

COL7655

Foundations of Visual Computing

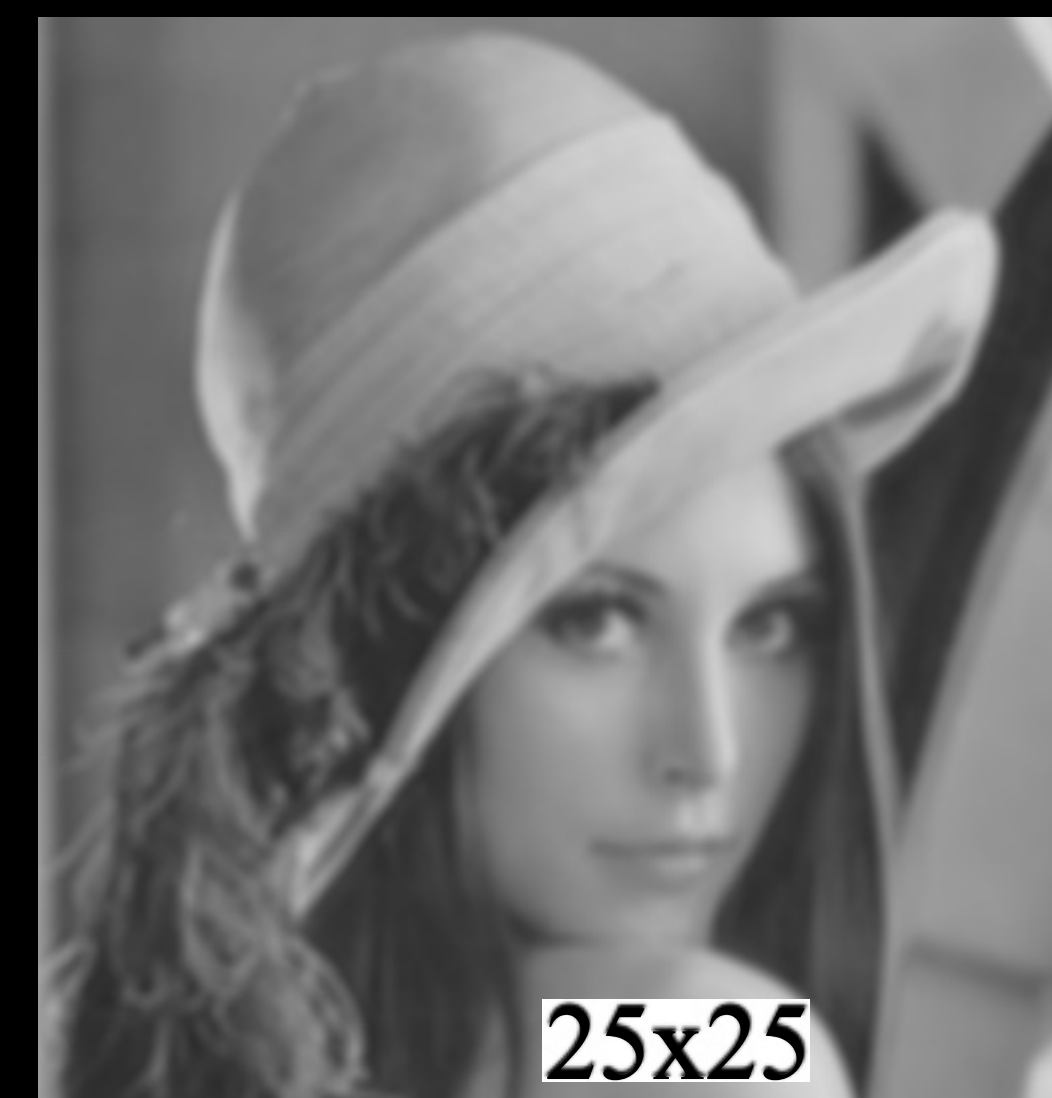
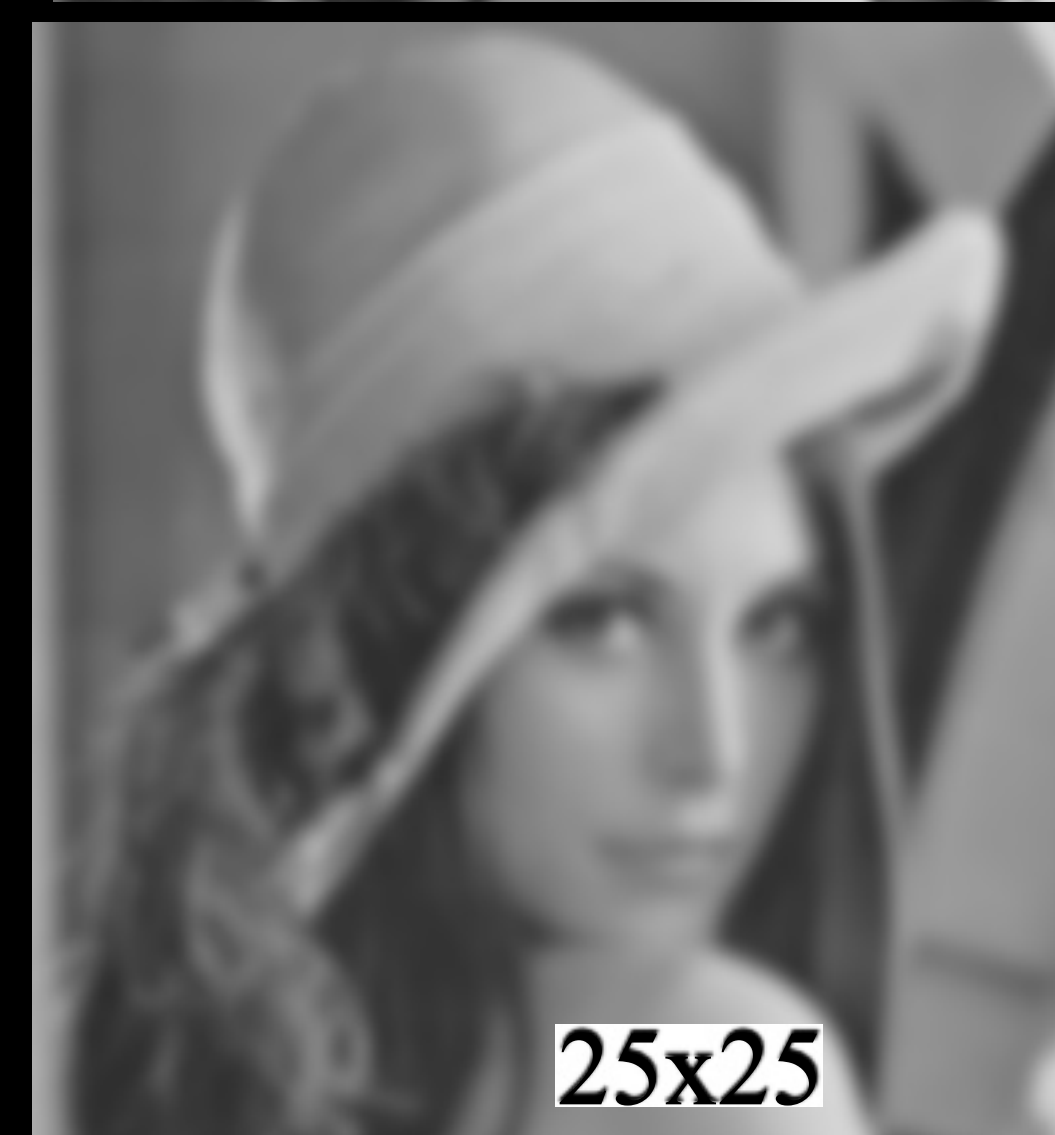
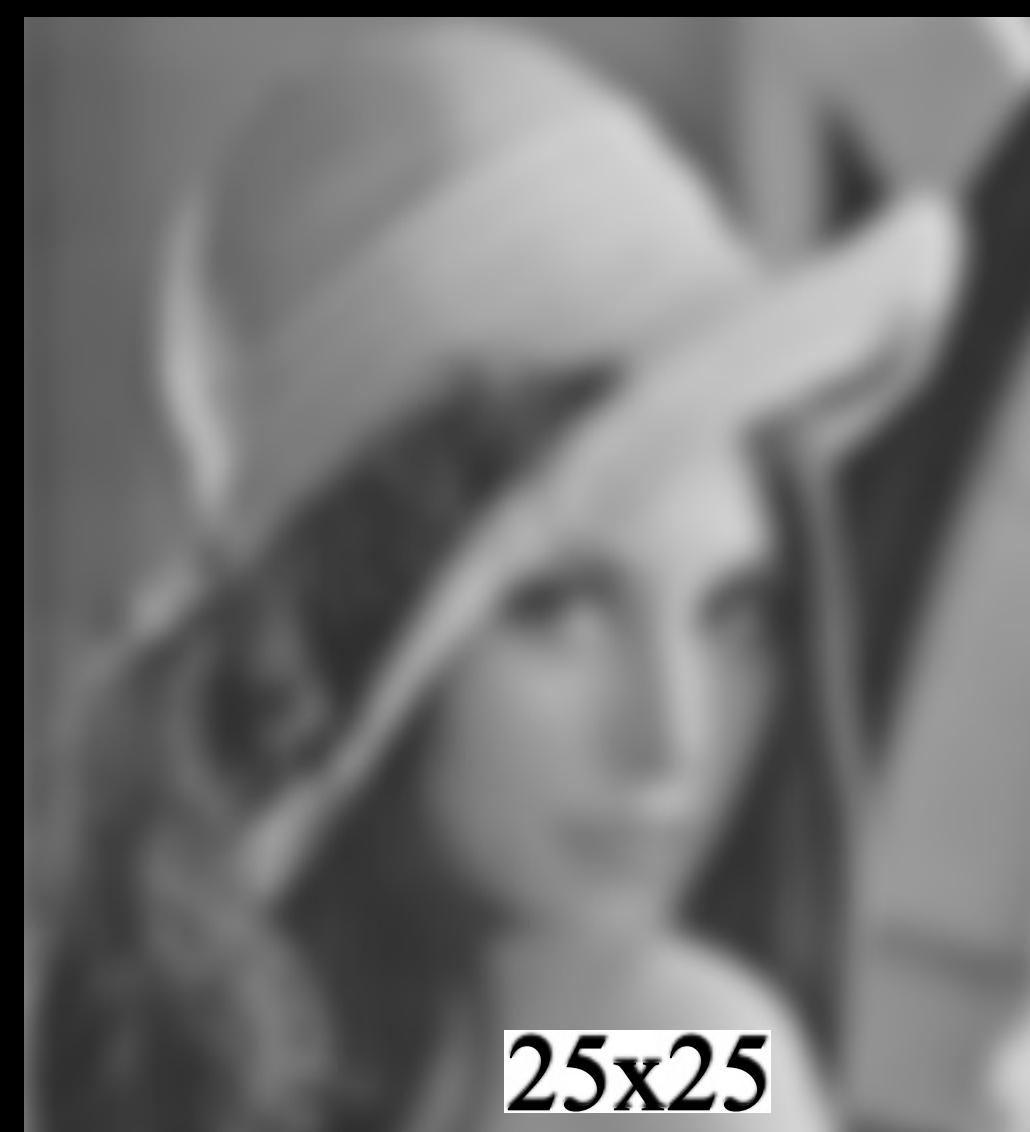
Subodh Kumar

