Visual Computing:
Fourier Transform

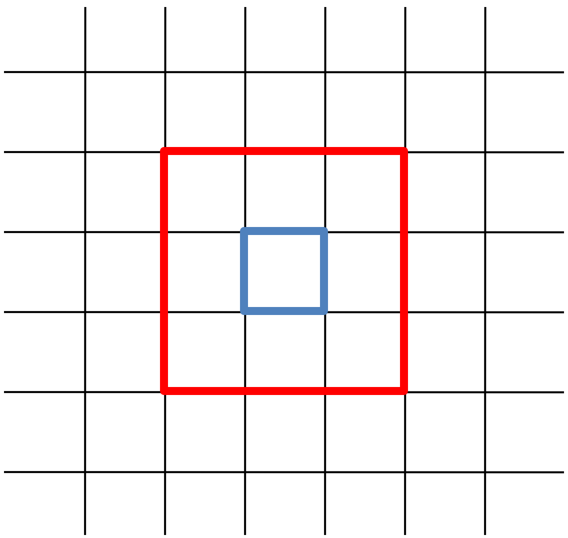
Prof. Marc Pollefeys

Video

Last week

Correlation

(e.g. Template-matching)

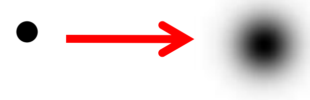


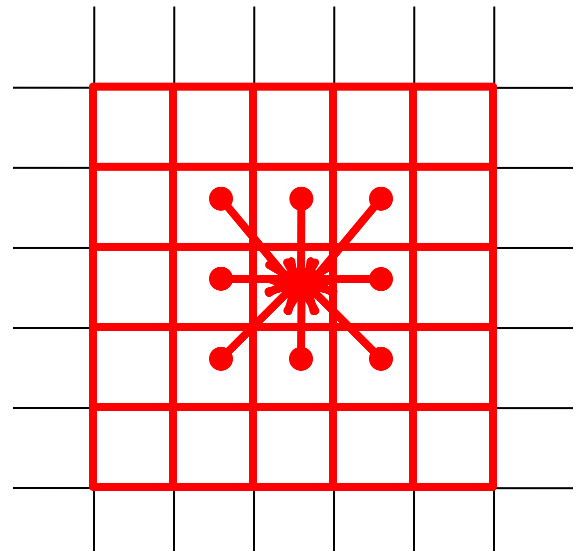
$$I' = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(i,j) I(x+i, y+j)$$

(matlab default)

Convolution

(e.g. point spread function)





$$I' = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(i,j) I(x-i, y-j)$$

Last lecture

- Edges
  - Maximal gradients
  - Zero-crossing Laplacian

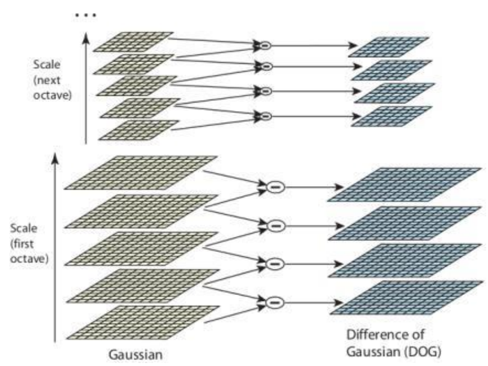
Canny edge detection

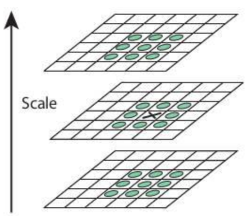
- Hough transform
- Corners
  - Maximal difference with neighbors

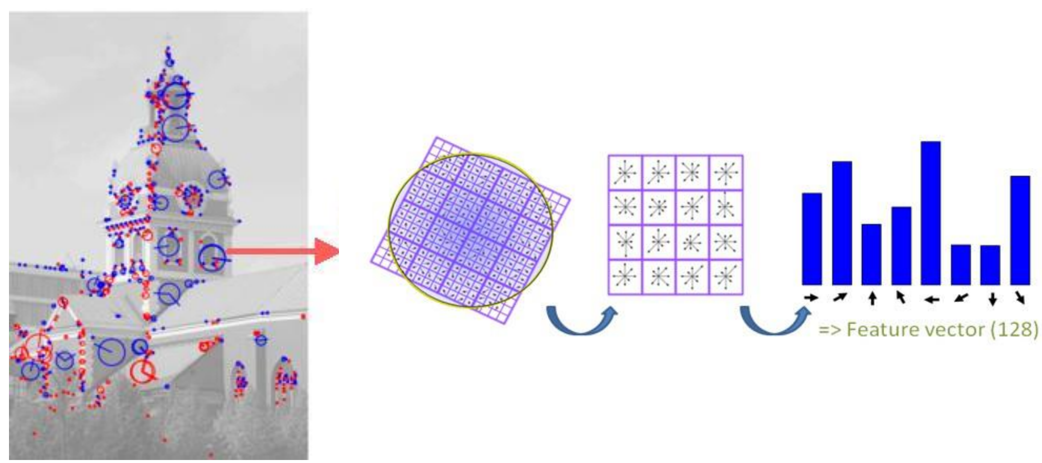
$$\mathbf{M} = \begin{bmatrix} \sum_{(x,y)\in window} f_x^2(x,y) & \sum_{(x,y)\in window} f_x(x,y)\, f_y(x,y) \\ \sum_{(x,y)\in window} f_x(x,y)\, f_y(x,y) & \sum_{(x,y)\in window} f_y^2(x,y) \end{bmatrix}$$

Scale Invariant Feature Transform (SIFT)

