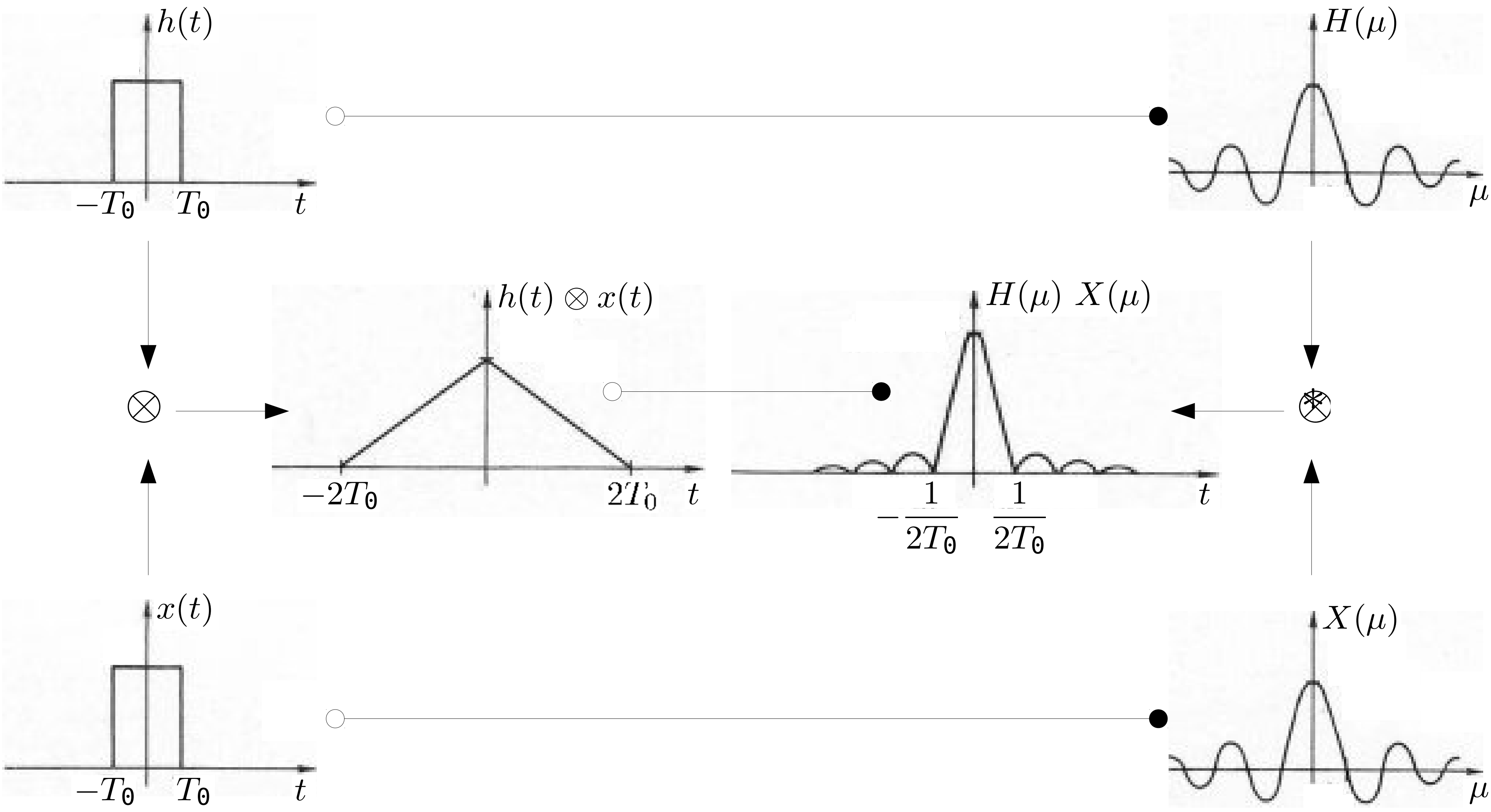
$$= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} x(\tau)\left[\int_{-\infty}^{\infty} h(t-\tau)\exp(-\imath\mu t)\mathrm{d}t\right]\mathrm{d}\tau$$

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

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

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

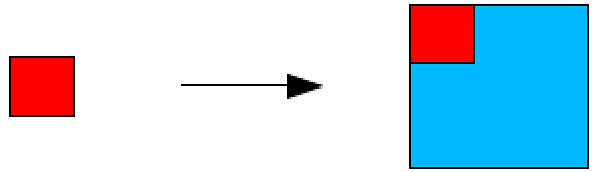
Convolution Theorem



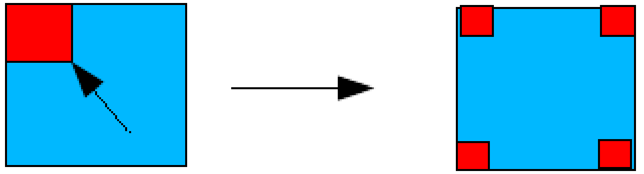
Convolution Theorem

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))