subodh@cse.iitd.ac.in

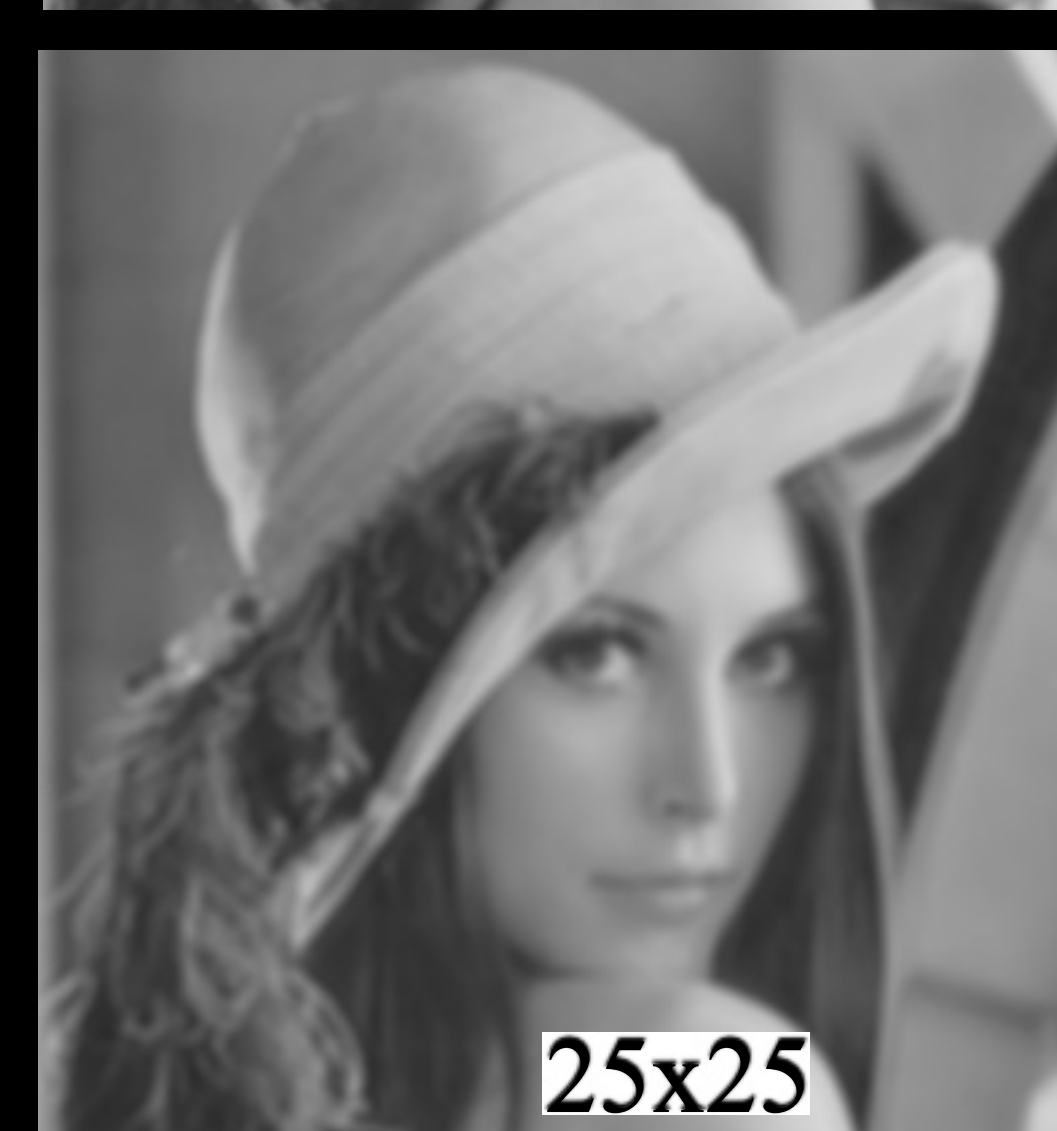
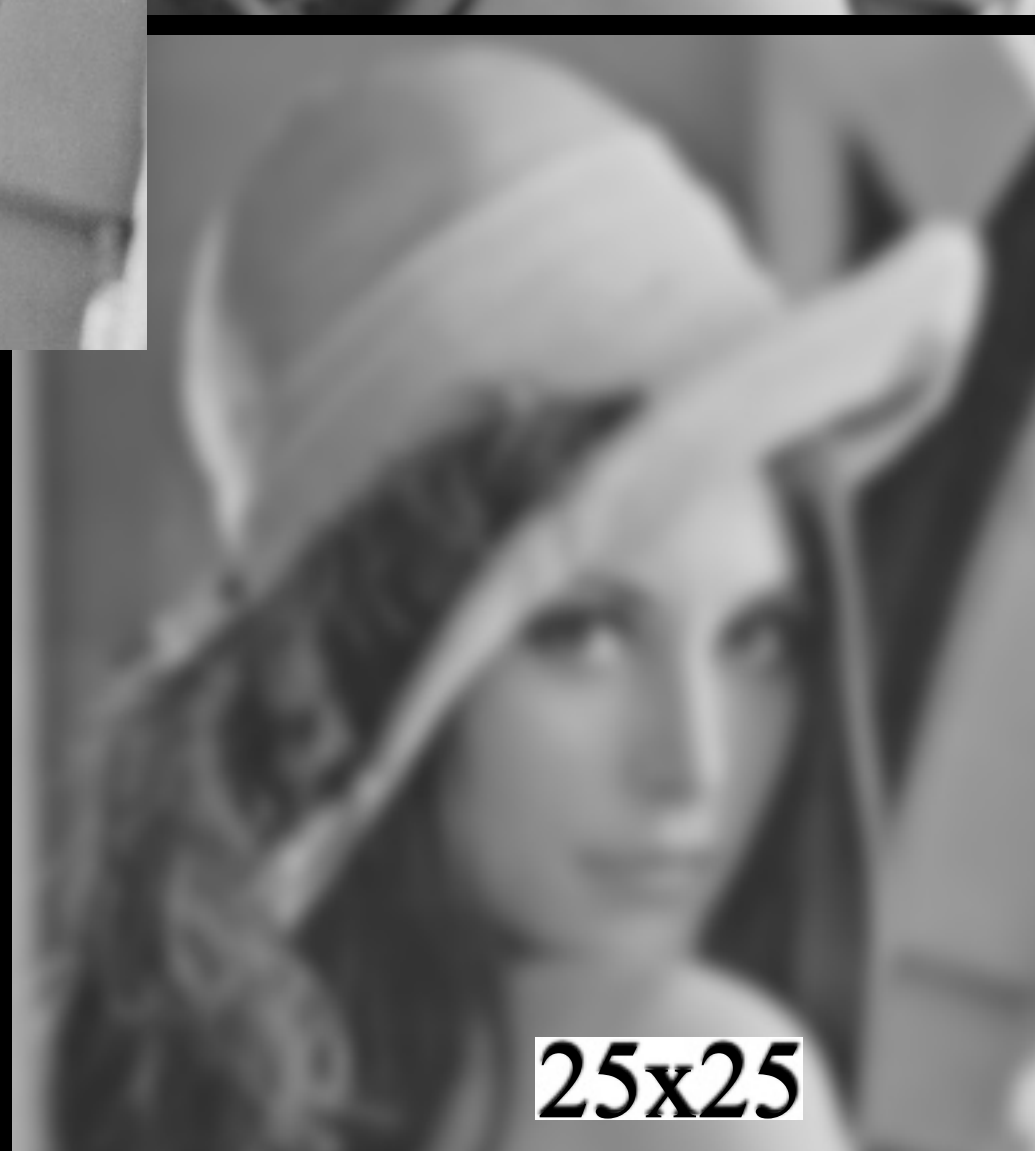
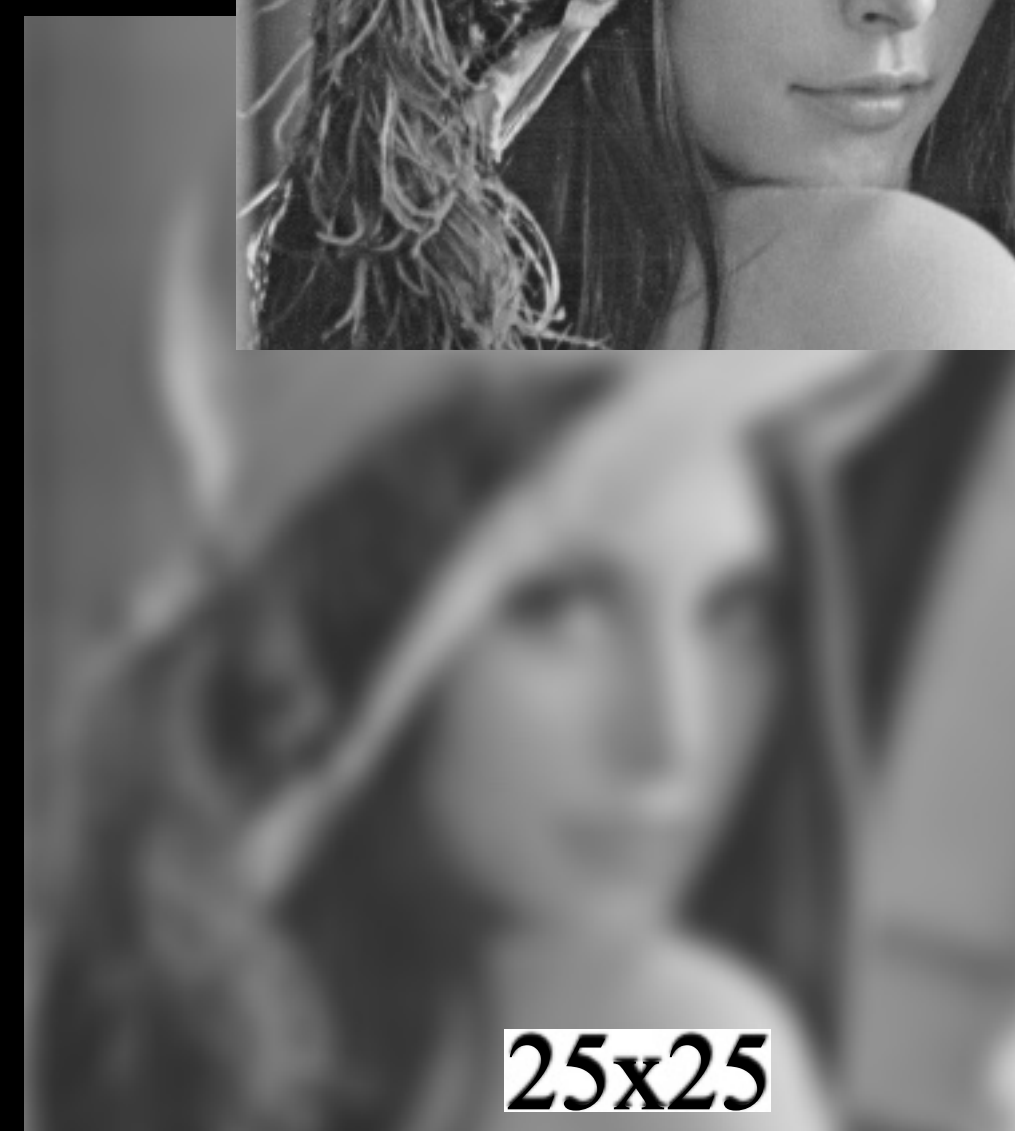
COL7655