Lowe ICCV99







Visual Computing:
Fourier Transform

Prof. Marc Pollefeys

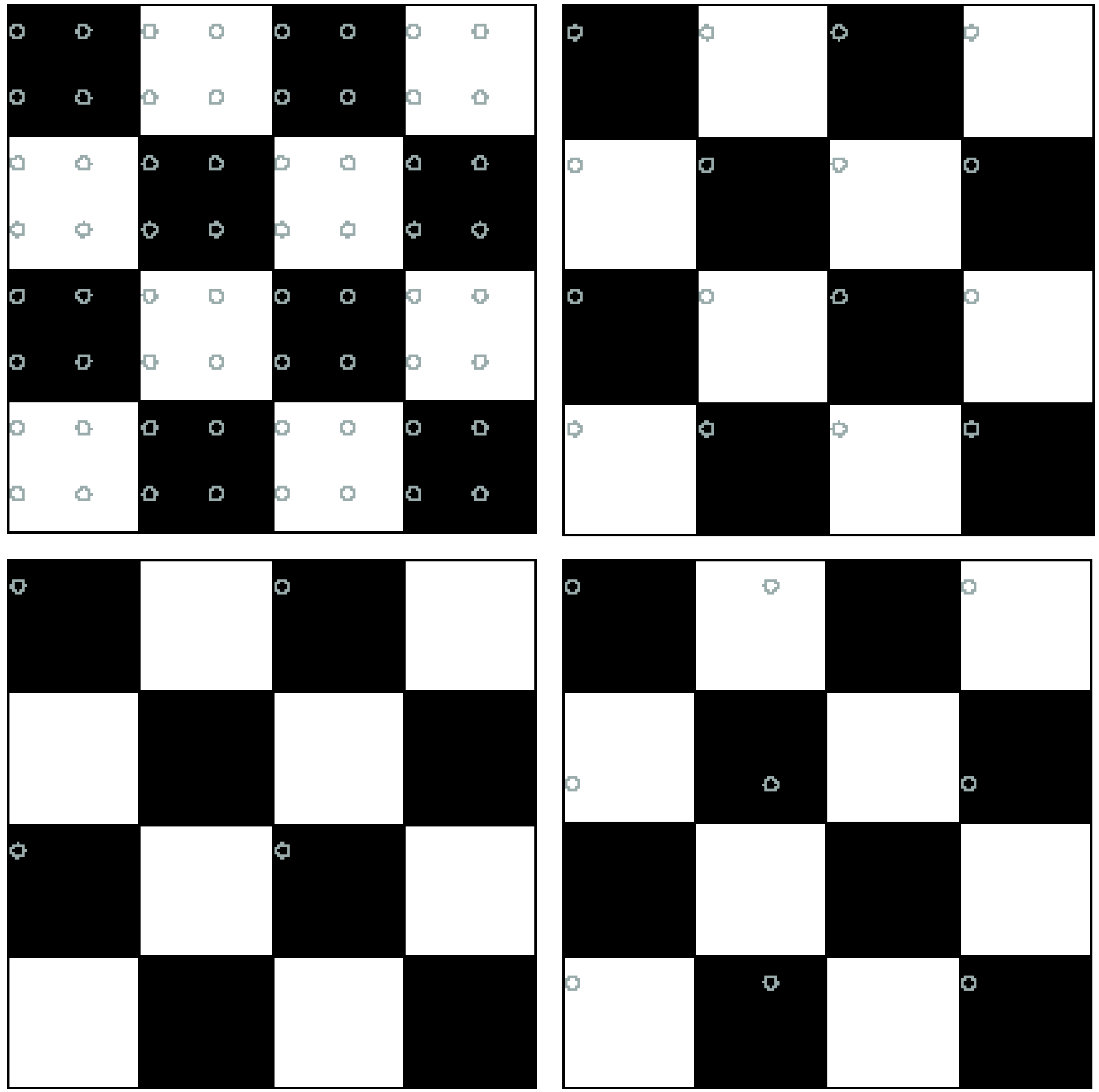
ETH

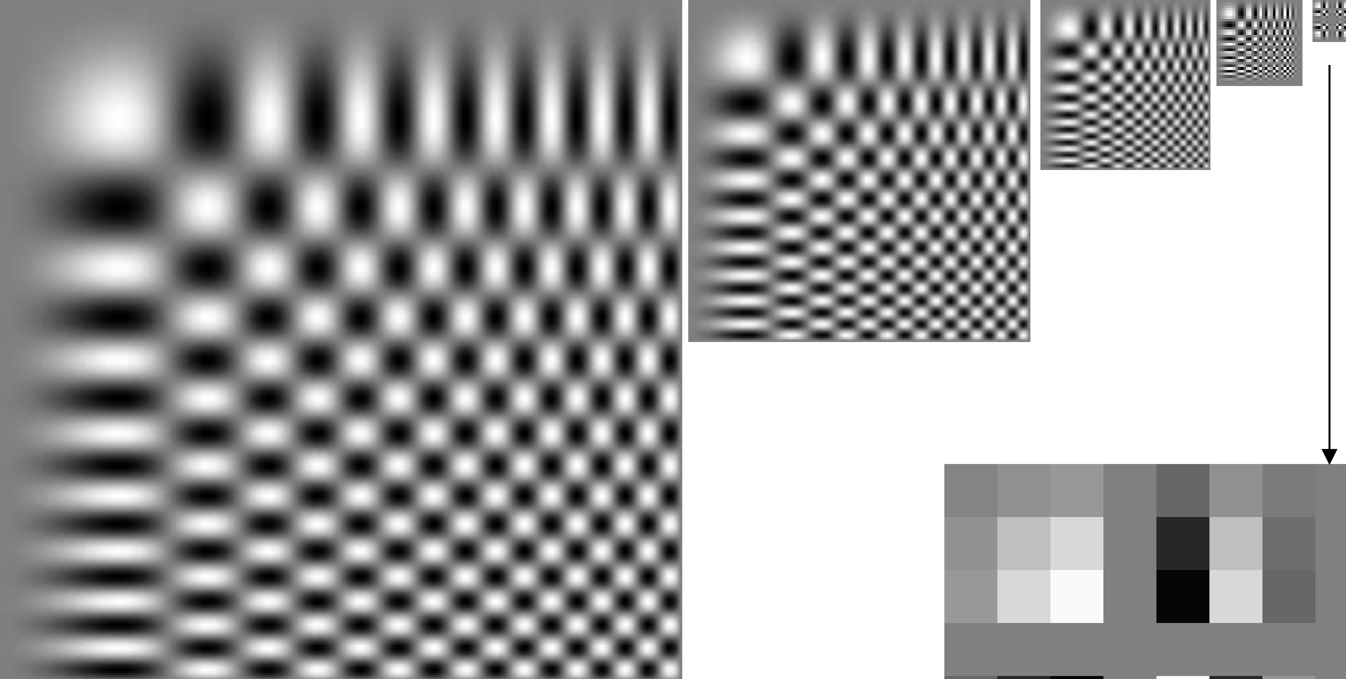
inf | Informatik
Computer Science

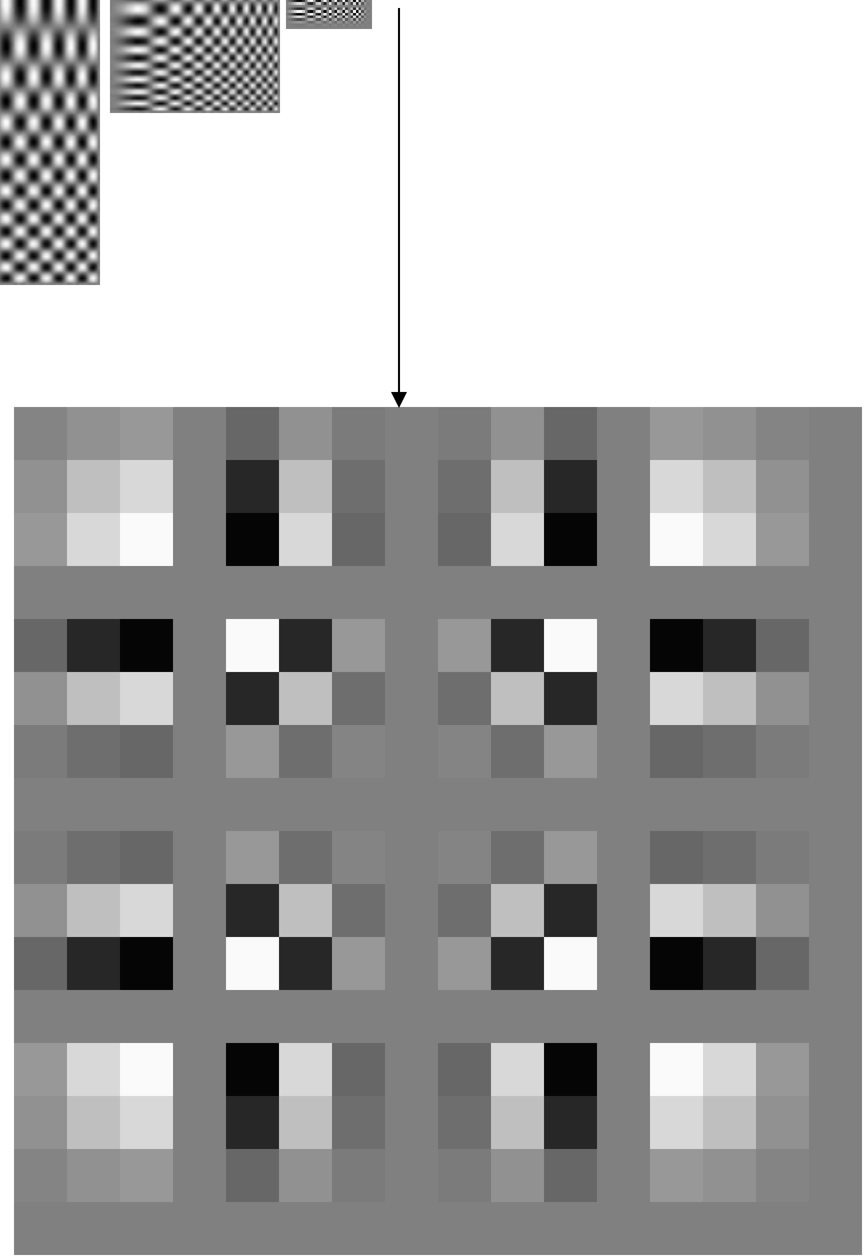
Aliasing

- Can't shrink an image by taking every second pixel
- If we do, characteristic errors appear
  - In the next few slides
  - Typically, small phenomena look bigger; fast phenomena can look slower
  - Common phenomenon
    - Wagon wheels rolling the wrong way in movies
    - Checkerboards misrepresented in ray tracing
    - Striped shirts look funny on color television

example image
example video







Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer

Open questions

- What causes the tendency of differentiation to emphasize noise?
- In what precise respects are discrete images different from continuous images?
- How do we avoid aliasing?

- General thread: a language for fast changes

The Fourier Transform

The Fourier Transform

- Represent function on a new basis
  - Think of functions as vectors, with many components
  - We now apply a linear transformation to transform the basis
    - dot product with each basis element
- In the expression, u and v select the basis element, so a function of x and y becomes a function of u and v
- basis elements have the form $e^{-i2\pi(ux+vy)}$

$= \cos 2\pi(ux+vy) - i \sin 2\pi(ux+vy)$

$$\boxed{F\big(g(x,y)\big)(u,v) = \iint_{\mathbb{R}^2} g(x,y)\, e^{-i2\pi(ux+vy)}\, dx\, dy}$$

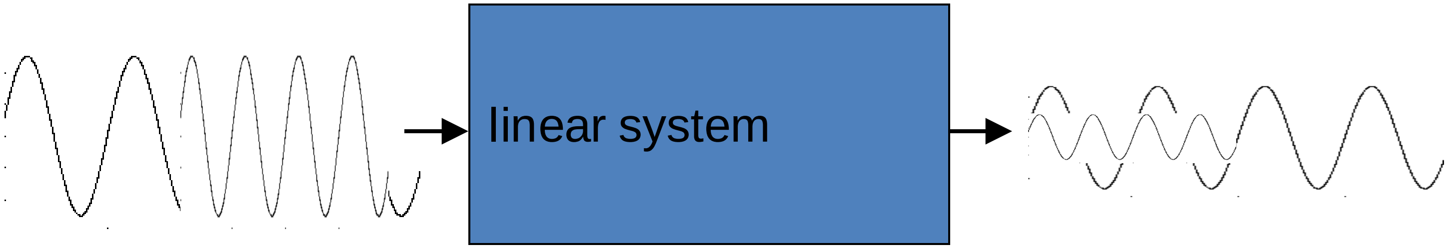
Discrete FT: transformed image $\boxed{F = \mathbf{U}f}$ vectorized image

Fourier transform base,
also possible Wavelets, steerable pyramids, etc.