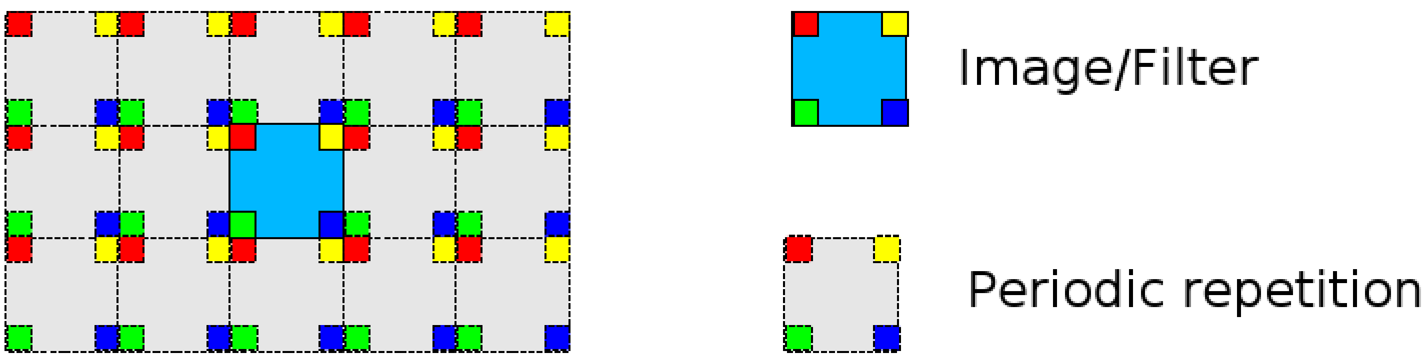


3. Fourier transform image and the centred kernel

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

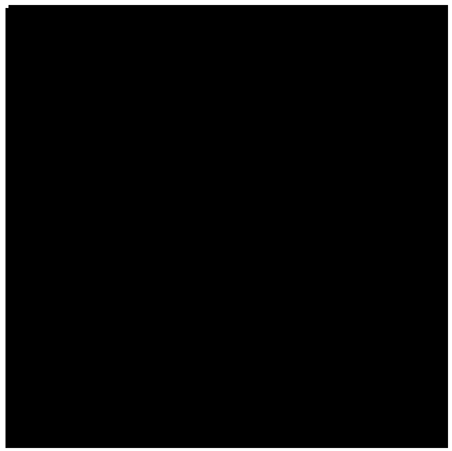
Convolution Theorem

- 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:



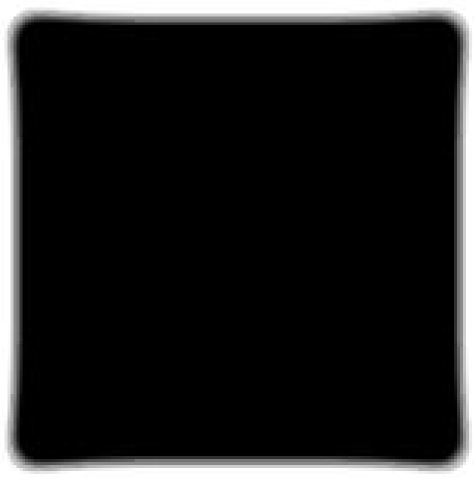
- For an image of resolution (M,N) in x and y, respectively:
  - The point (-y,-x) is identical to the point (M-y, N-x)
- Elements that lie outside the kernel after centring simply re-appear on the opposite side of the kernel matrix!