Filters

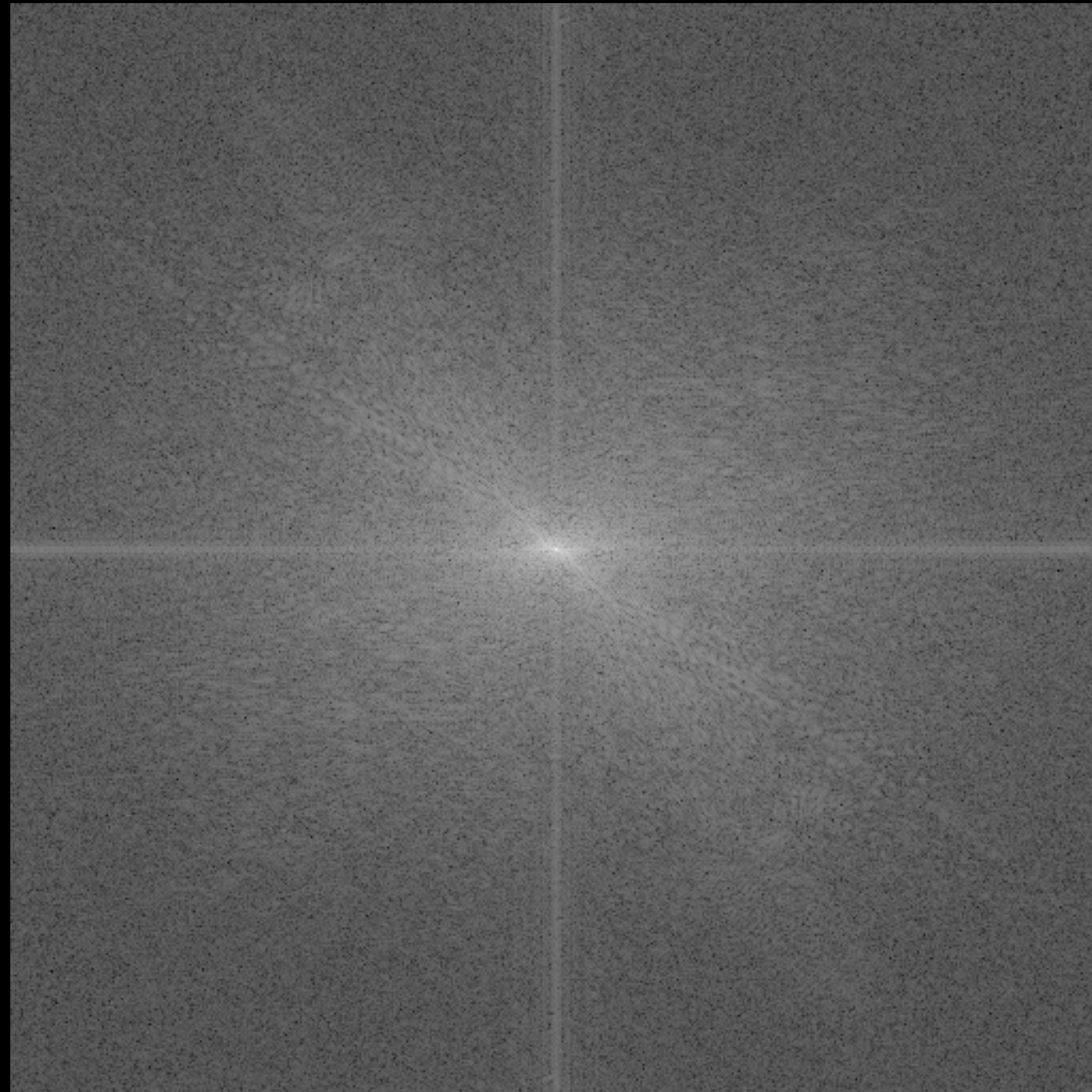
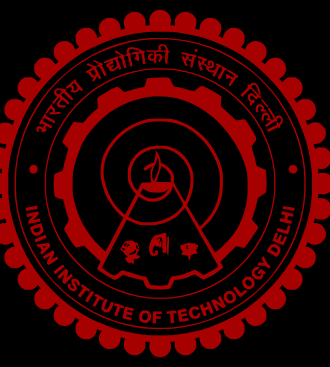
Filter Comparison