Fourier transform and linear systems

- Basis functions of Fourier transform are eigenfunctions of linear systems

(or why Electrical Engineers love the Fourier transform)



Fourier basis element

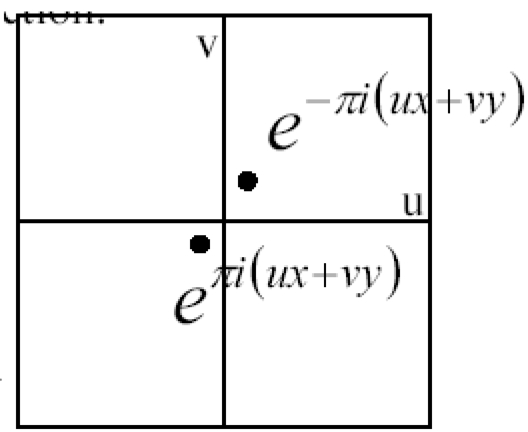
$$e^{-i2\pi(ux+vy)}$$

example, real part

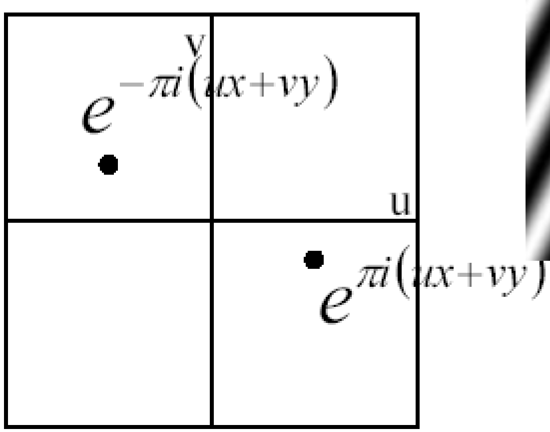
$F^{u,v}(x,y)$

$F^{u,v}(x,y)$=const. for
$(ux+vy)$=const.

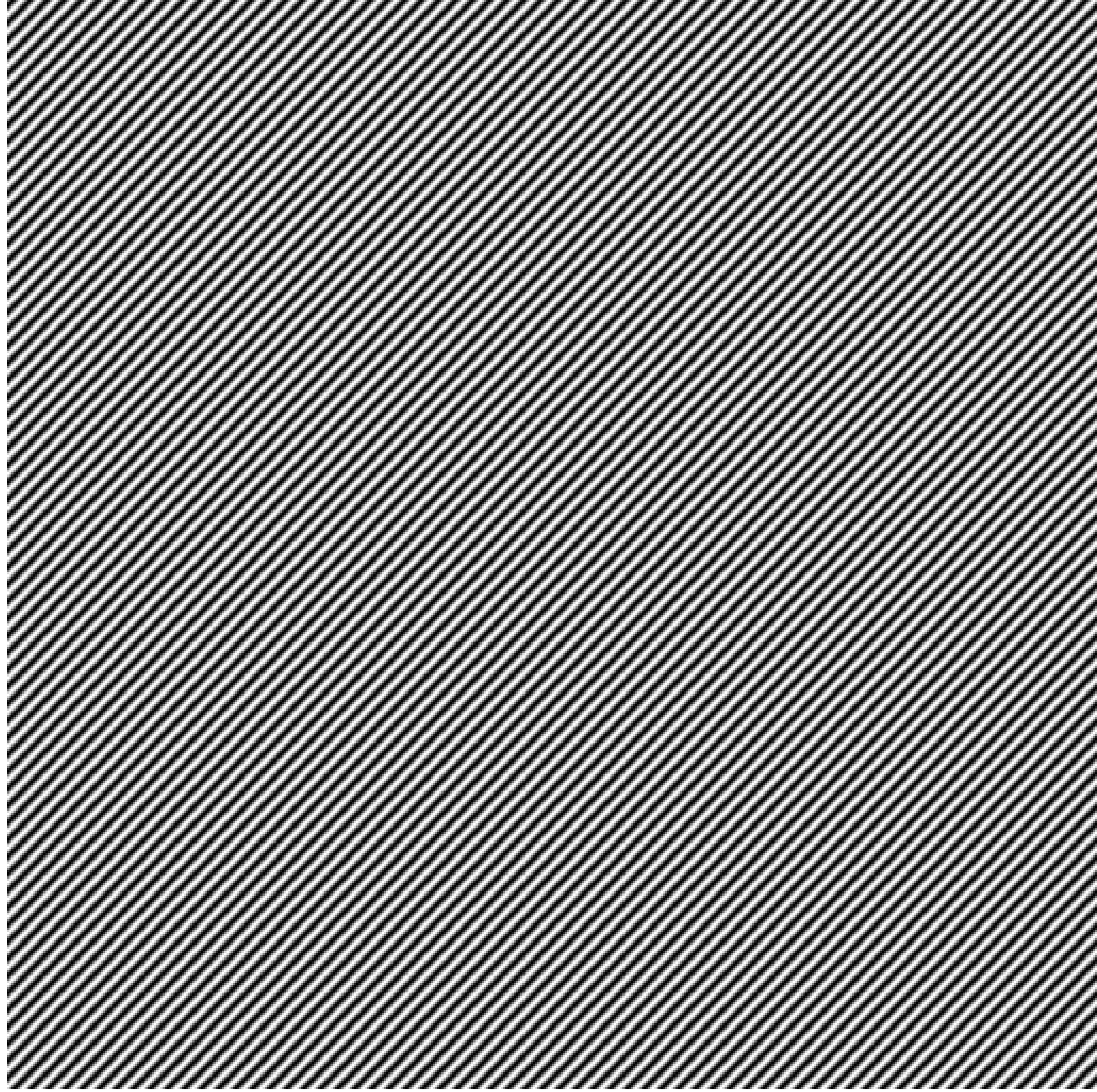
Vector (u,v)
• Magnitude gives frequency
• Direction gives orientation.

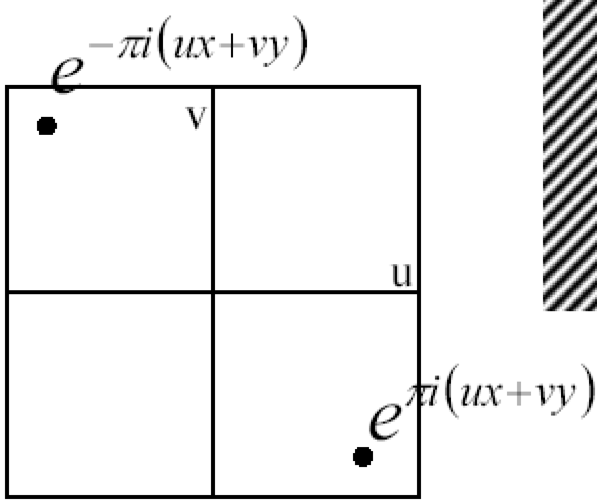


Here u and v are larger than in the previous slide.



And larger still...