Unsharp Masking



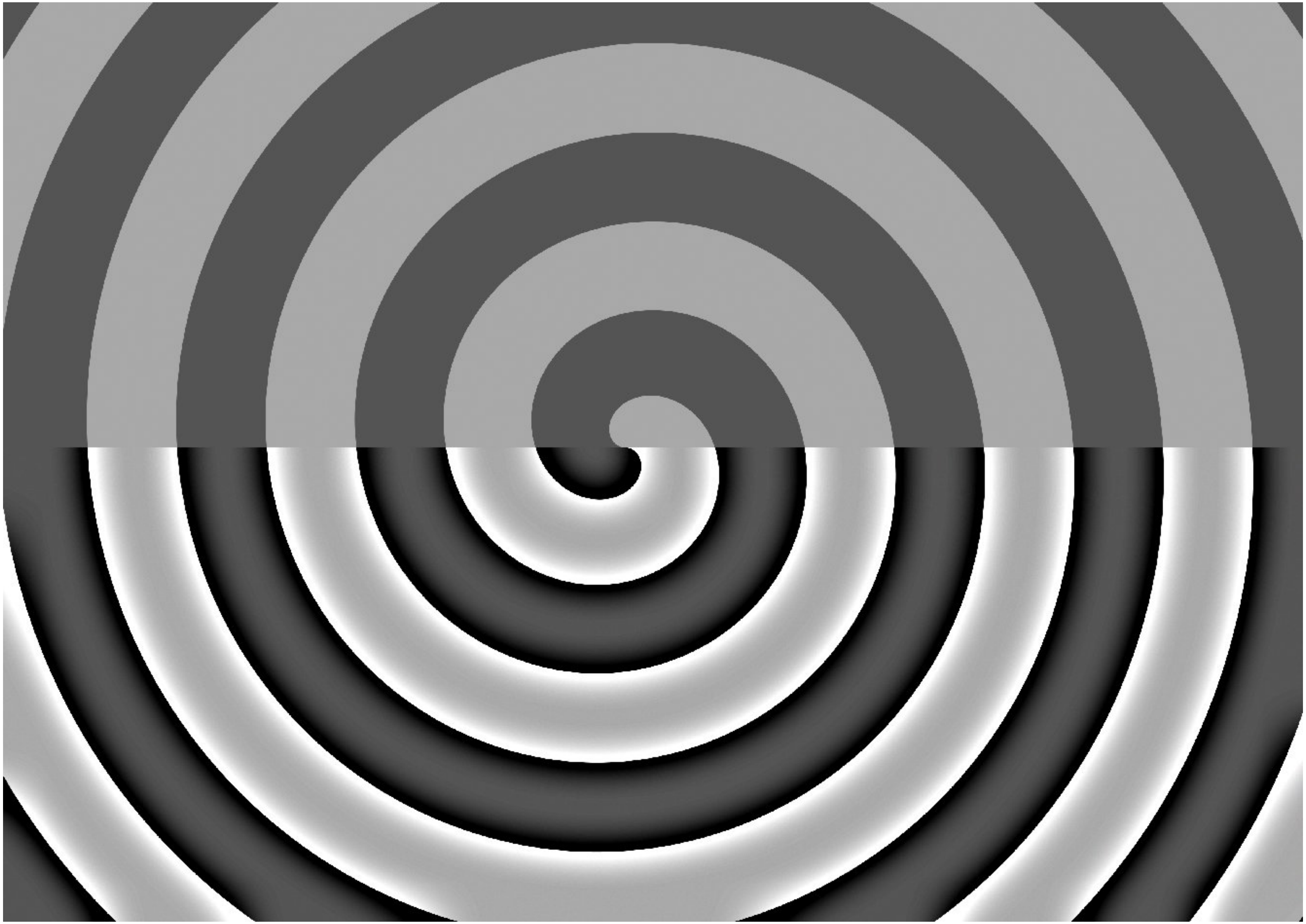
Original image

Distorted image



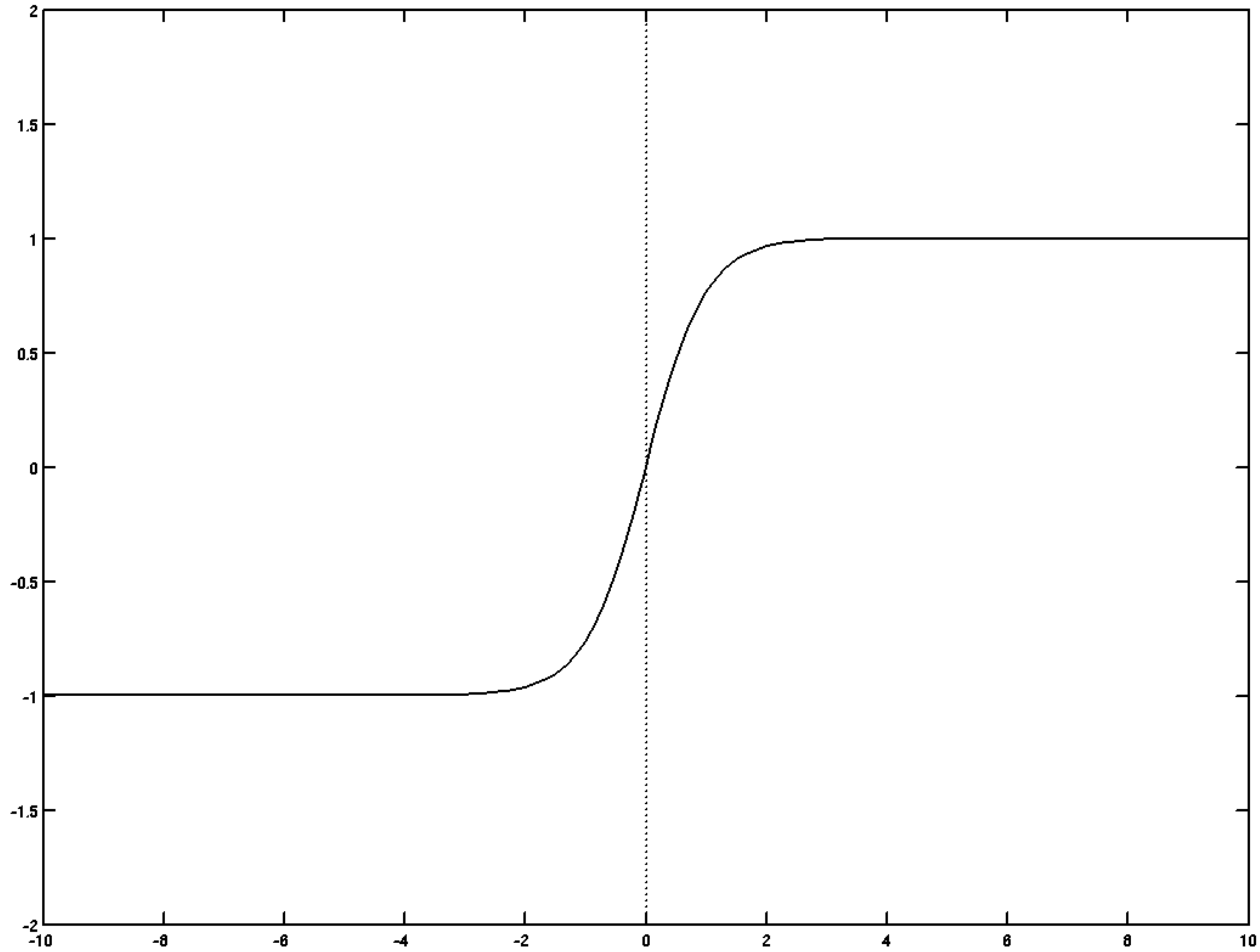
Enhanced image

Unsharp Masking



Unsharp Masking

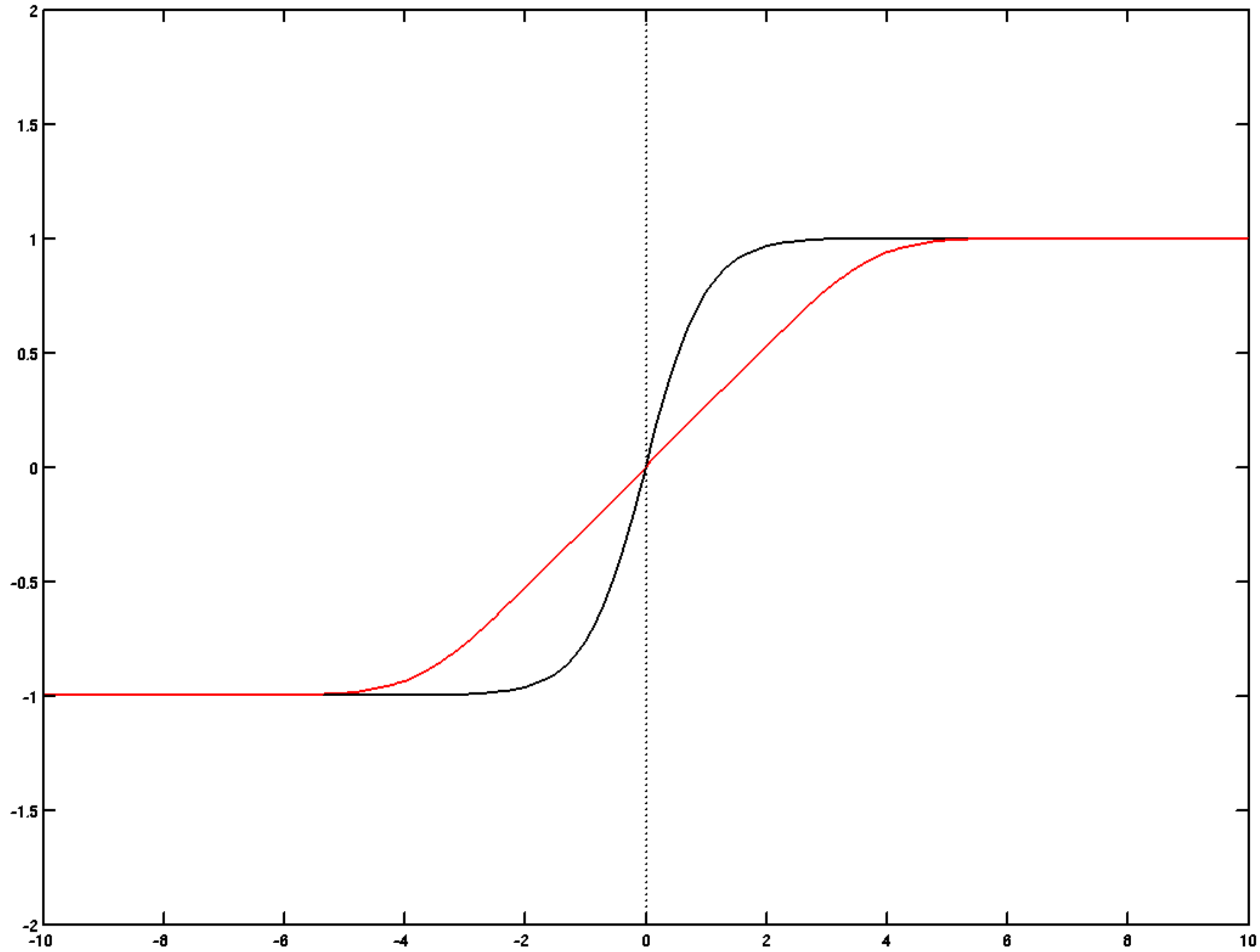