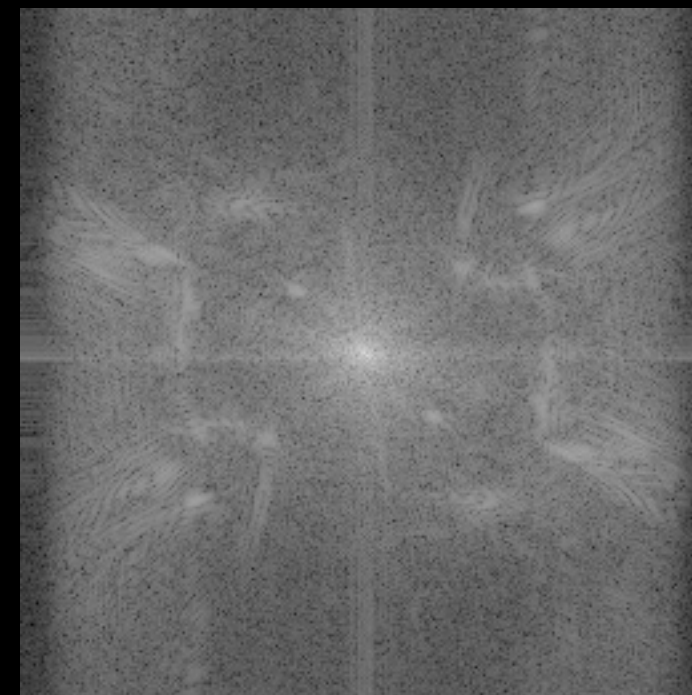
Filter Comparison



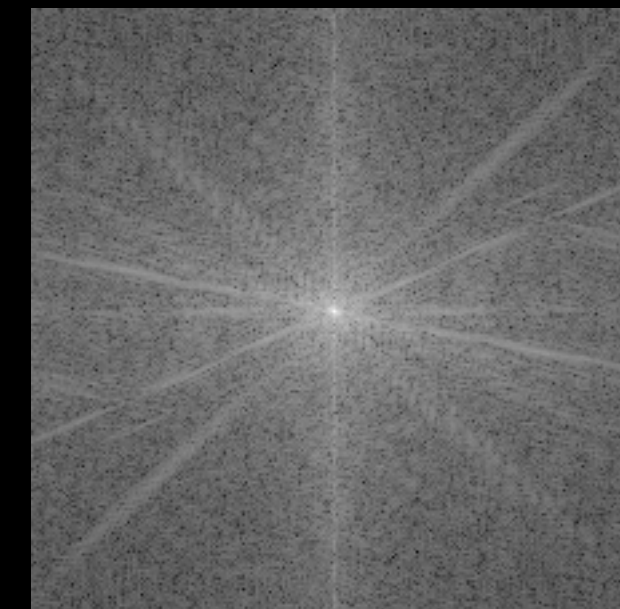
Frequency Domain



Lena



Barbara



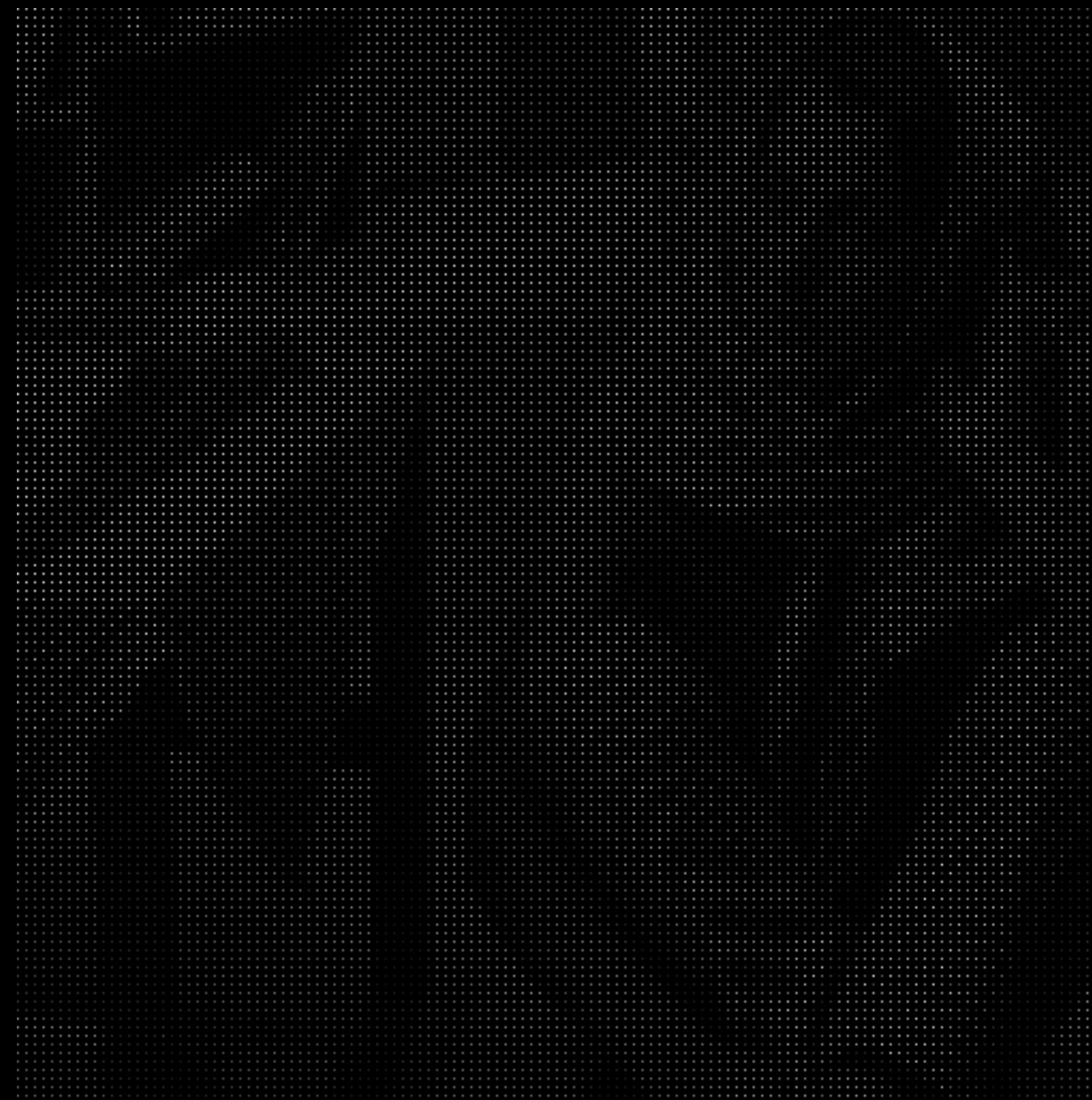
Cameraman

Sampling High Frequency Image

With pre-filter



Original image, 512x512



Sampled every 4 pixels

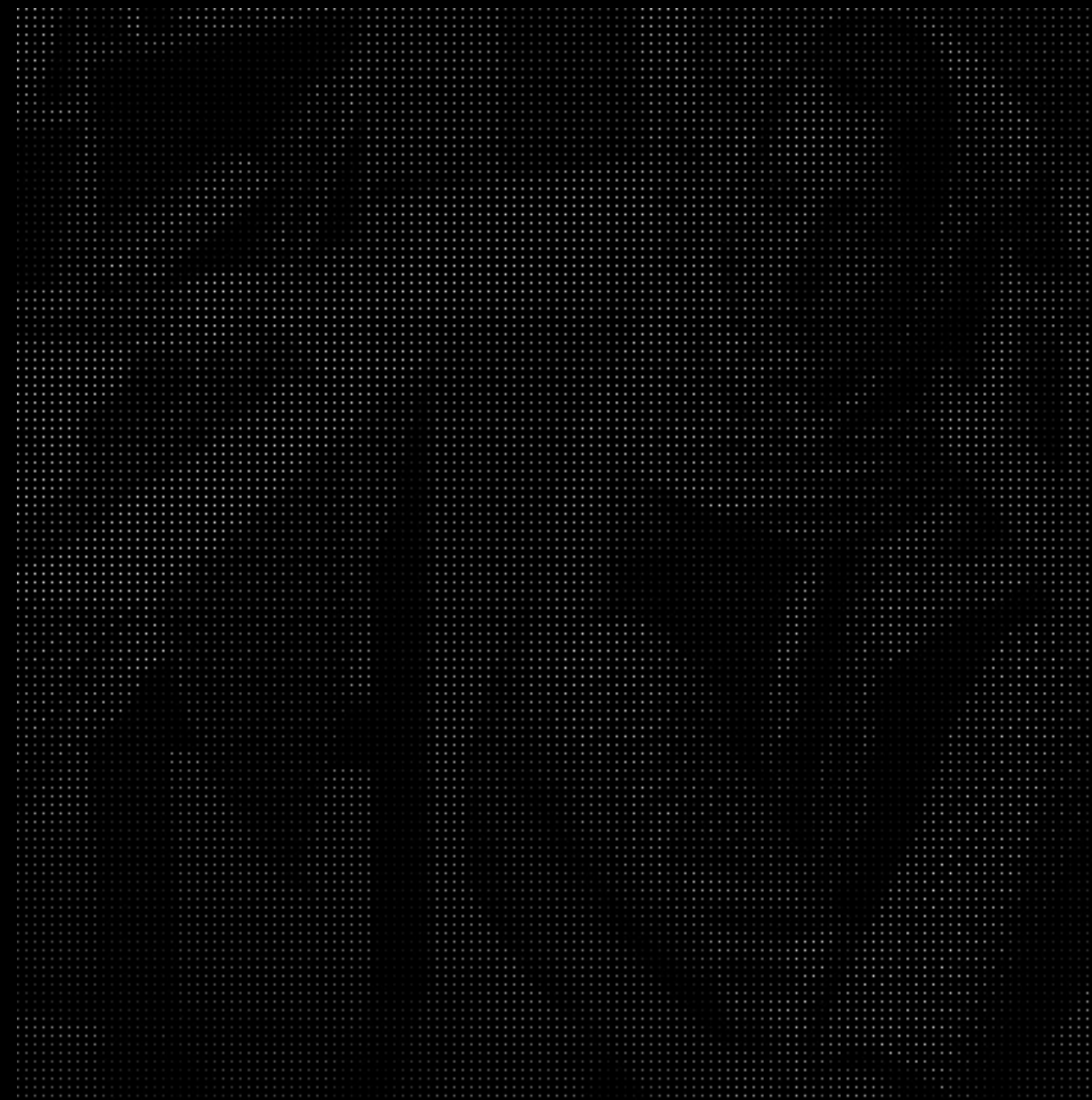


Rescaled image, 128x128

Sampling High Frequency Image



Blurred image, 512×512

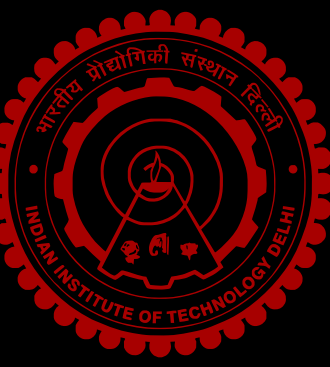


Sampled every 4 pixels



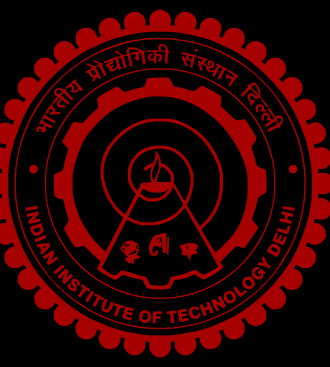
Rescaled image, 128×128

Rescale



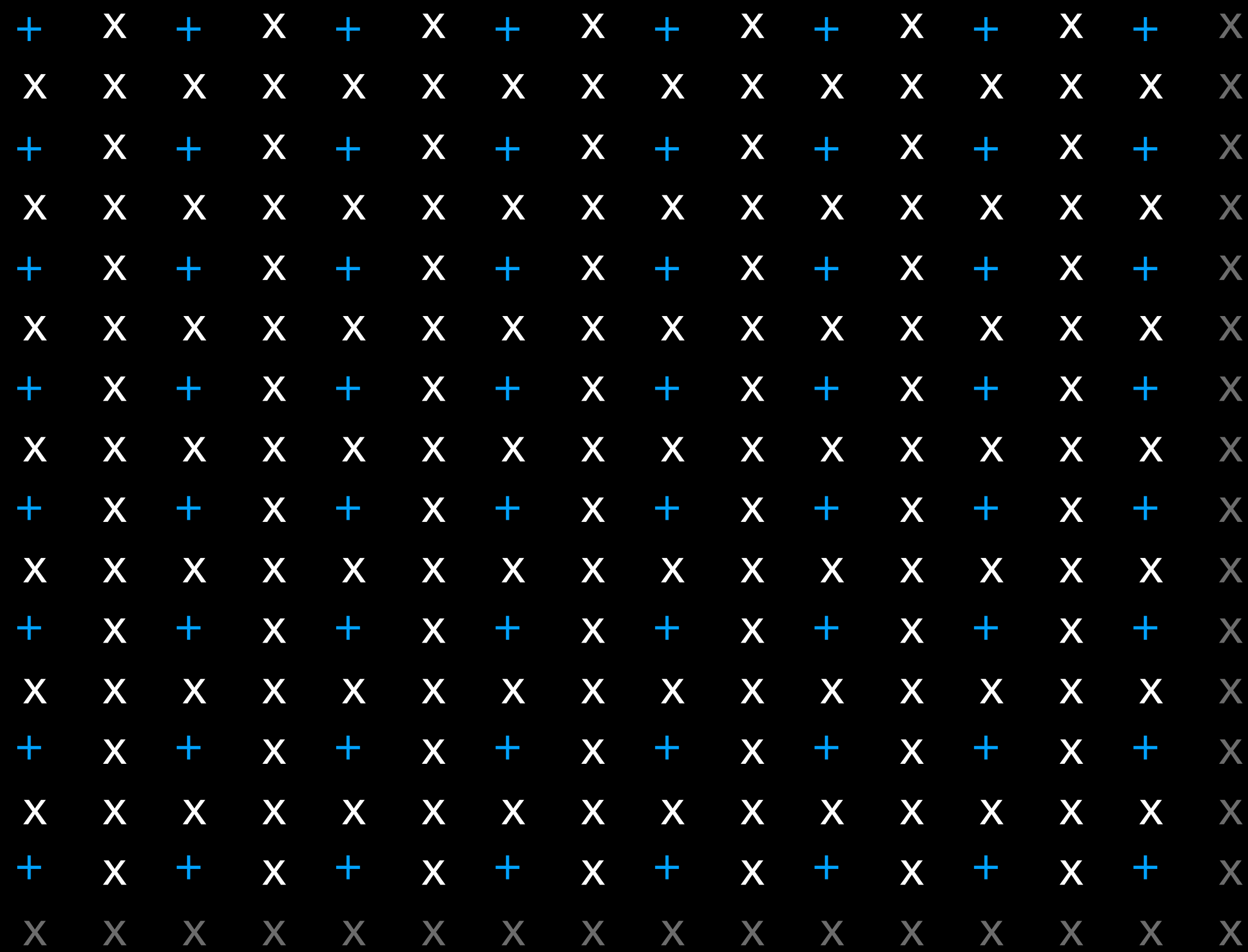
- 3x3 image $\rightarrow$ 6x6?

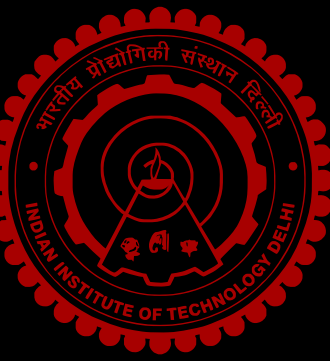
a	.	b	.	c	.
.	.	.	.	.	.
d	.	e	.	f	.
.	.	.	.	.	.
g	.	h	.	i	.
.	.	.	.	.	.



How to Rescale

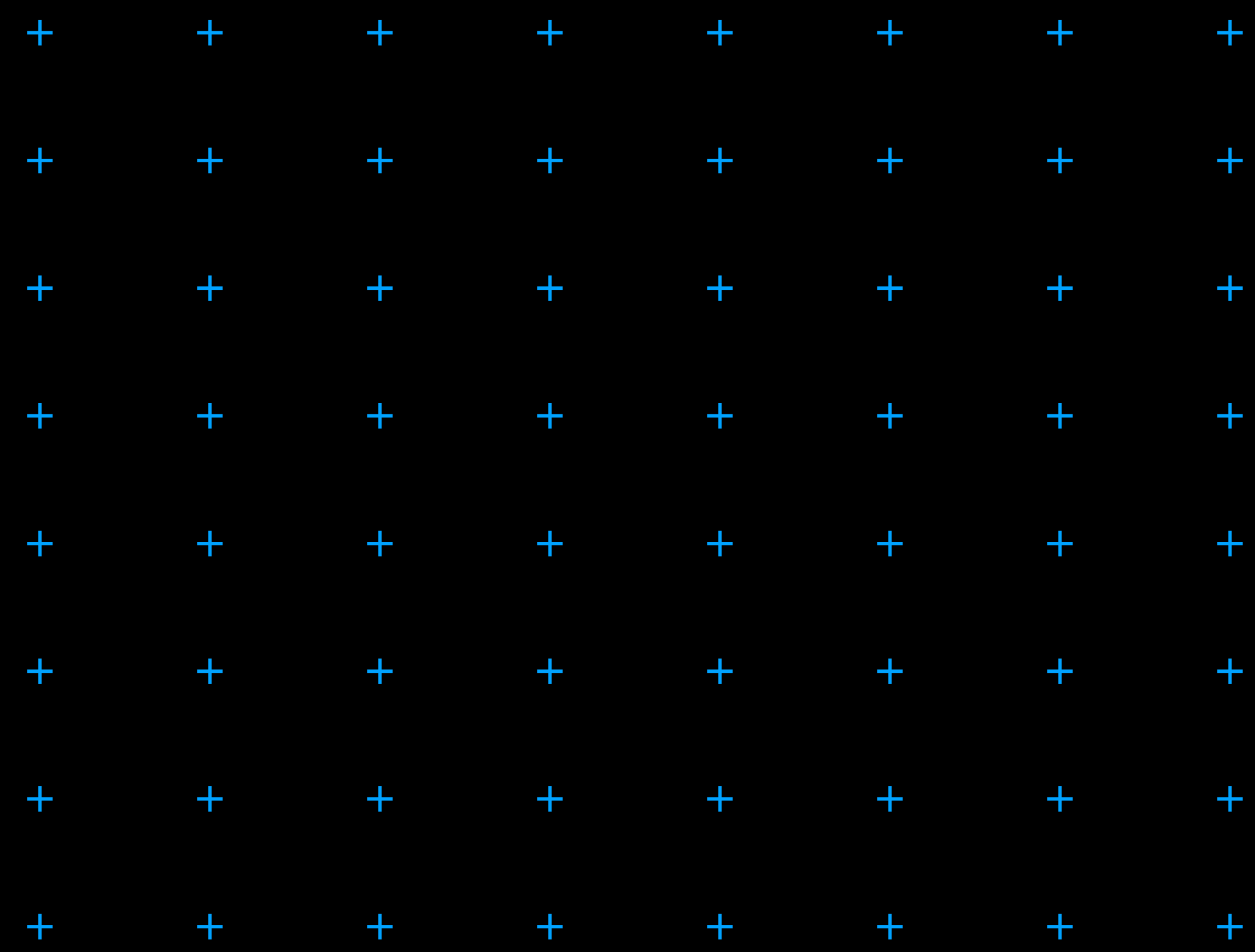
Interpolation is also filtering





How to Rescale

Interpolation is also filtering



What are the extents
of the image?