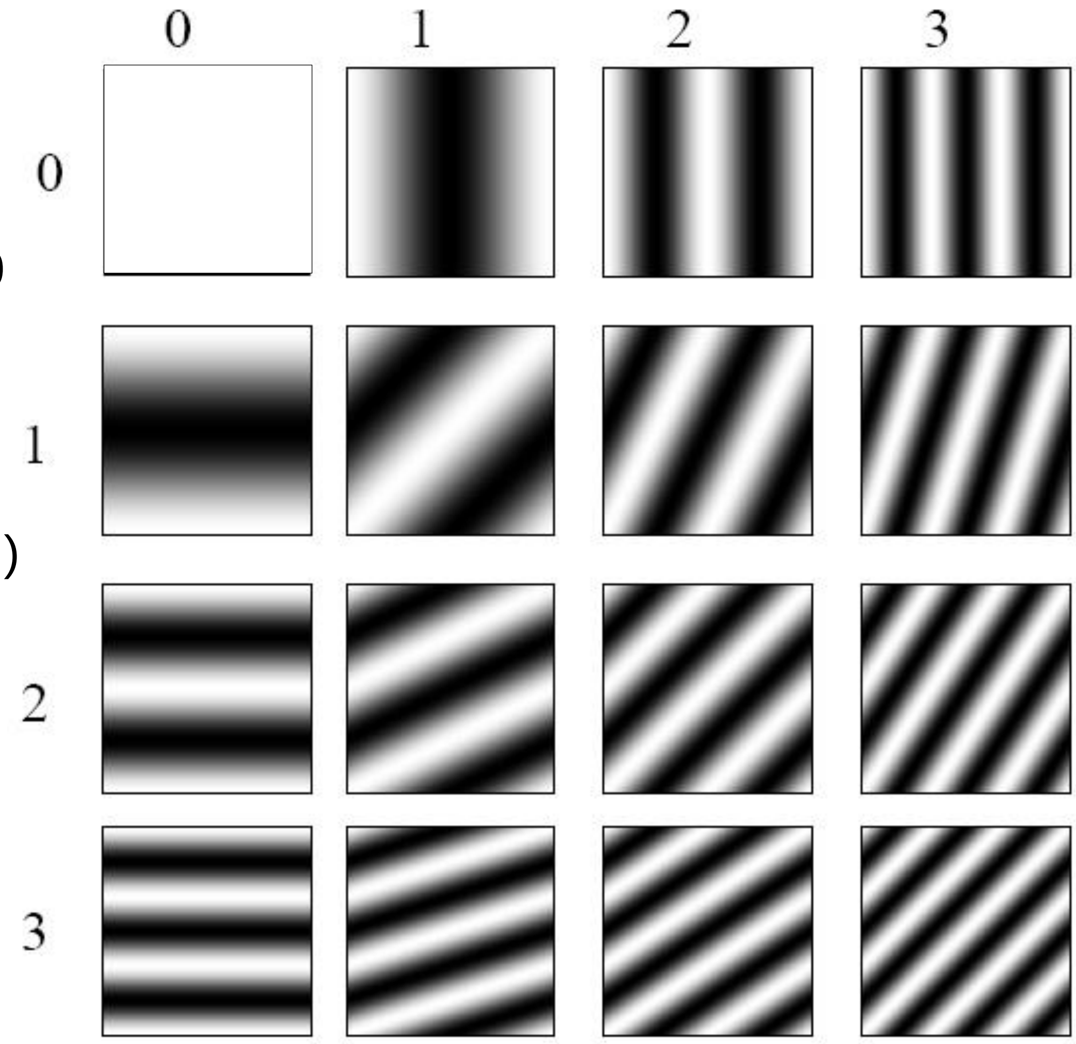


Fourier basis functions

$$e^{-i2\pi(ux+vy)}$$

$= \cos 2\pi(ux+vy) - i \sin 2\pi\ (ux+vy)$

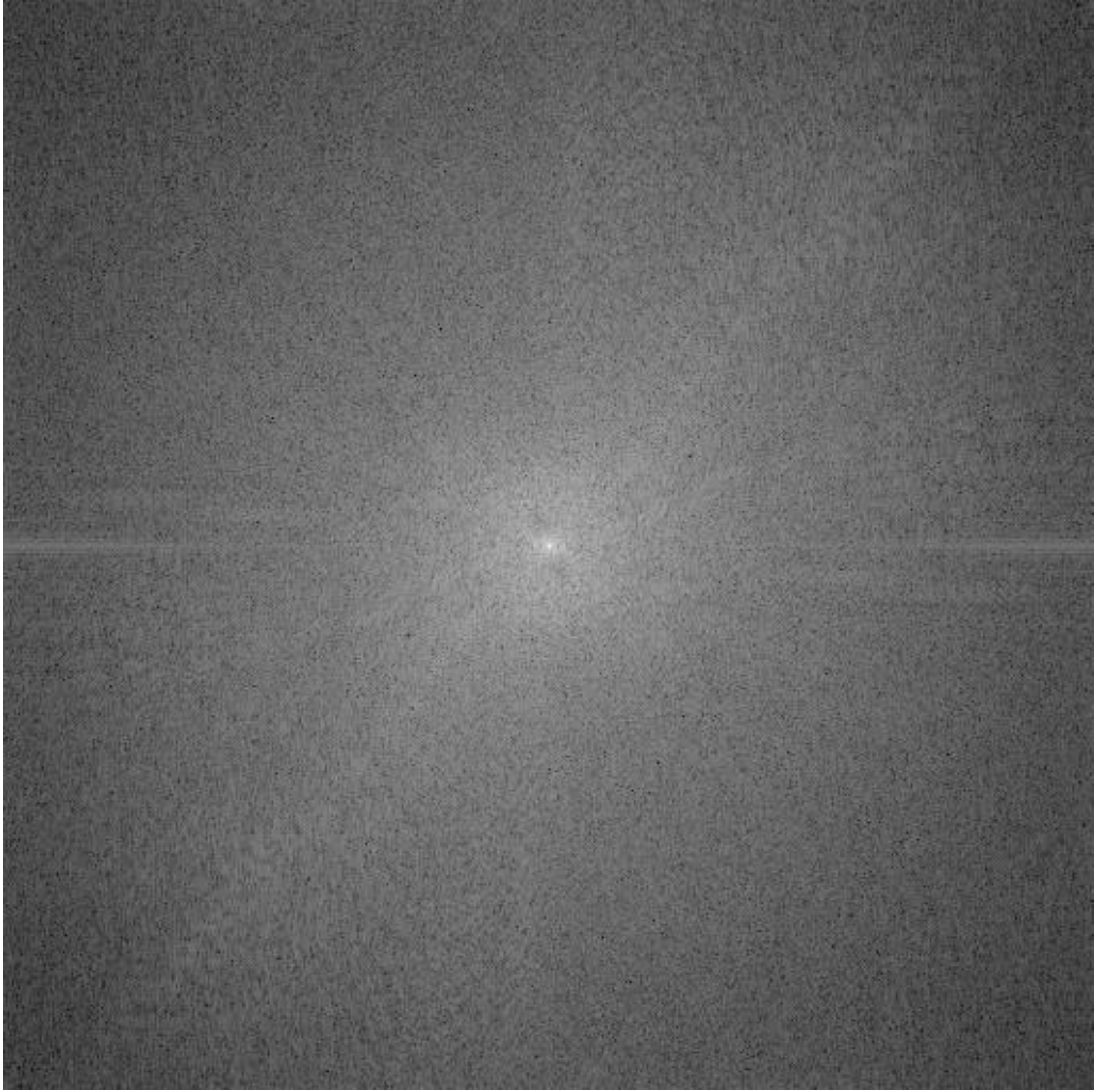
(note: basis functions are global)