Black: Original edge y



Unsharp Masking

Black: Original edge y

Red: Degraded edge y_0



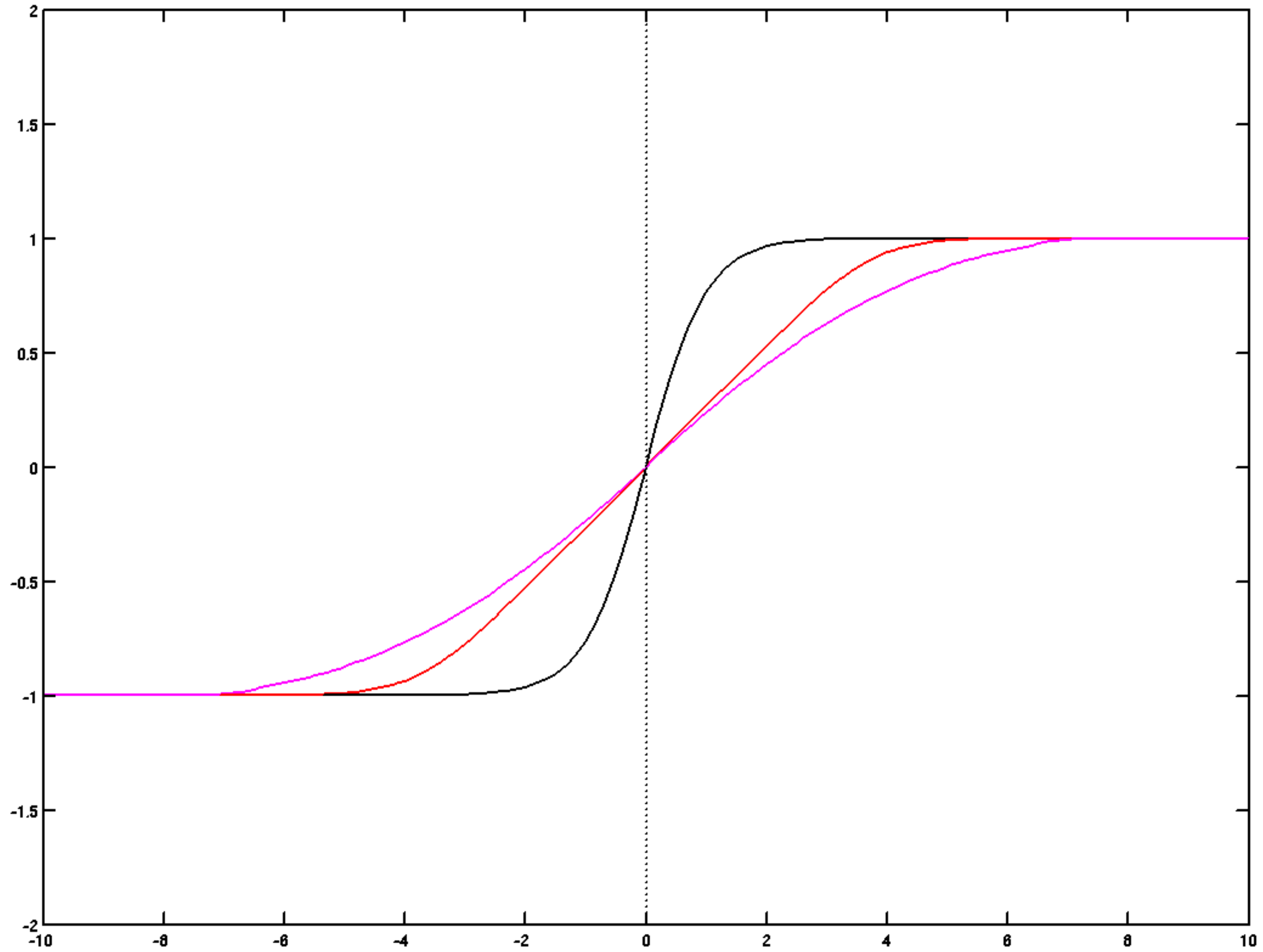
Unsharp Masking

Black: Original edge y

Red: Degraded edge y_0

USM (magenta):