Image Space

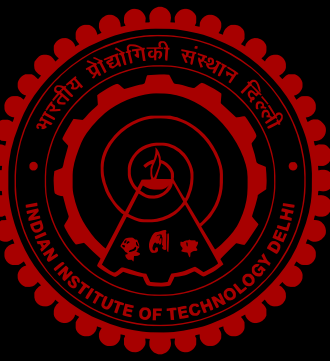
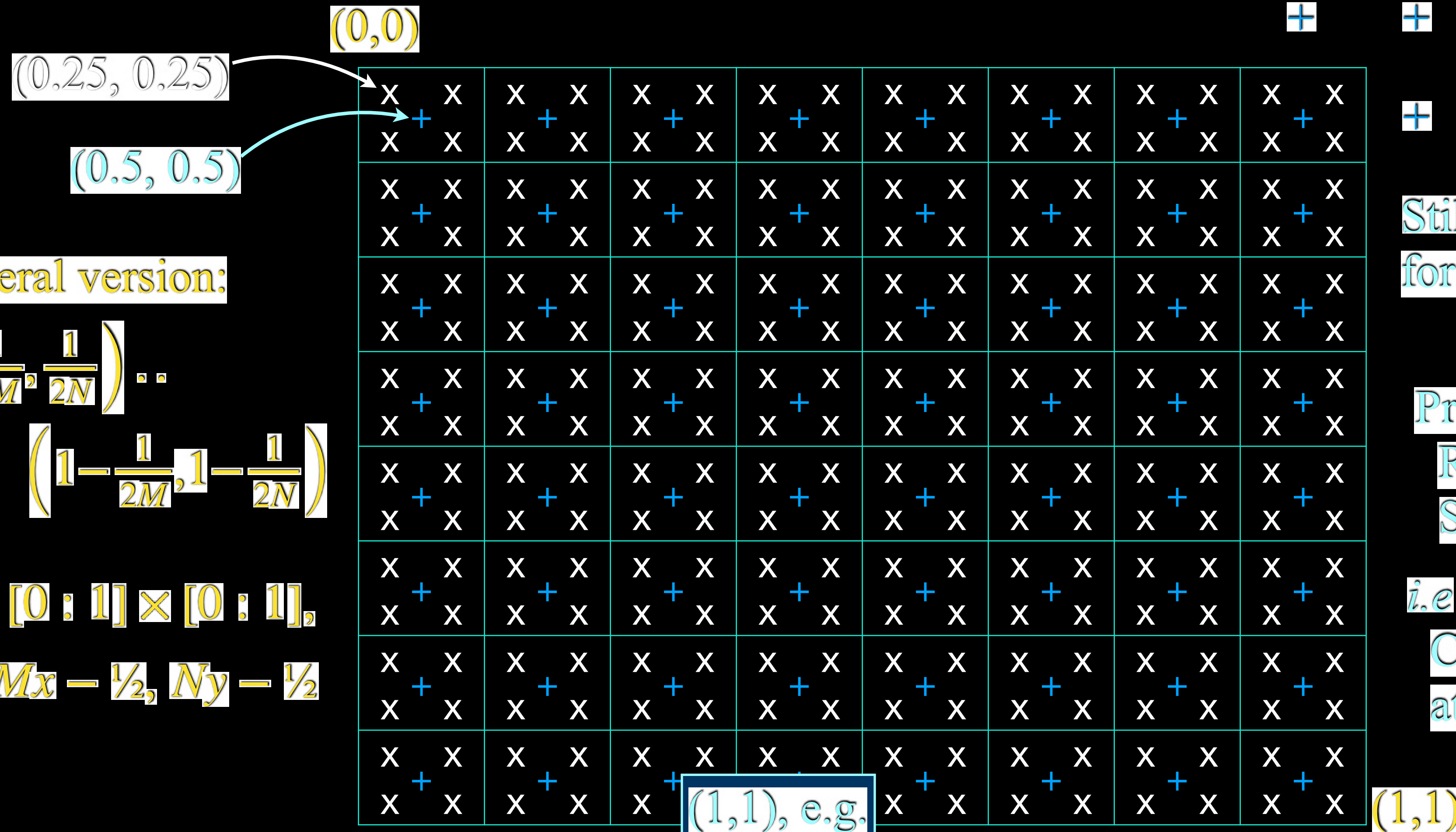


Image Space



The general version:

$$\left(\frac{1}{2M}, \frac{1}{2N} \right) \cdots \left(1 - \frac{1}{2M}, 1 - \frac{1}{2N} \right)$$

$$\forall (x, y) \in [0 : 1] \times [0 : 1],$$

$$j, k = Mx - \frac{1}{2}, Ny - \frac{1}{2}$$

Still need border extension
for interpolation

Principle:

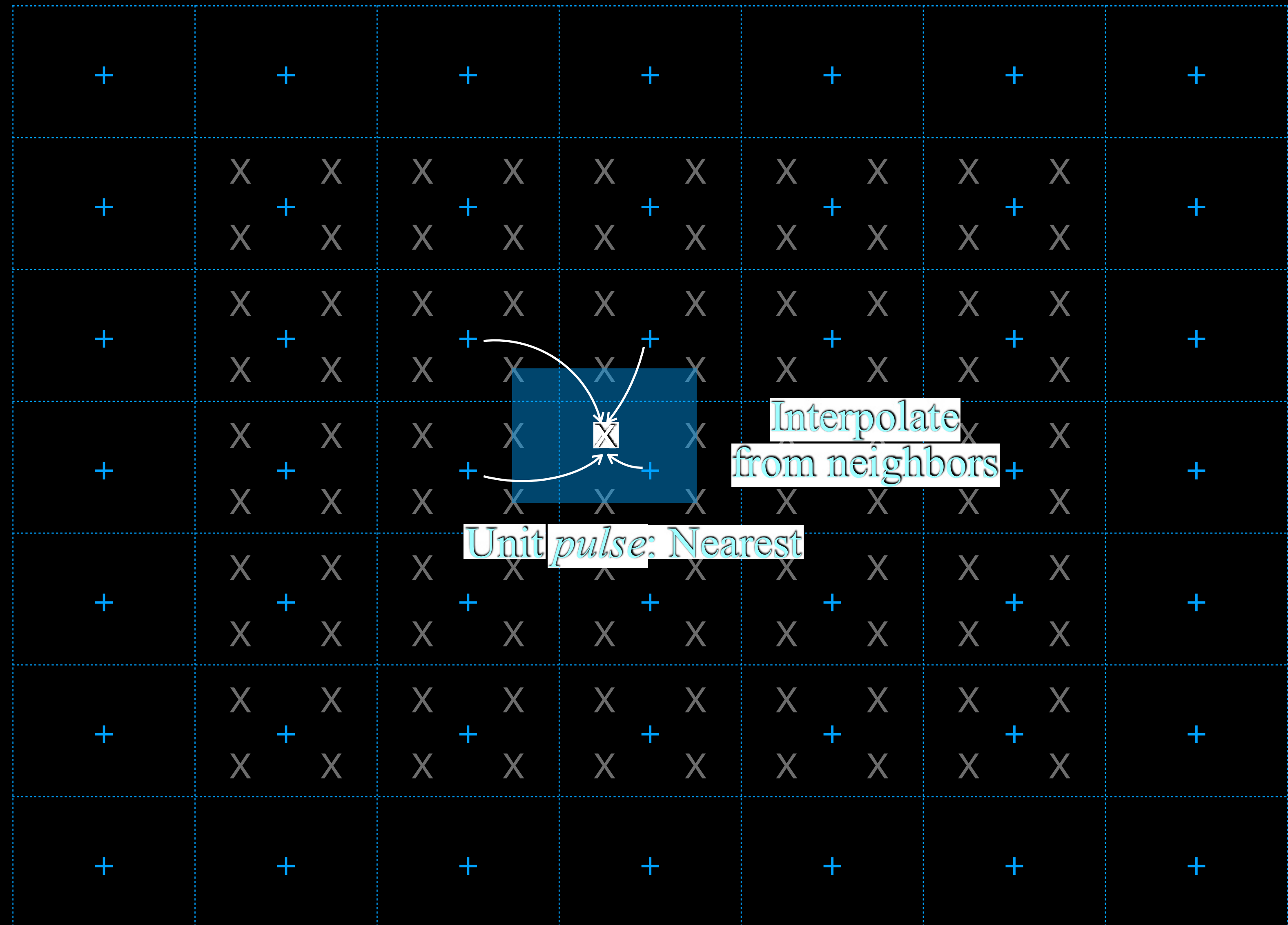
Reconstruct from samples;
Sample from reconstruction

i.e

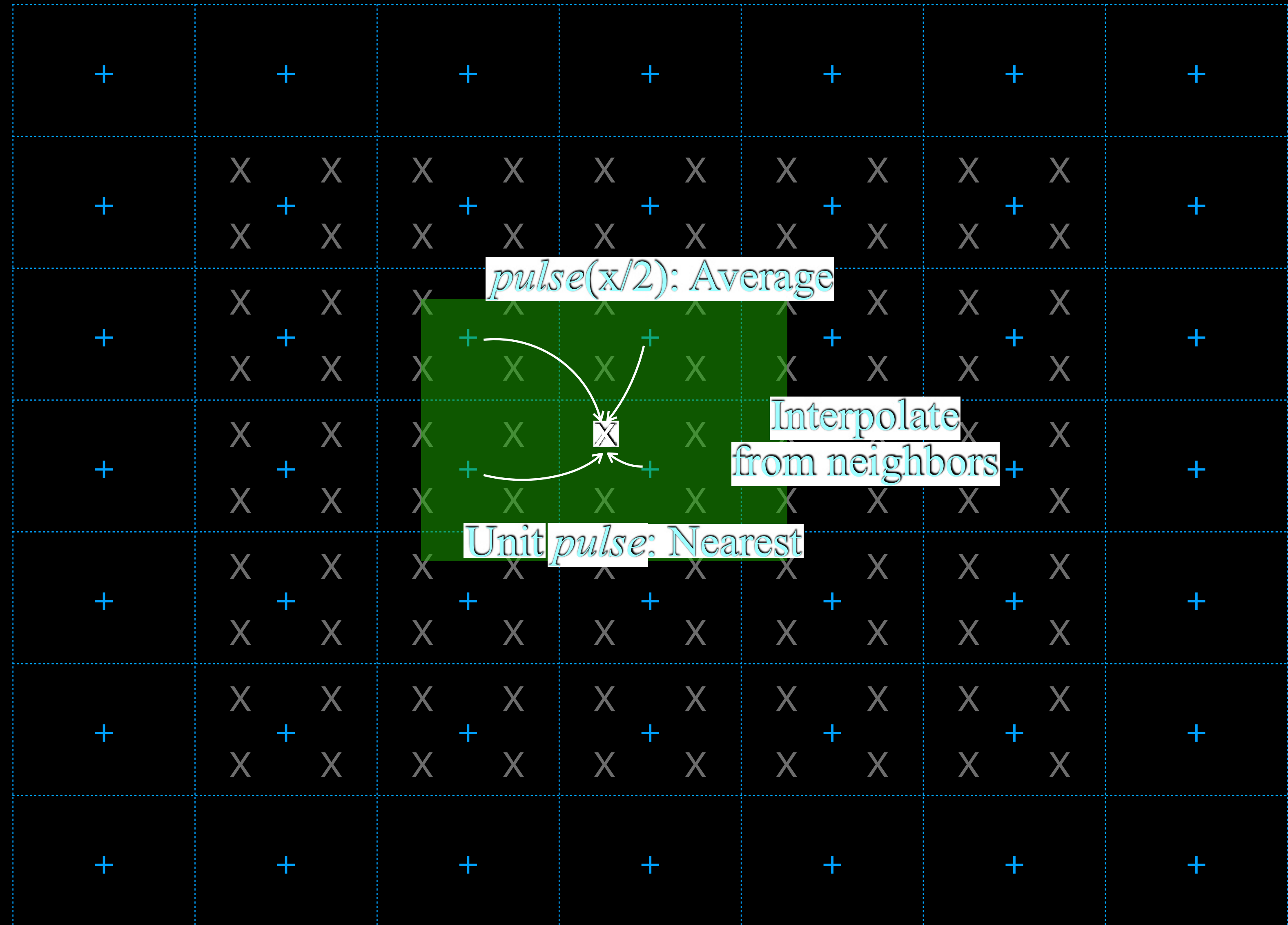
Compute convolution
at sample locations

For $M \times N$ pixels, if inter-pixel distance is $(\Delta x, \Delta y)$, $(\Delta u, \Delta v) = \left(\frac{1}{\Delta x}, \frac{1}{\Delta y} \right)$. Image rectangle = $(M\Delta x \times N\Delta y)$