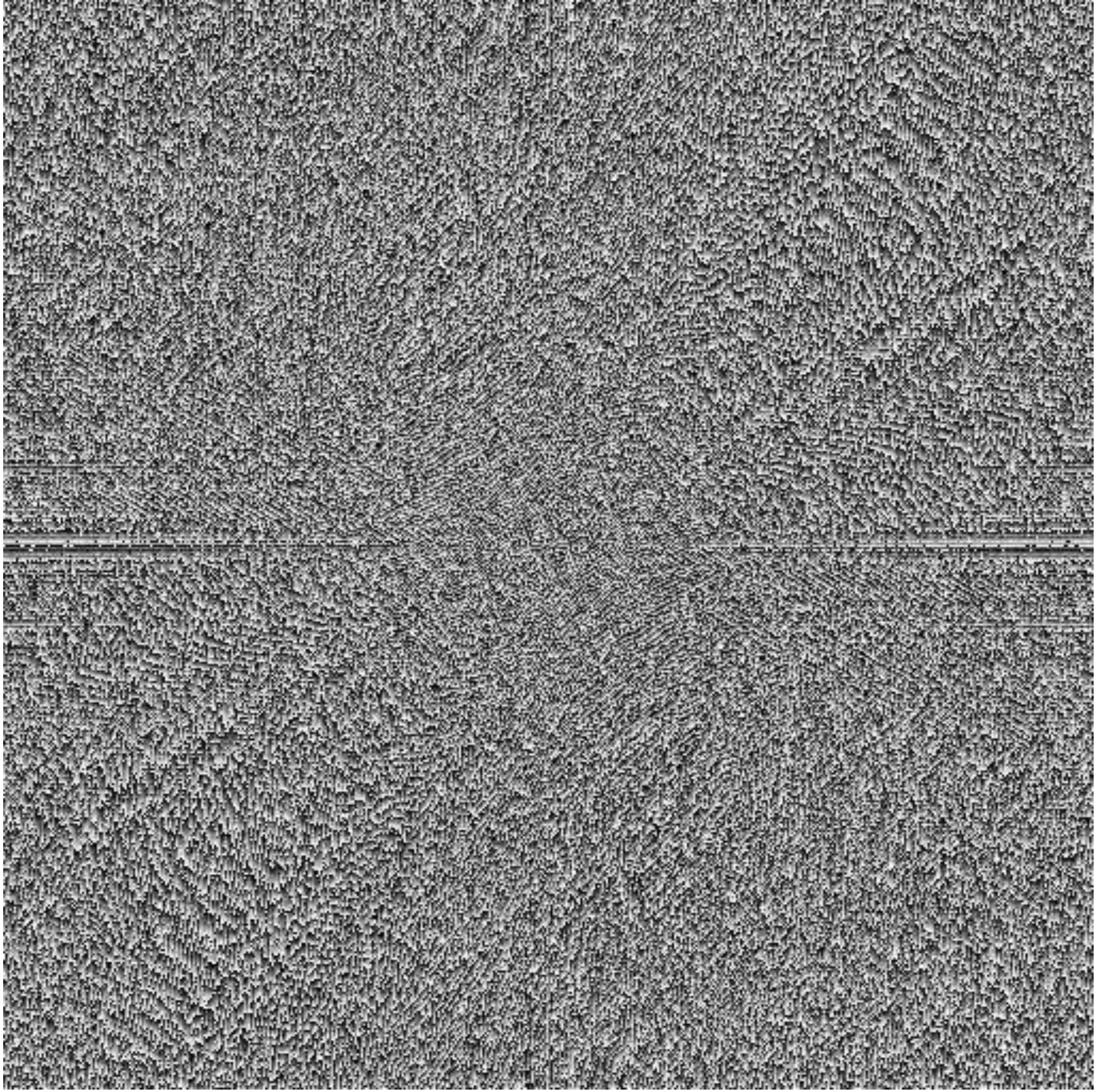
Phase and Magnitude

- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform
- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn't
- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?

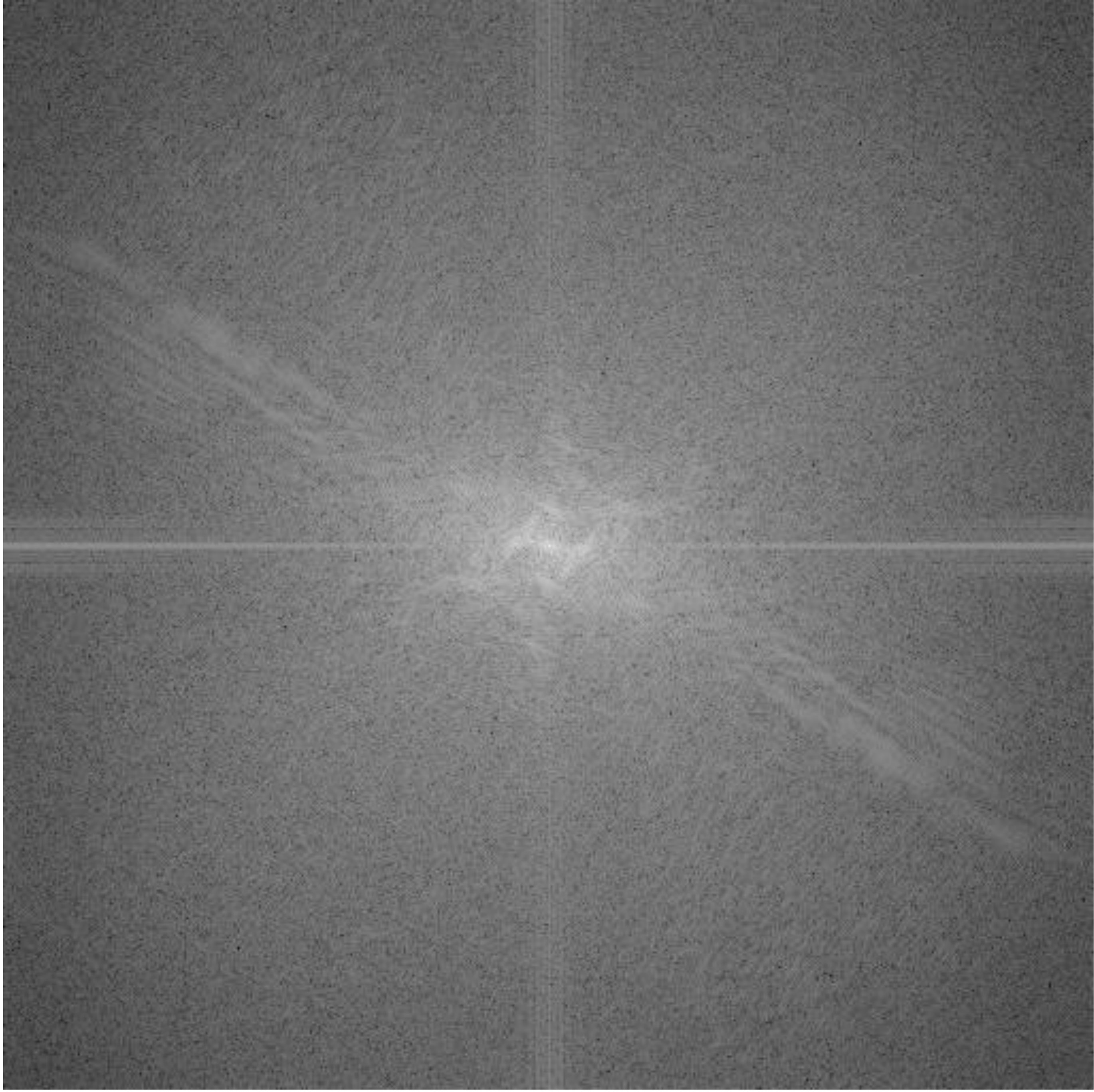
This is the magnitude transform of the cheetah pic



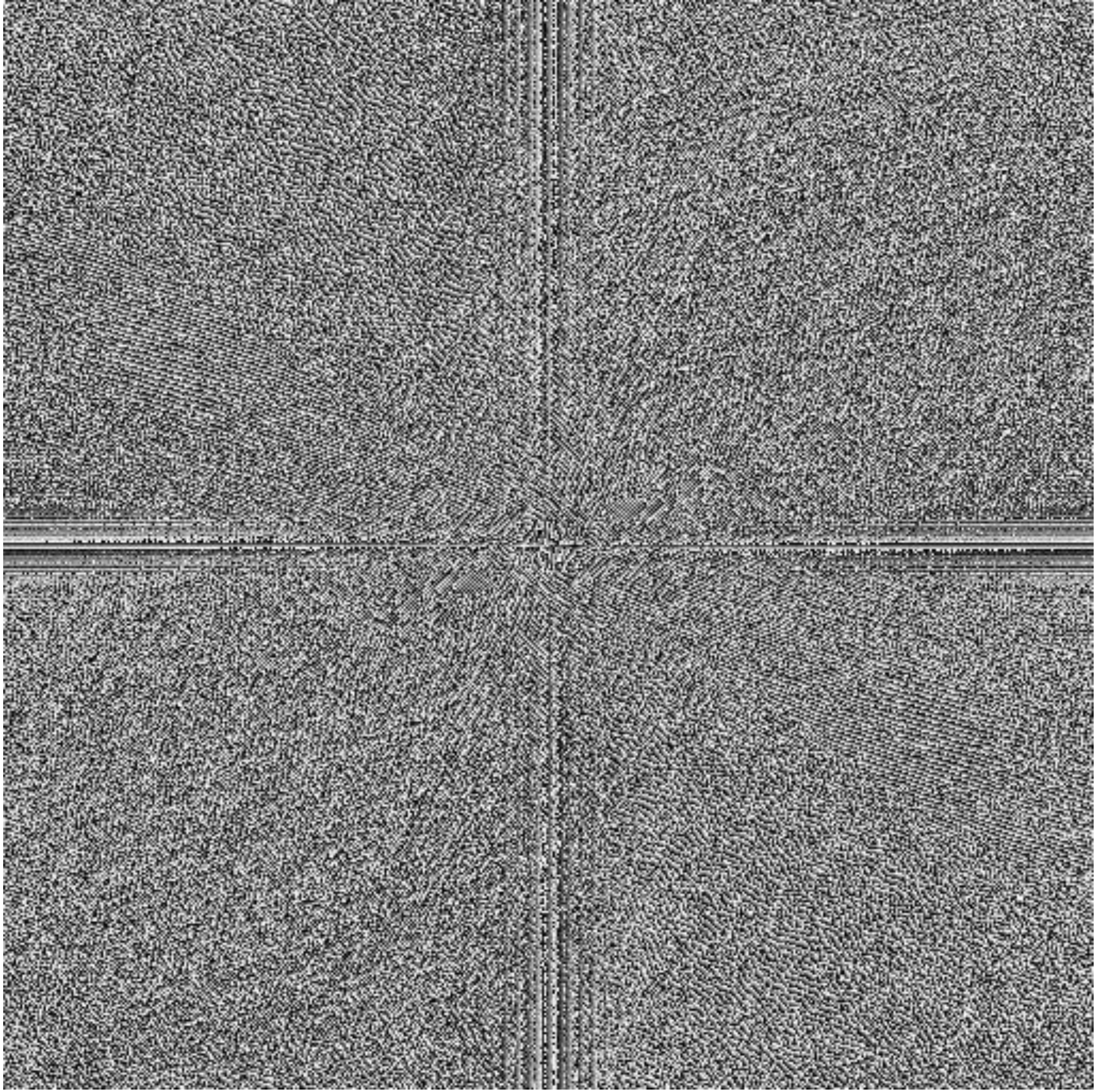
This is the phase transform of the cheetah pic