1. Smooth: $y_0 \rightarrow y_1$



Unsharp Masking

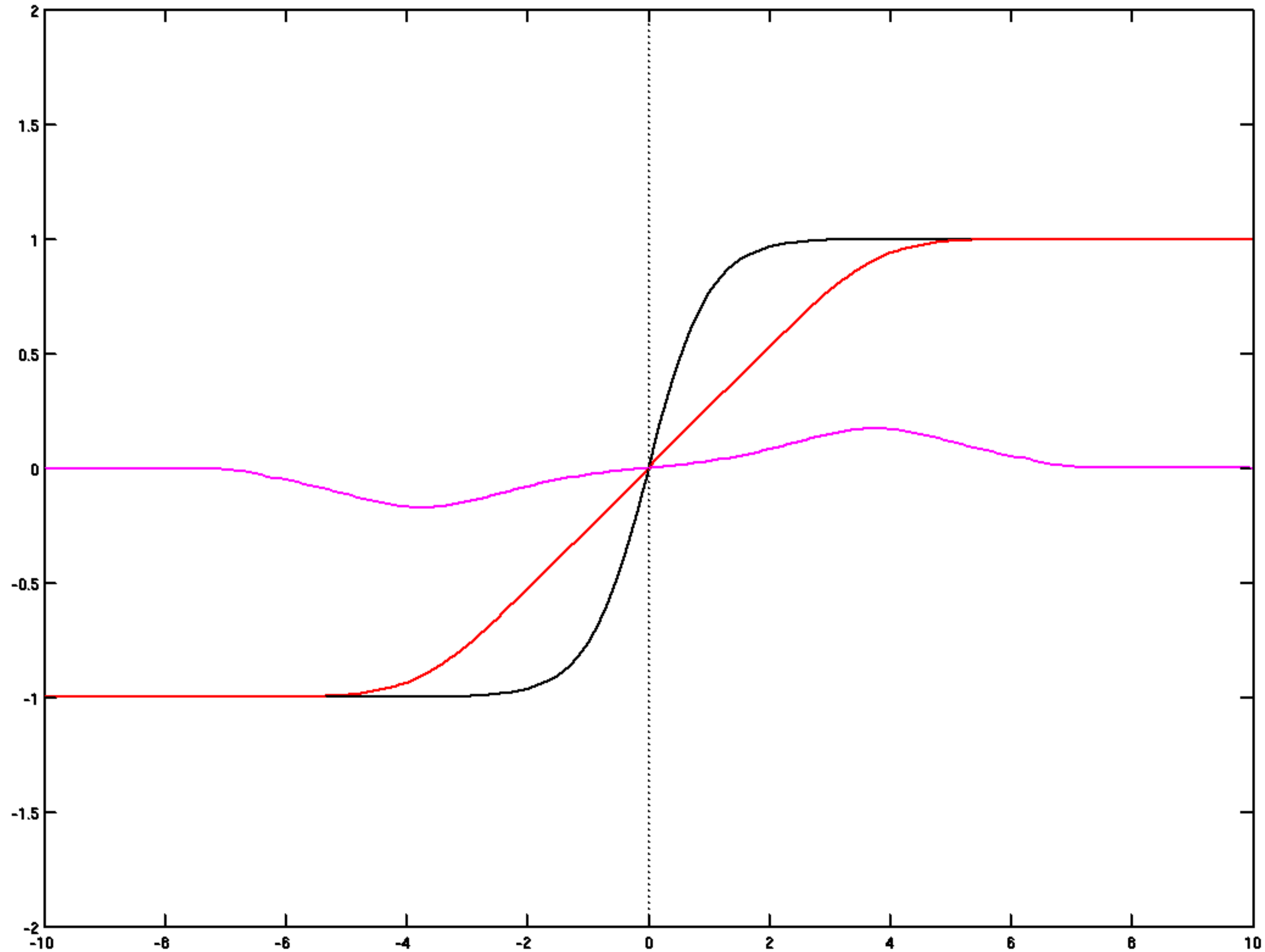
Black: Original edge y

Red: Degraded edge y_0

USM (magenta):

1. Smooth: $y_0 \rightarrow y_1$

2. Subtract: $y_2 = y_0 - y_1$



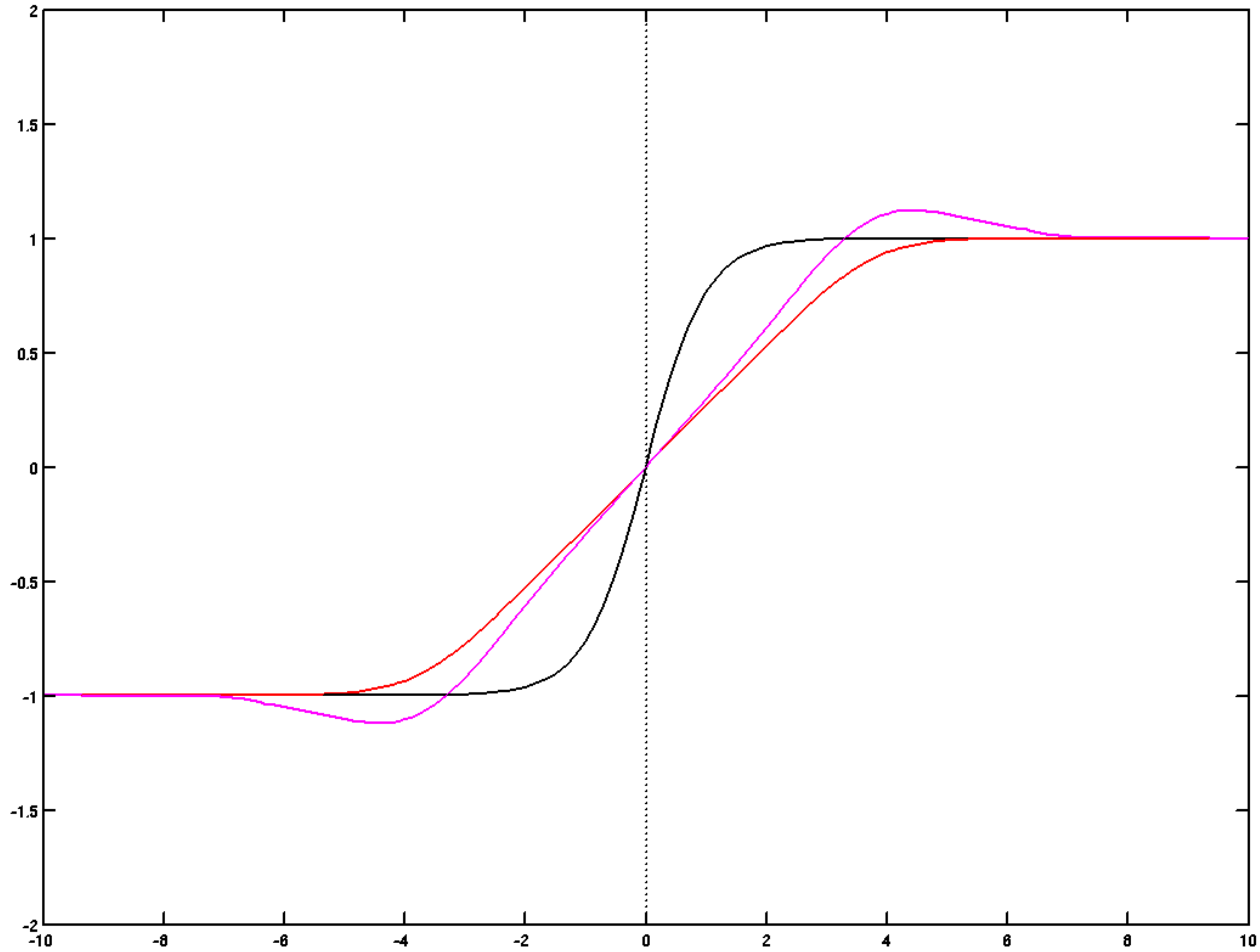
Unsharp Masking

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):