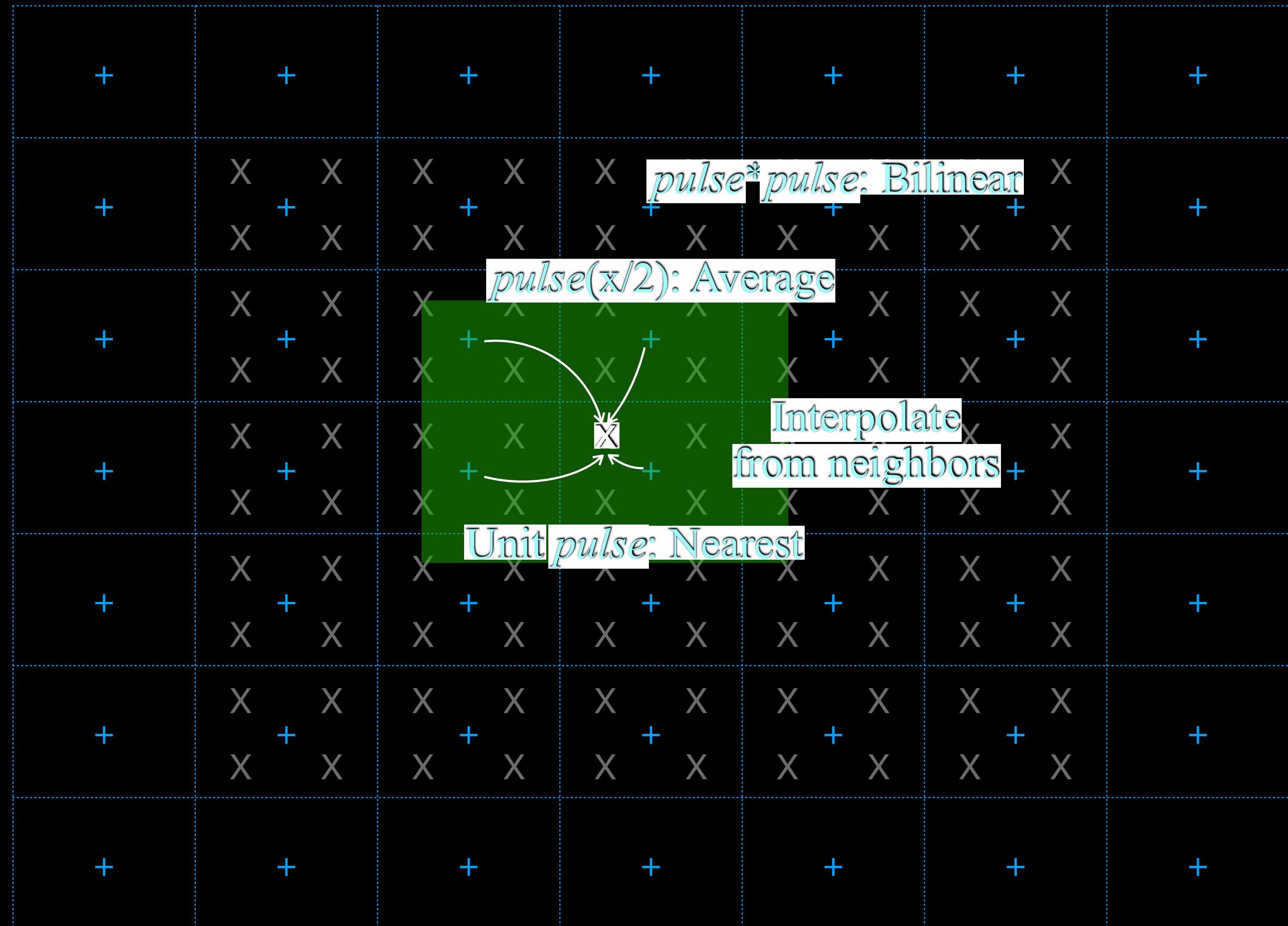
Scaling is Reconstruction



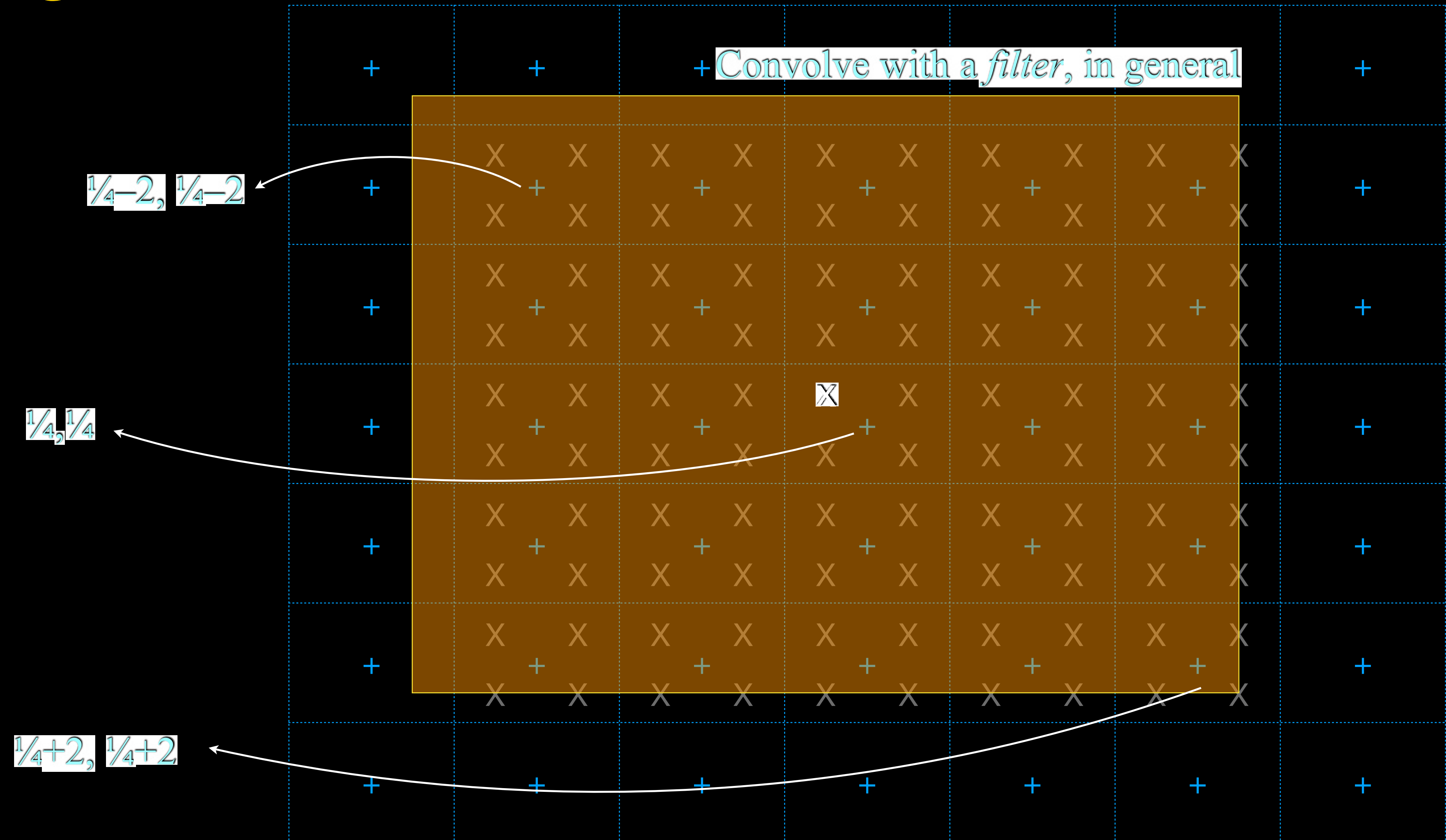
Scaling is Reconstruction



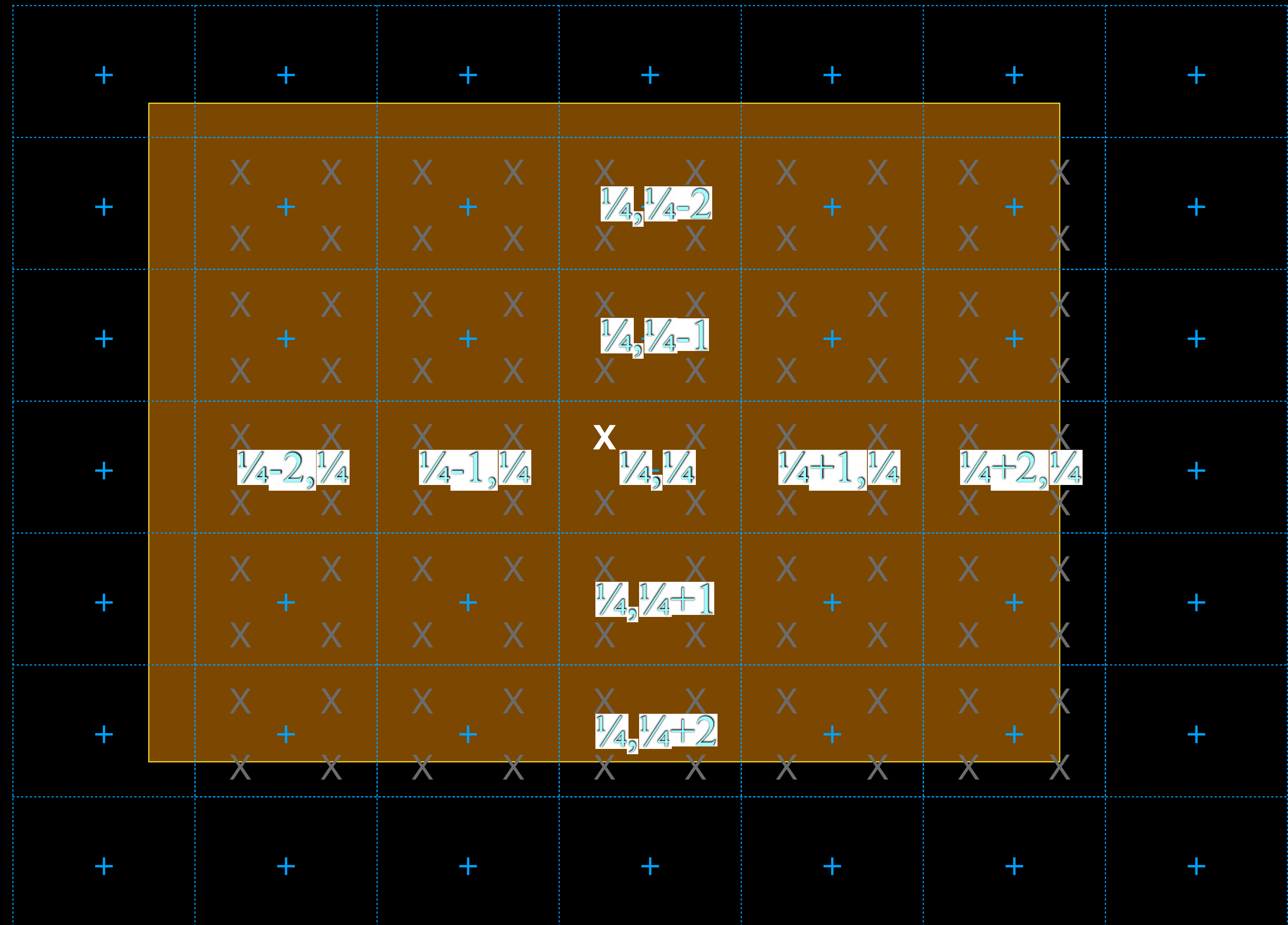
Scaling is Reconstruction



Scaling is Reconstruction



Scaling is Reconstruction



Can Transform Similarly

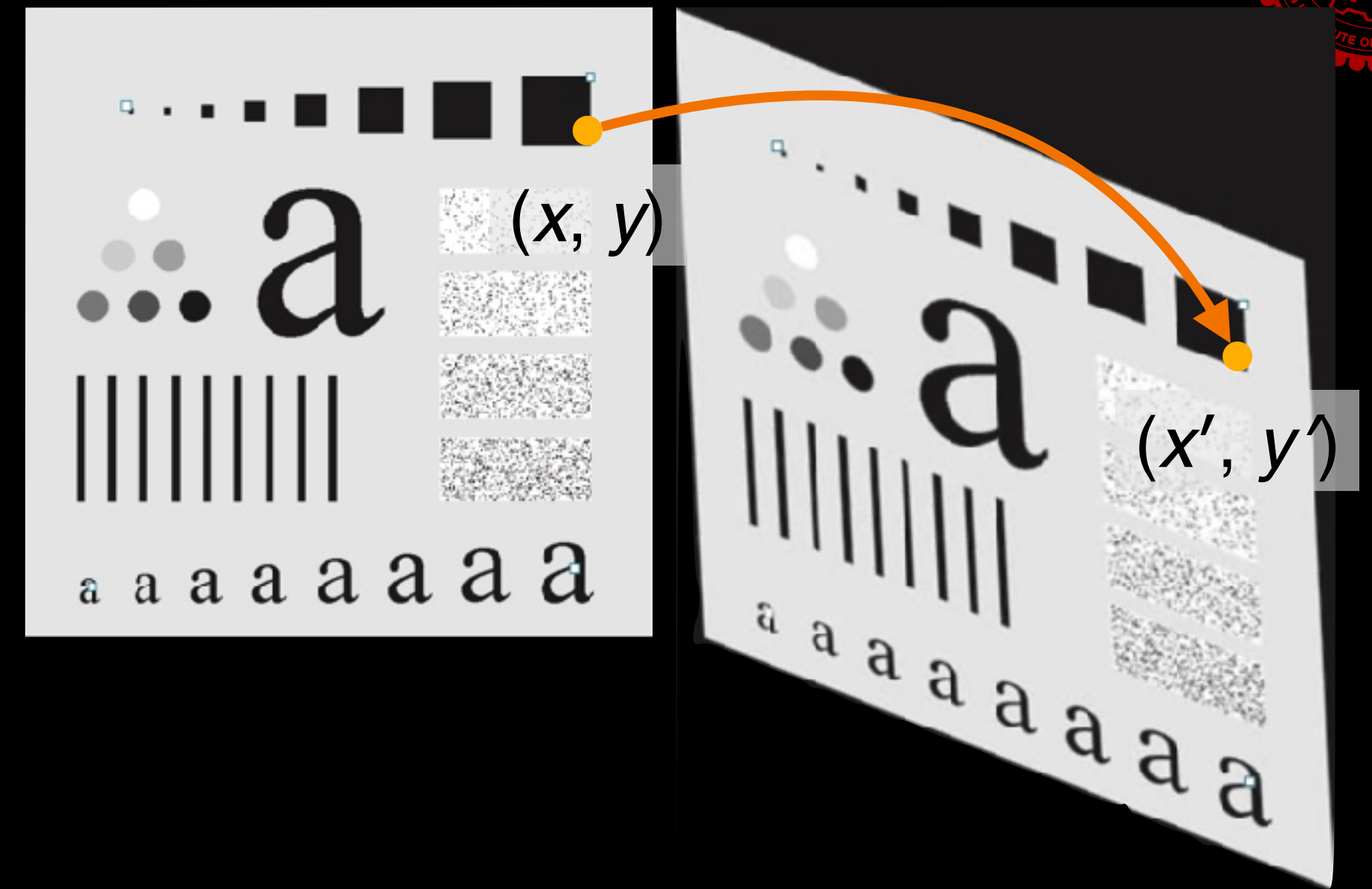
$$T(x, y) \rightarrow (x', y')$$

- Cannot iterate over and transform source pixels
 - All output pixels must exist
- Easier to iterate over destination, (x', y')
 - But require inverse transform

$$(x, y) = T^{-1}(x', y')$$

Not necessarily integers

Not necessarily within the extents of the original image