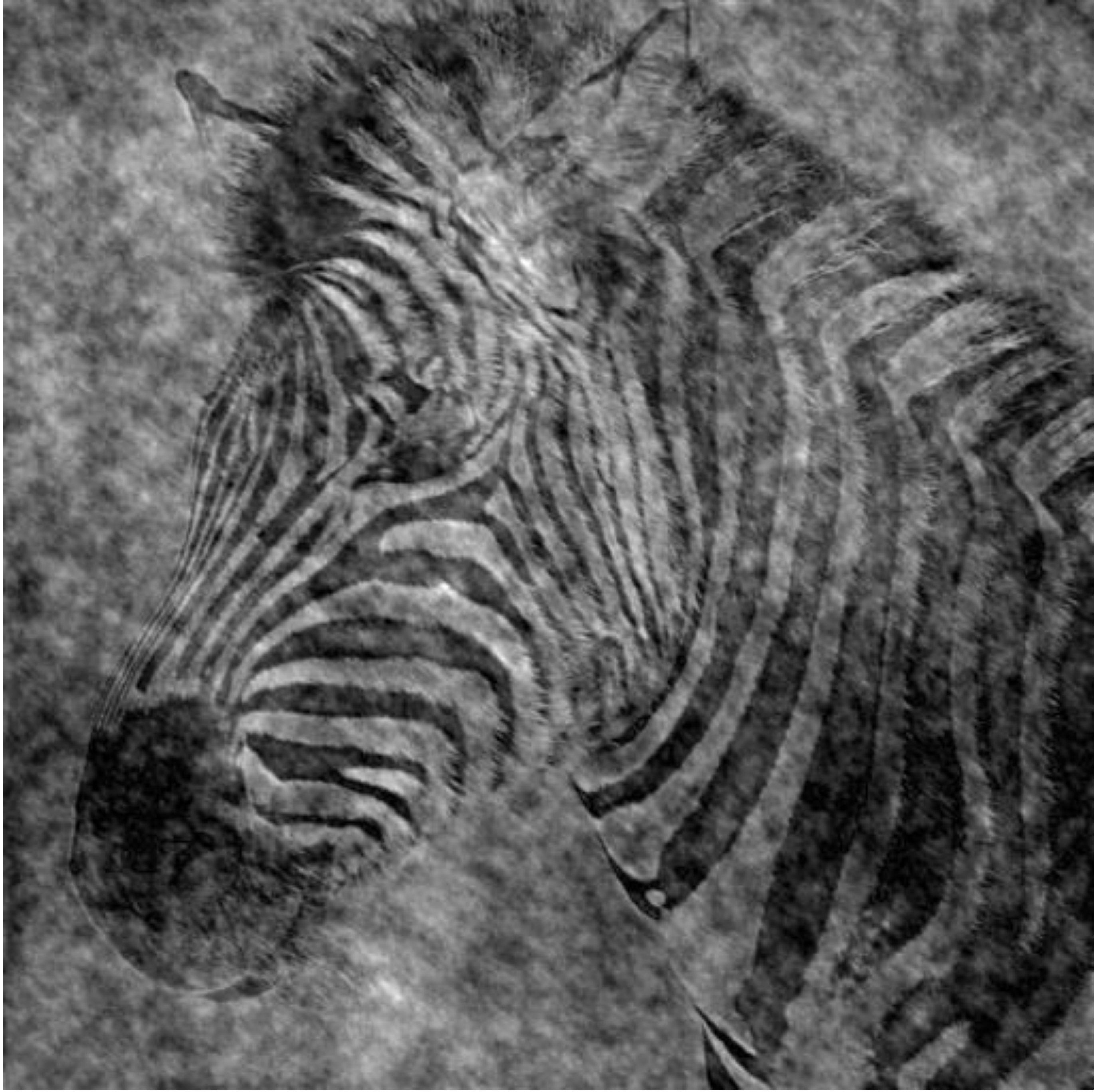
This is the magnitude transform of the zebra pic



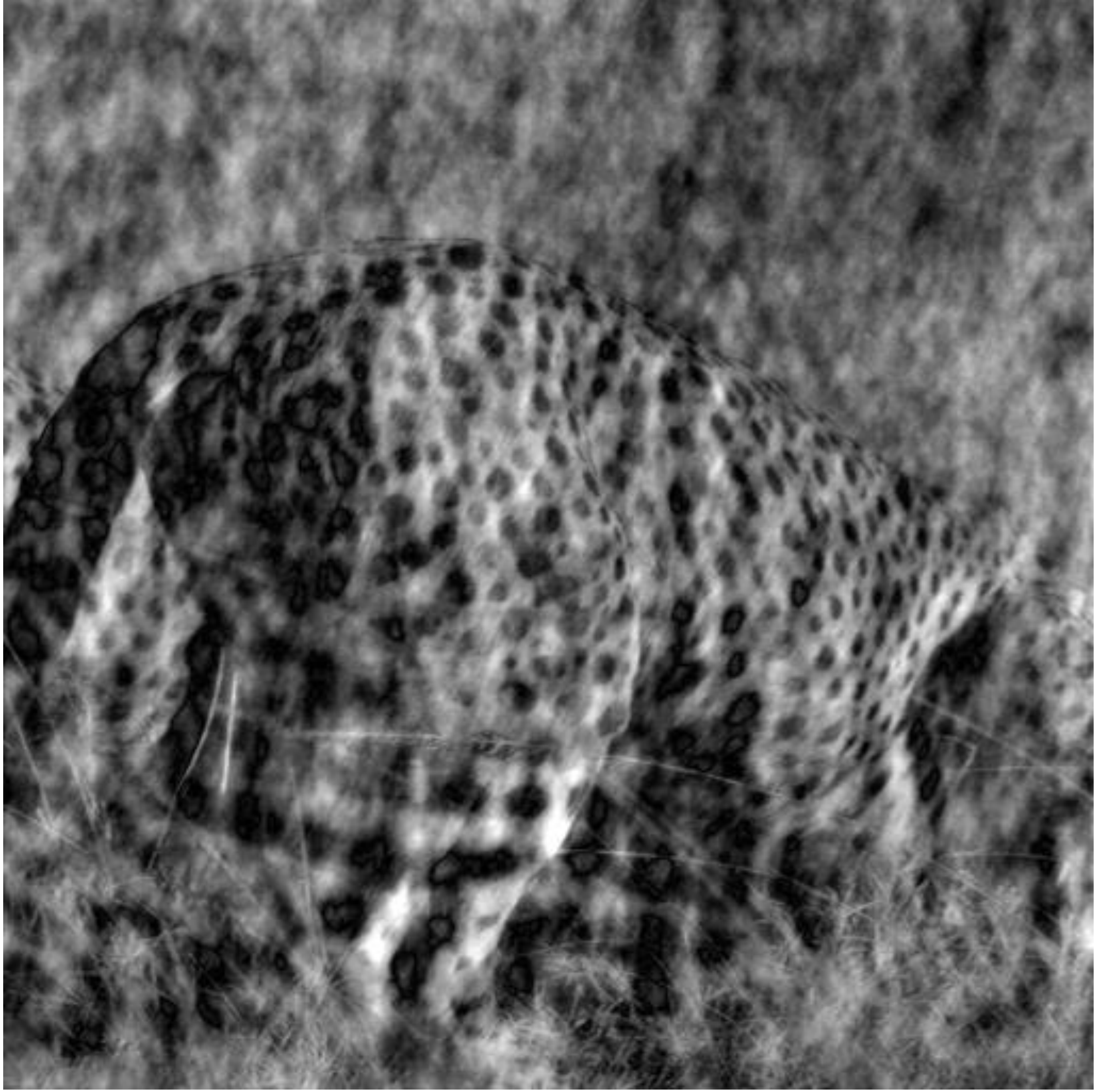
This is the phase transform of the zebra pic