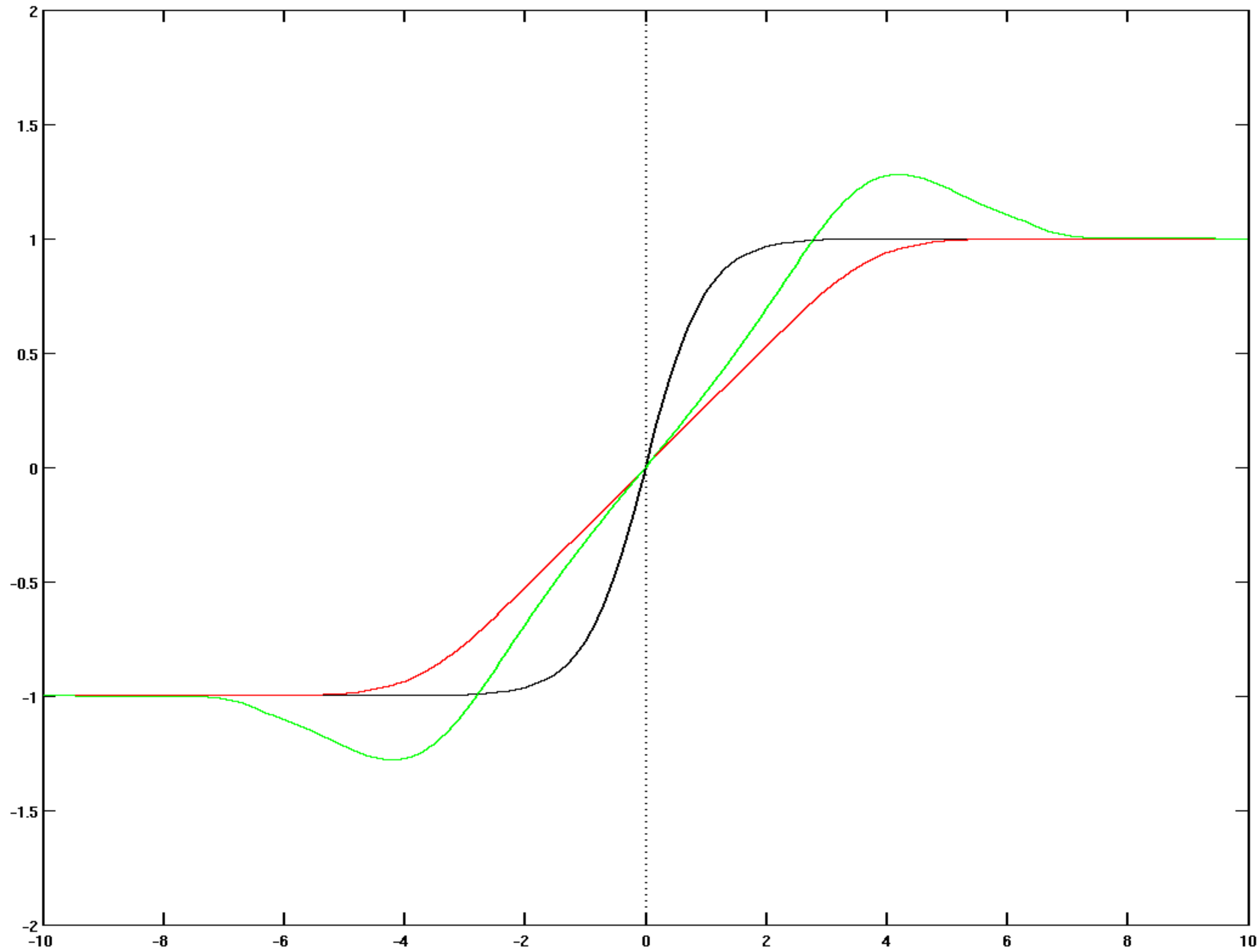
1. Smooth: $y_0 \rightarrow y_1$

2. Subtract: $y_2 = y_0 - y_1$

2.5. Scale: $ý_2 = s * y_2$

3. Add: $y_3 = y_0 + ý_2$

→ Final (green)



Unsharp Masking

Original image

Enhanced image

Unsharp Masking

Original image

Enhanced image
with threholding

Unsharp Masking

Original image

Sharpened image

Highly sharpened image

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:

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

Parameter:

k, T, s

Given

```
main(int argc, char** argv)
```

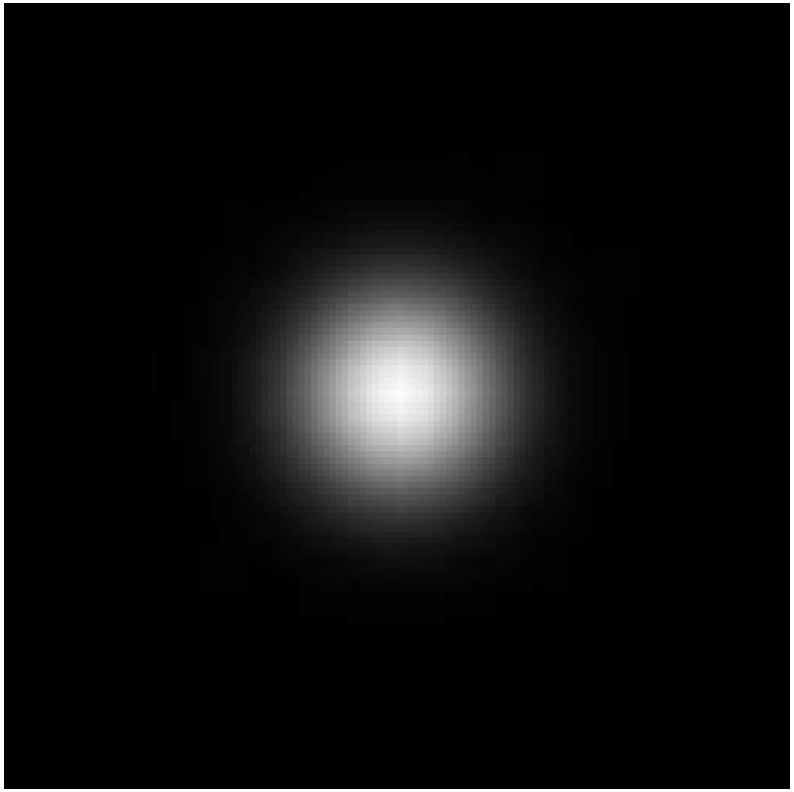
- Declares variables
- Displays and saves images
- Applies USM-function
- → Measures and saves time

To Do

`void createKernel(Mat& kernel, int kSize, string name)`

`kernel`:	the generated filter kernel
`kSize`:	the kernel size
`name`:	specifies type of filter, eg "gaussian", "average", ...