How to describe T?

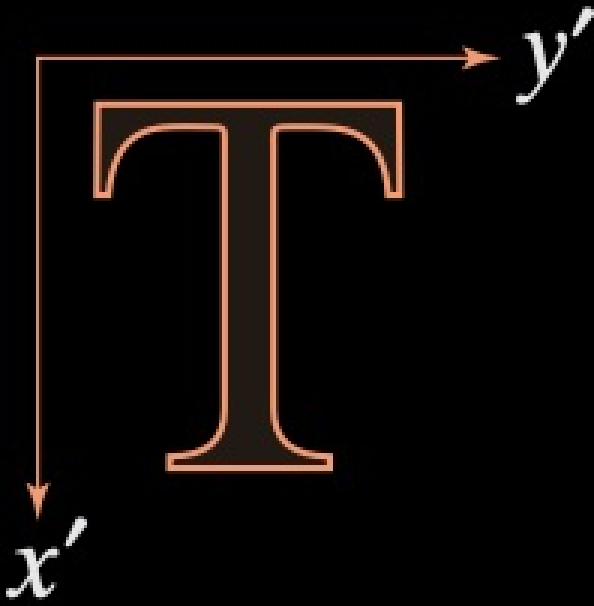
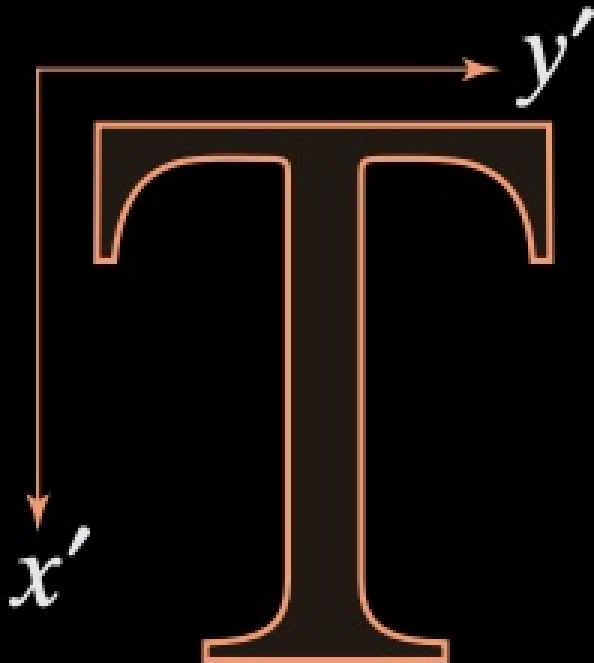
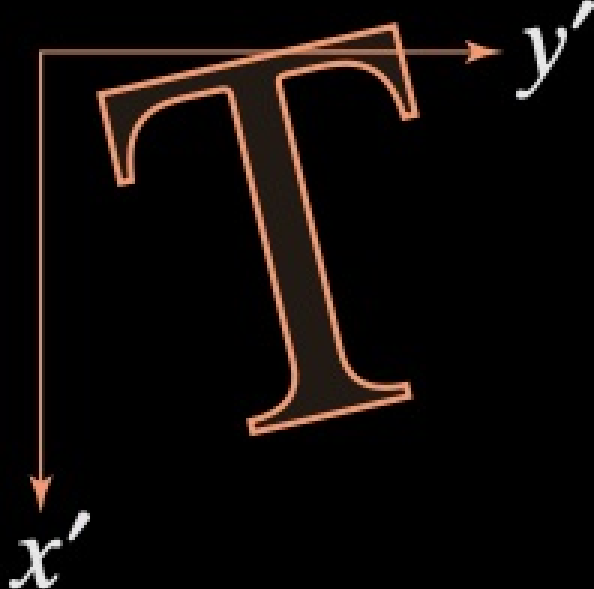
- Many transformations of interest are *affine*
 - Translation, rotation, scaling, shearing, etc.
- Can be represented as matrix-vector operations:

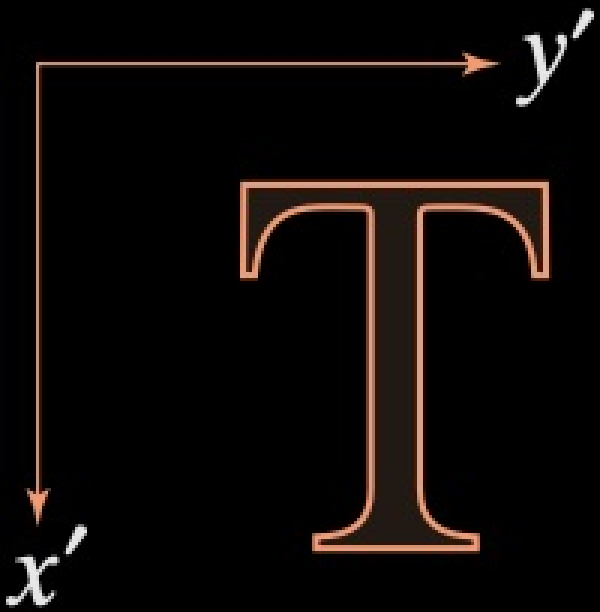
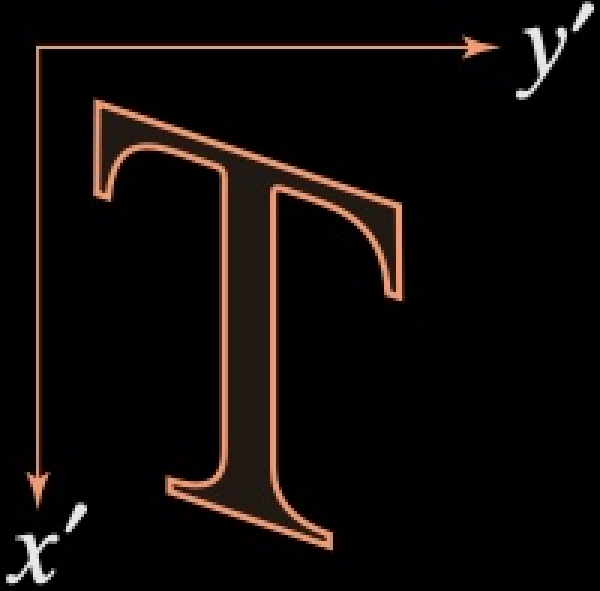
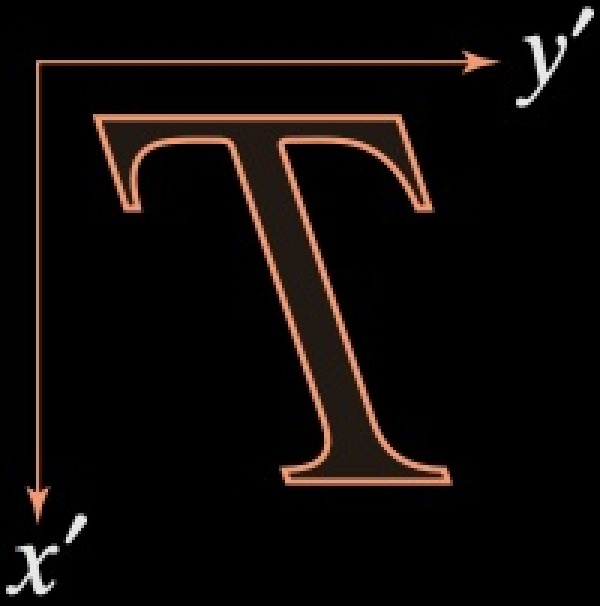
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

- or, introduce homogeneous coordinates:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Add a third coordinate = 1 (for now)