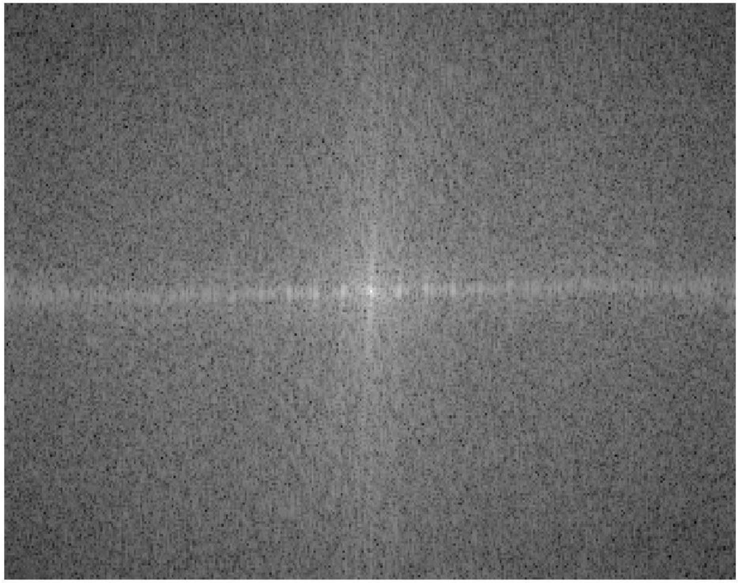


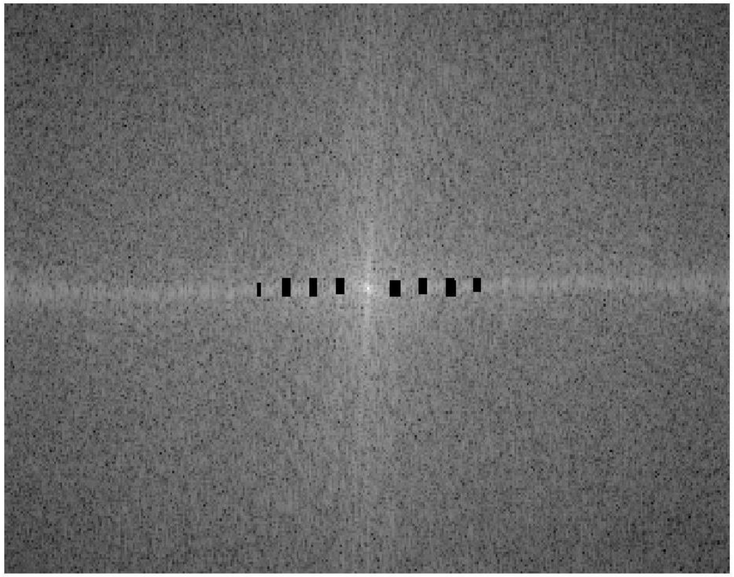
Reconstruction with zebra phase, cheetah magnitude



Reconstruction with cheetah phase, zebra magnitude

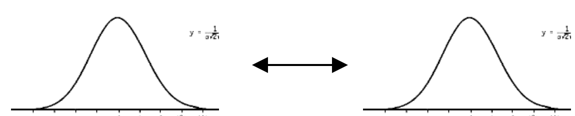




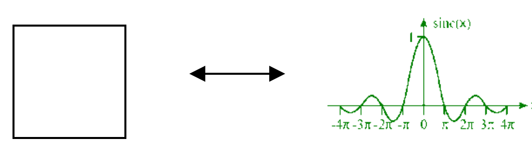


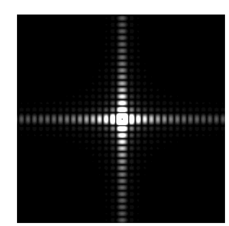
Various Fourier Transform Pairs

- Important facts
  - The Fourier transform is linear
  - There is an inverse FT $$f = \mathbf{U}^{-1}F$$
  - scale function down $\Leftrightarrow$ scale transform up
    i.e. high frequency = small details
  - The FT of a Gaussian is a Gaussian.



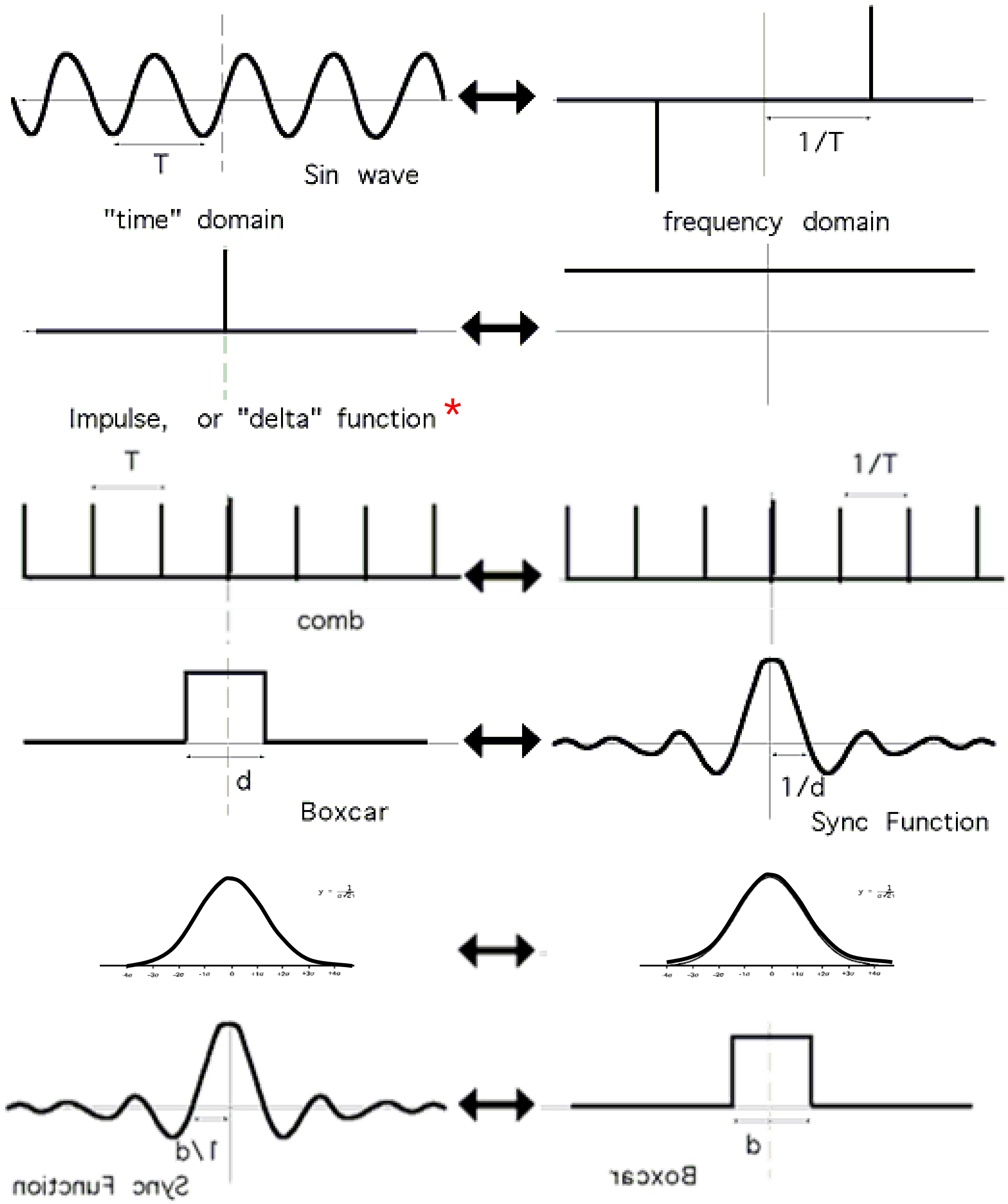
compare to box function transform





Fourier Transform of important functions

spatial domain | frequency domain



* see in a few slides

Convolution theorem

- The convolution theorem
  - The Fourier transform of the convolution of two functions is the product of their Fourier transforms