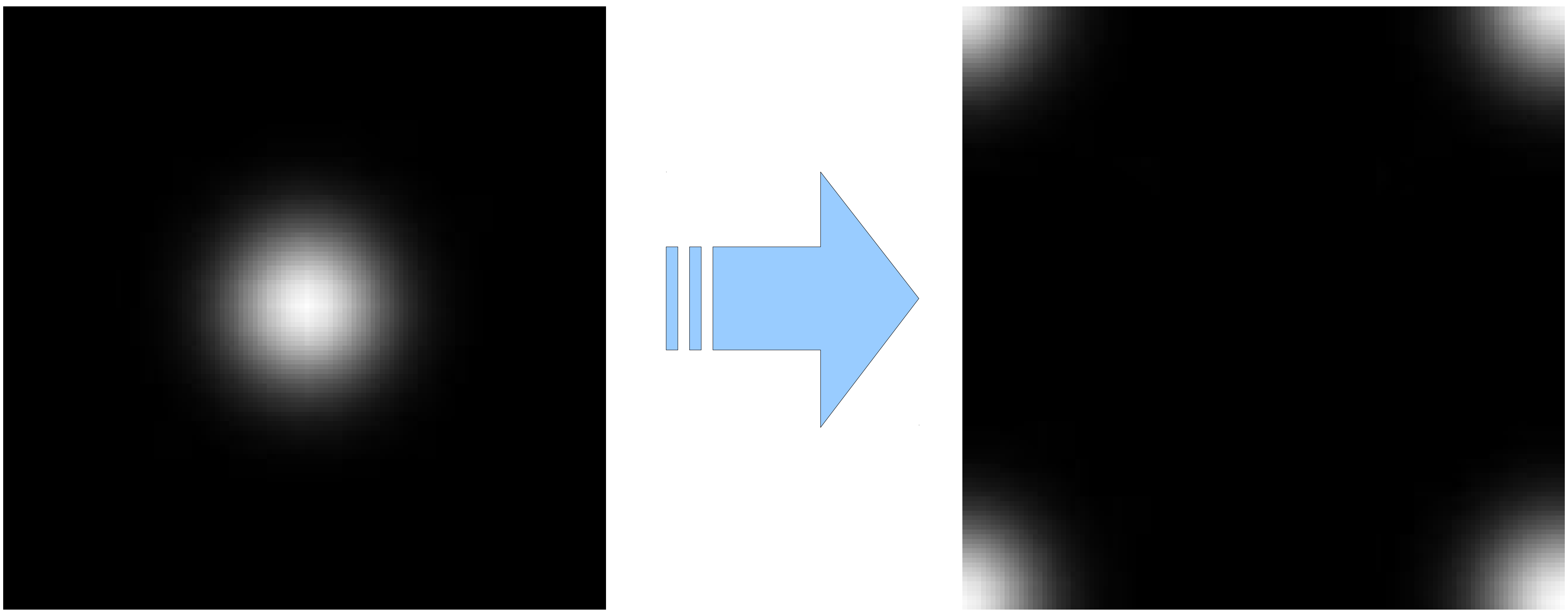
- Calculates (potentially different) filter kernels
- Filter type is specified by <name>
- In this exercise: Only Gaussian kernel is needed
  - $f(x,y)=\frac{1}{2\pi\sigma_x\sigma_y}\exp\left(-\frac{1}{2}\left[\frac{(x-\mu_x)^2}{\sigma_x^2}+\frac{(y-\mu_y)^2}{\sigma_y^2}\right]\right)$
  - Mean and variance are determined by kernel size (eg. $\sigma = kSize/5$)
  - Maximal value at centre
  - Must integrate to one



To Do

```
void circShift(Mat& in, Mat& out, int dx, int dy)
```

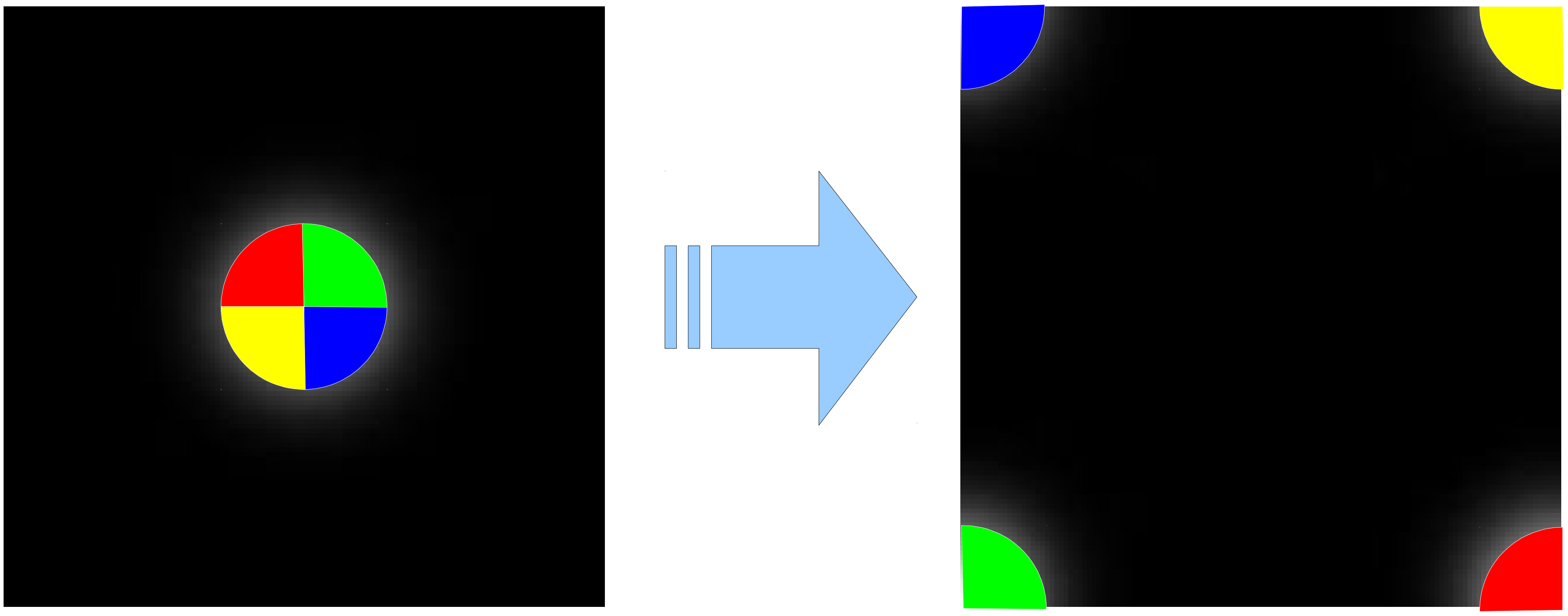
- Performes circular shift in (dx,dy) direction on matrix <in>