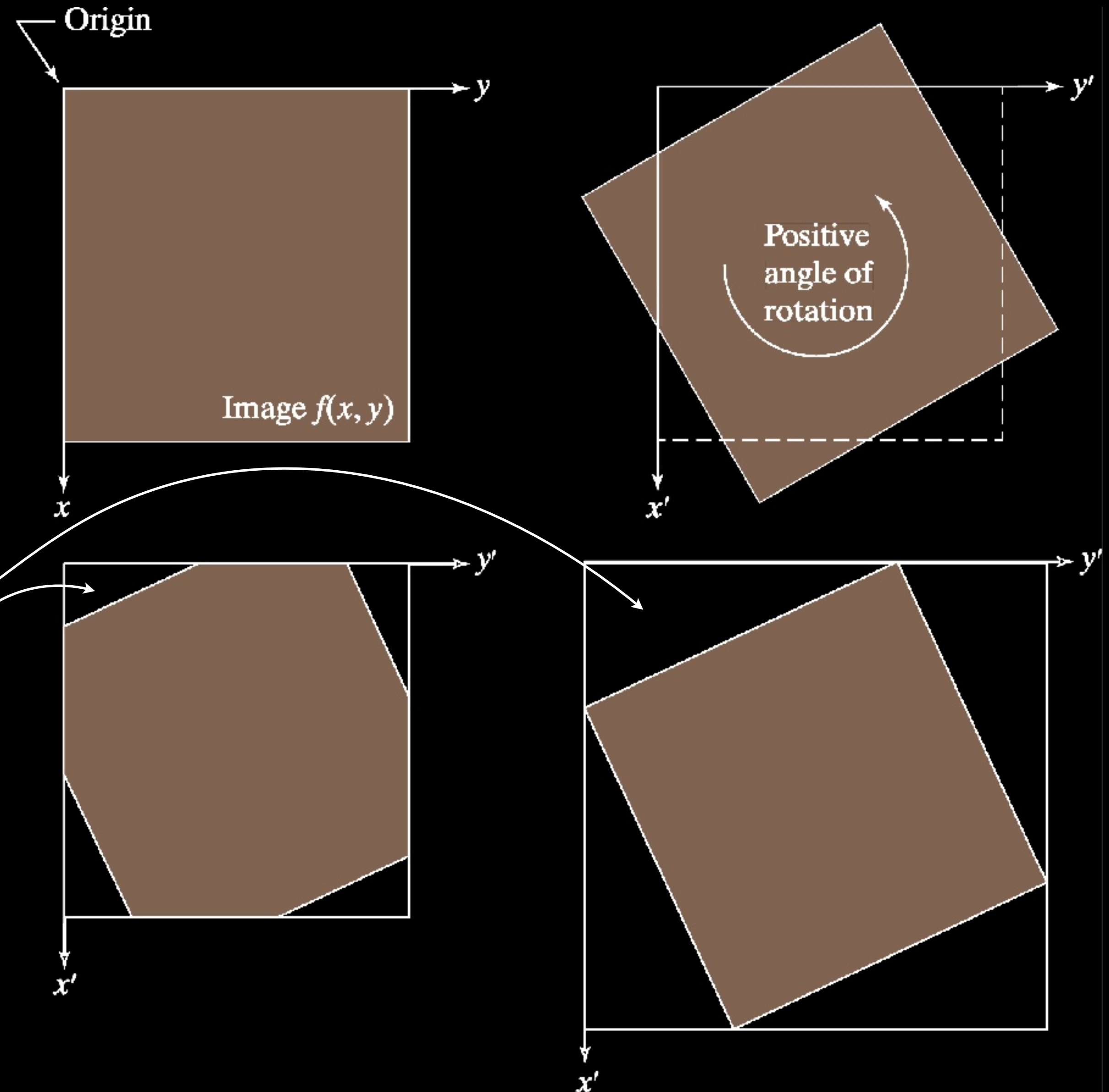
Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x' &= x \\ y' &= y \end{aligned}$	
Scaling/Reflection (For reflection, set one scaling factor to -1 and the other to 1)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x' &= c_x x \\ y' &= c_y y \end{aligned}$	
Rotation (about the origin)	$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{aligned} x' &= x \cos \theta - y \sin \theta \\ y' &= x \sin \theta + y \cos \theta \end{aligned}$	

Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + s_v y$ $y' = y$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

Output Size?

- Keep original size or expand to fit

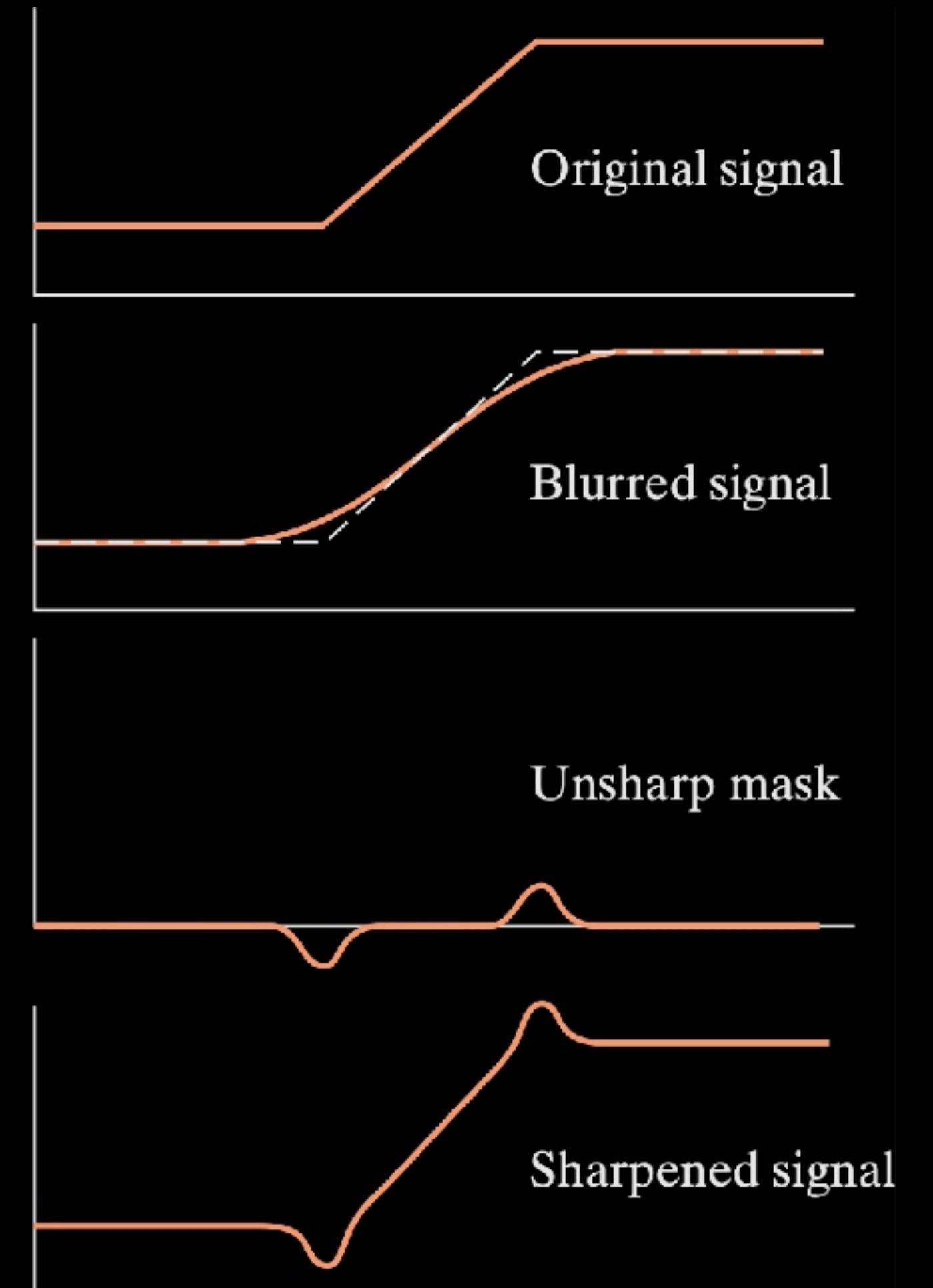
What goes here?



Exploiting Information in Blur

Unsharp Masking

- $c_b = \text{Blur}(c)$
 - “Unsharp”
- $g = c - c_b$
 - “mask”
- $c' = c + kg, k > 0$
 - Modulate the difference

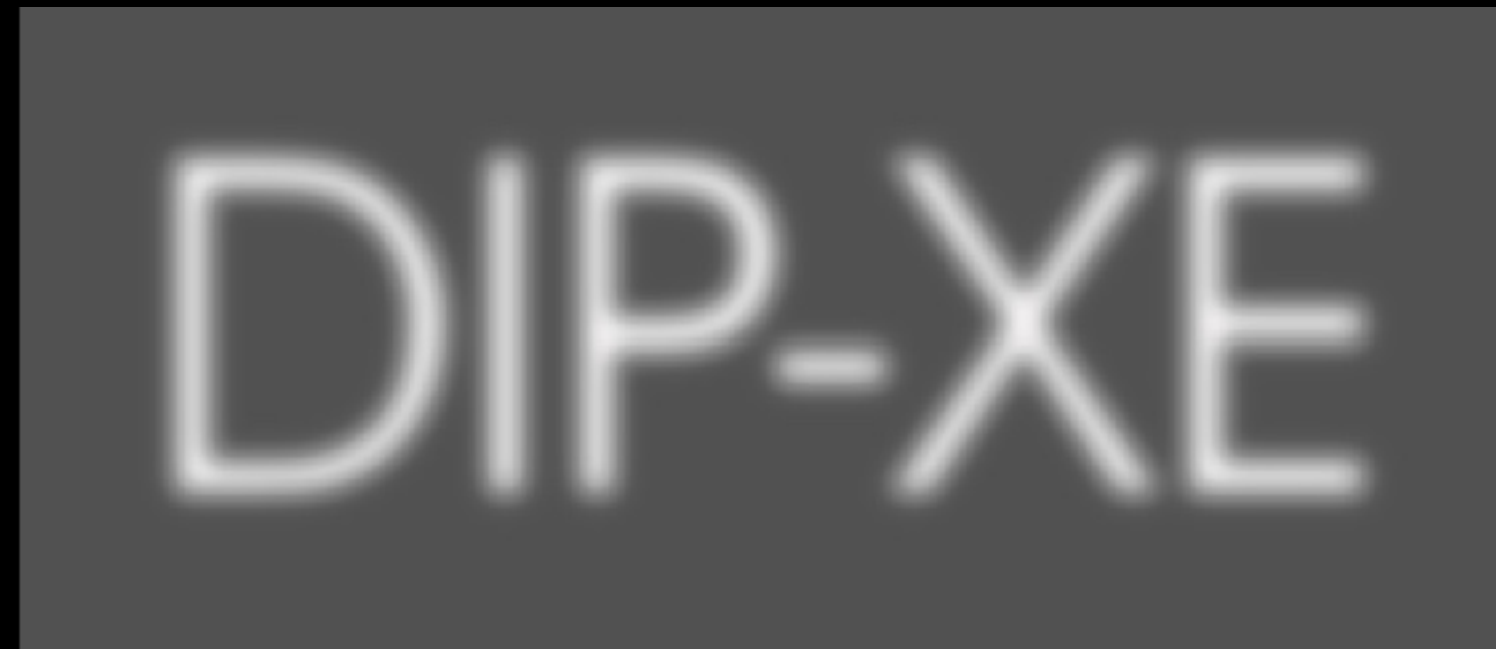


Unsharp Masking

Original image

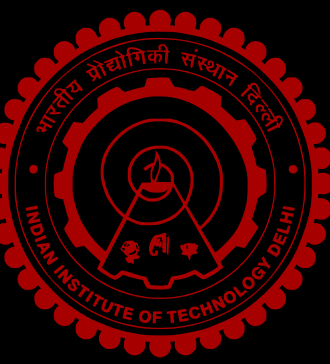
Smoothed image

Mask (+128)



Unsharp masking ($k = 1$)

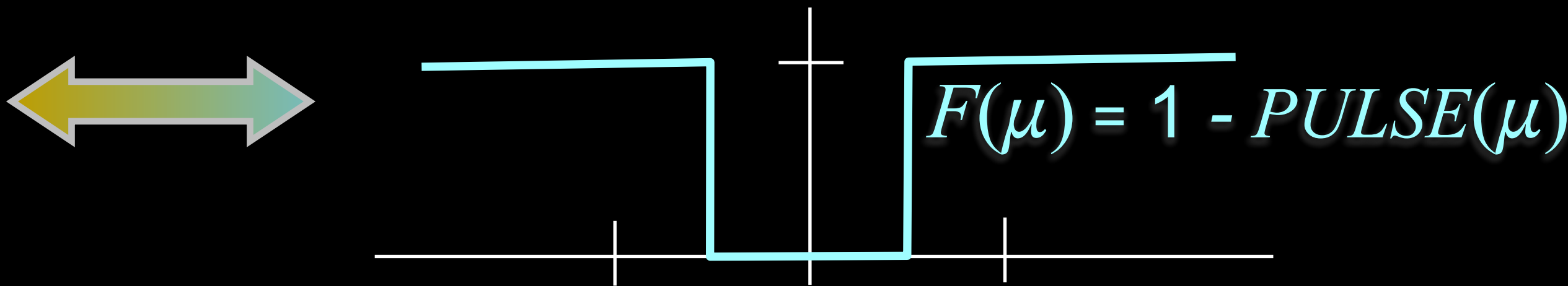
High-boost filtering ($k > 1$)



Sharpen, Really

Remove low frequency

- Impulse - Sinc(x)
 - Use linearity of (inverse) Fourier
- In general, HPF = Impulse - LPF



0	0	0	0	0
0	0	0	0	0
0	0	2	0	0
0	0	0	0	0
0	0	0	0	0

=

LPF				
0.0030	0.0133	0.0219	0.0133	0.0030
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0030	0.0133	0.0219	0.0133	0.0030

Total Weight remains 1

More control:

f_{sharp}

Identity + $k \cdot \text{HPF}$