To Do

`void circShift(Mat& in, Mat& out, int dx, int dy)`

- Performes circular shift in (dx,dy) direction on matrix <in>



To Do

```
void frequencyConvolution(Mat& in, Mat& out, Mat& kernel)
```

- 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}	. . .
. . .	. . .	. . .	. . .	. . .	. . .

To Do

`void usm(Mat& in, Mat& out, int smoothType, int size, double thresh, double scale)`

`in, out`:	in- and output image, respectively
`smoothType`:	smoothing type (1==spatial, 2==frequency)
`size`:	kernel size k
`thresh`:	threshold T
`scale`:	scale s

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

To Do

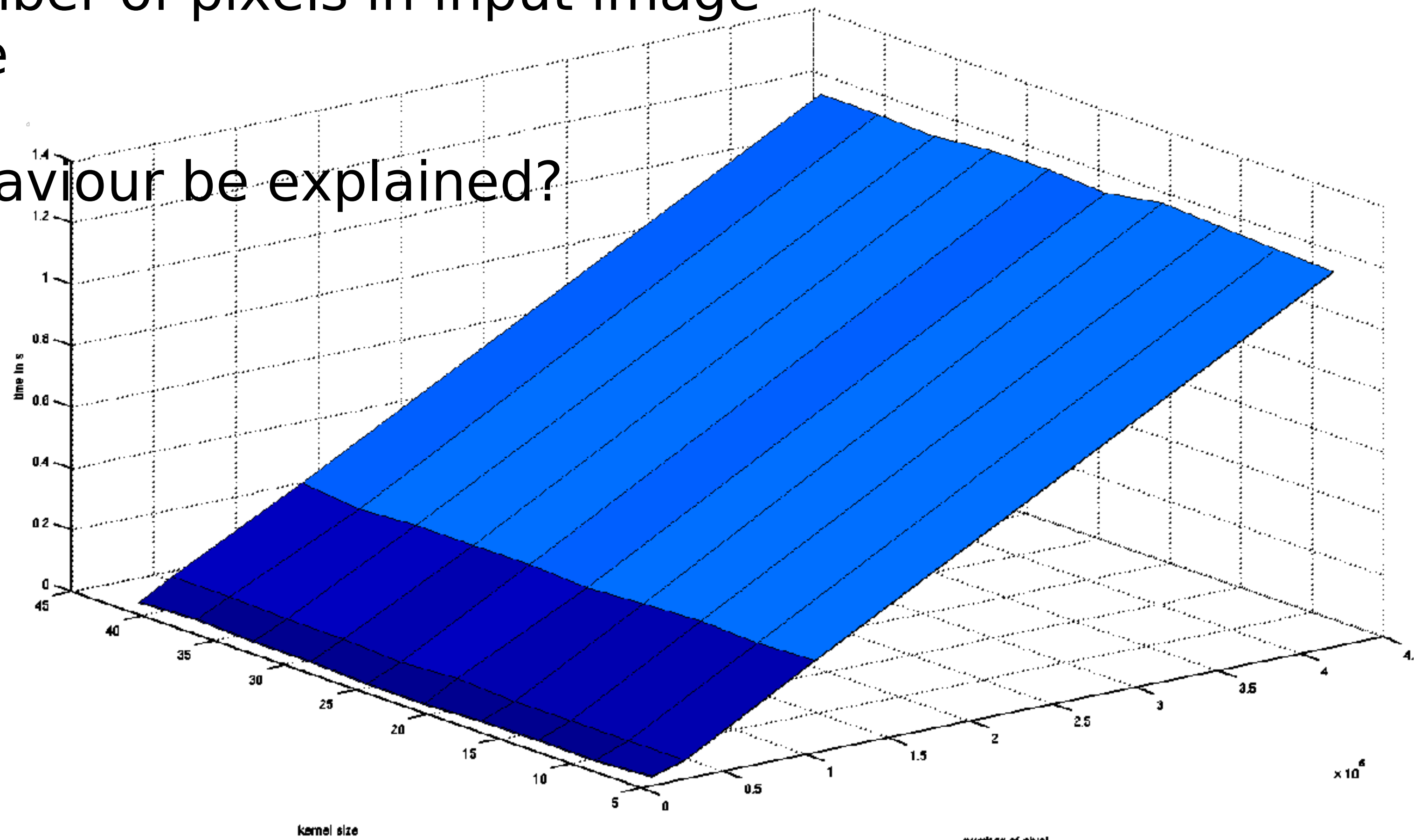
Make images

- Different sizes
- Eg: 256x256 => 1024x1024
- (not necessarily square)

To Do

- Prepare two graphs that describe time behaviour of convolution by usage of 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?



To Do

Main function: Set parameter for USM

- Size of operator is already defined
- Which threshold T?
- Which scaling?

```
int numberOfKernelSizes = 10;         // number of differently sized smoothing kernels
double thresh = 0;                    // difference necessary to perform operation
double scale = 1;                     // scaling of edge enhancement

[...]

for(int s=1; s<=numberOfKernelSizes; s++){

    int size = 4*s+1; // use this size for smoothing

    [...]

    // perform unsharp masking
    usm(v_plane, tmp, type+1, size, thresh, scale);

  }
}
```

Exams

- Mid-term
  - Friday, **01.06.2012, 10:15pm, E020**
  - Duration: ca. 30 min
  - No grade, but pass is necessary to take part at the final exam

- Final
  - Tuesday, **10.07.2012, 10:15pm**
  - Duration: ca. 60 min

- Topics from lecture and exercise
- Questions in English, answers in English or German
- No books, no calculator, no script, no paper, ...