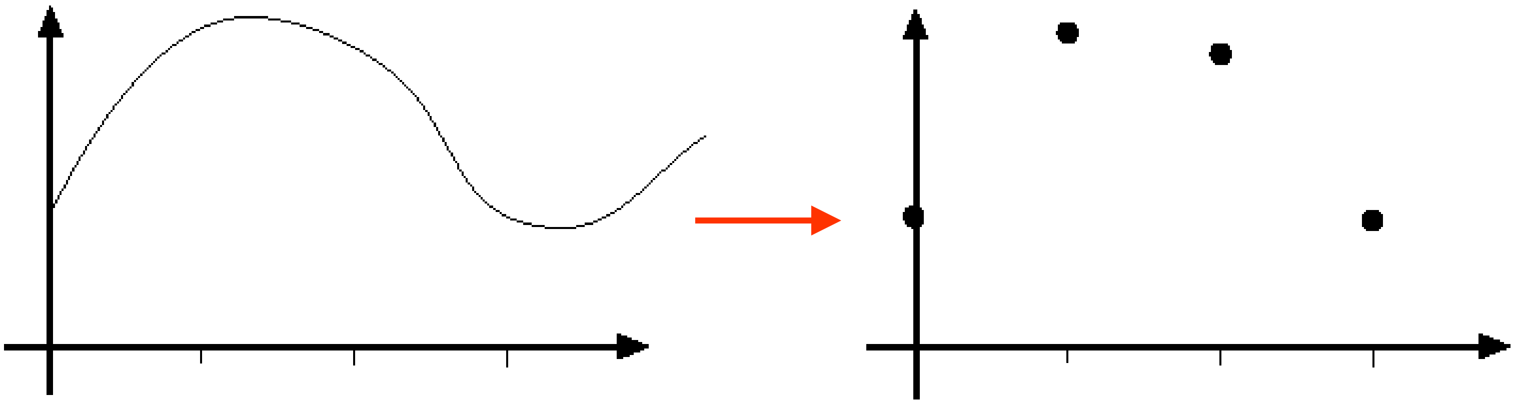
$$F.G = \mathbf{U}\left(f ** g\right)$$ (cfr. filtering)

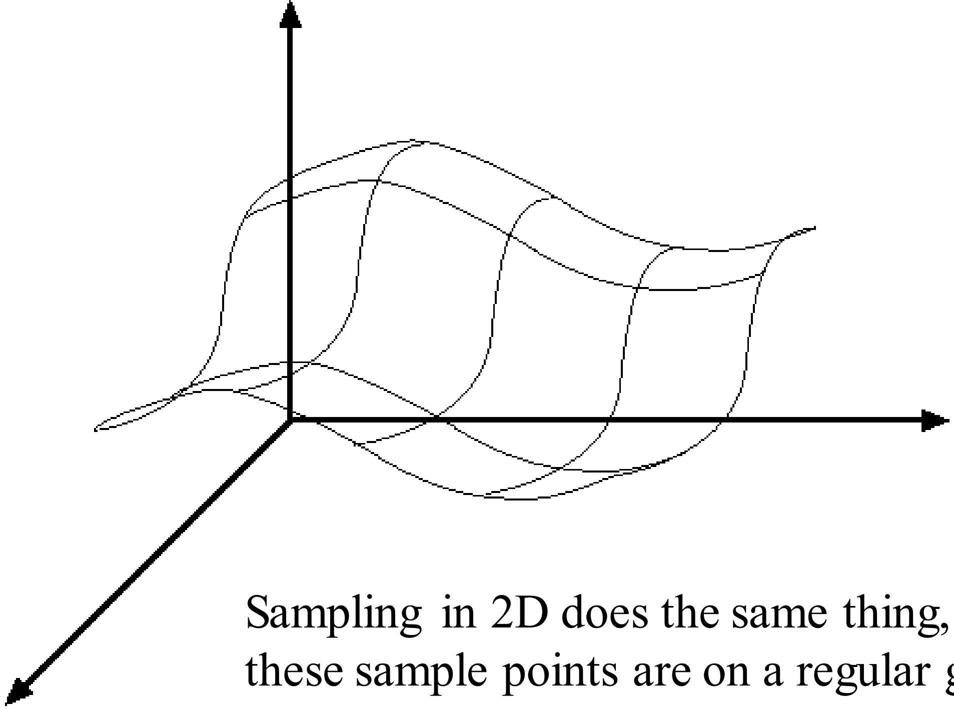
  - The Fourier transform of the product of two functions is the convolution of the Fourier transforms

$$F ** G = \mathbf{U}\left(f . g\right)$$ (cfr. sampling)

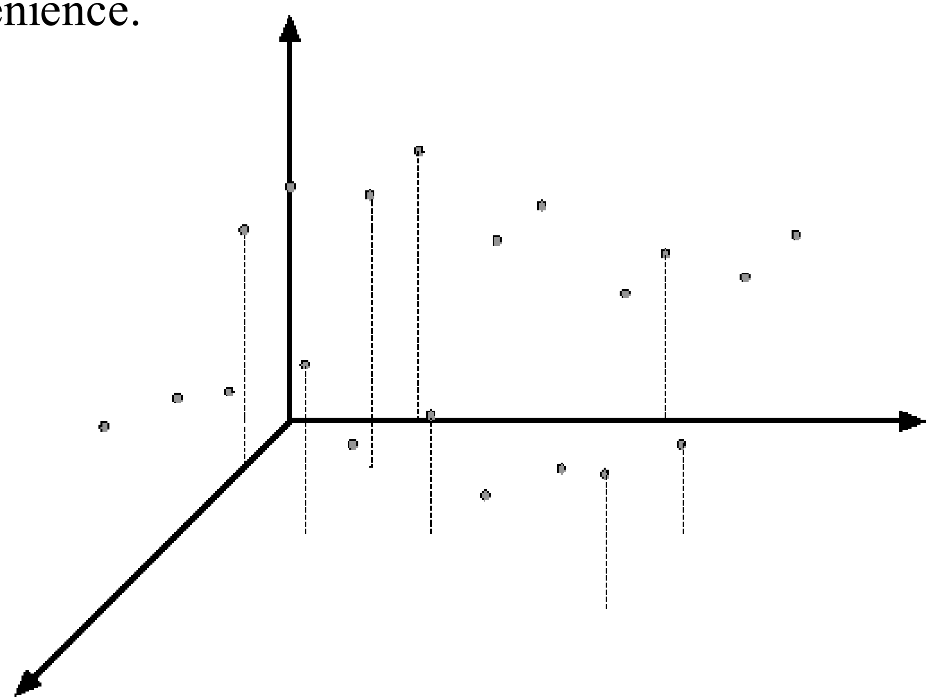
Sampling

- Go from continuous world to discrete world, from function to vector
- Samples are typically measured on regular grid





Sampling in 2D does the same thing, only in 2D. We'll assume that these sample points are on a regular grid, and can place one at each integer point for convenience.



A continuous model for a sampled function

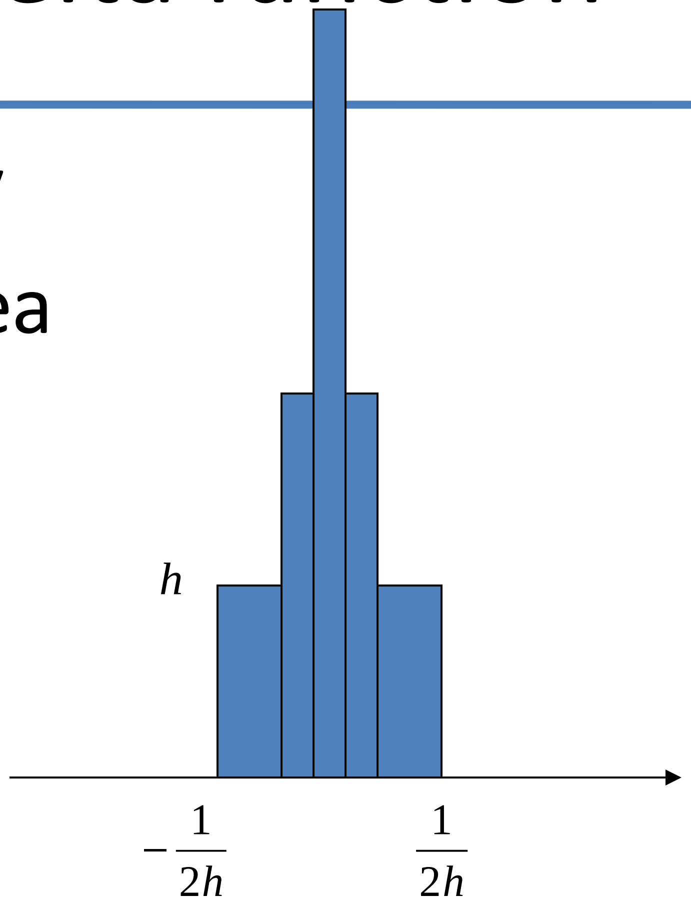
- We want to be able to approximate integrals sensibly [remember: $\iint_{\mathrm{R}^2} g(x,y) e^{-i2\pi(ux+vy)} dx dy$]
- Leads to
  - the delta function
  - model on right

$$\mathrm{Sample}_{2\mathrm{D}}(f(x,y)) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(x,y)\delta(x-i, y-j)$$

$$= f(x,y) \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(x-i, y-j)$$

Delta function

- limit to infinity of constant area function:



A continuous model for a sampled function

- We want to be able to approximate integrals sensibly
- Leads to
  - the delta function
  - model on right

$$\mathrm{Sample}_{2D}\left(f(x,y)\right)=\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f(x,y)\delta(x-i,y-j)$$

$$= f(x,y)\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x-i,y-j)$$

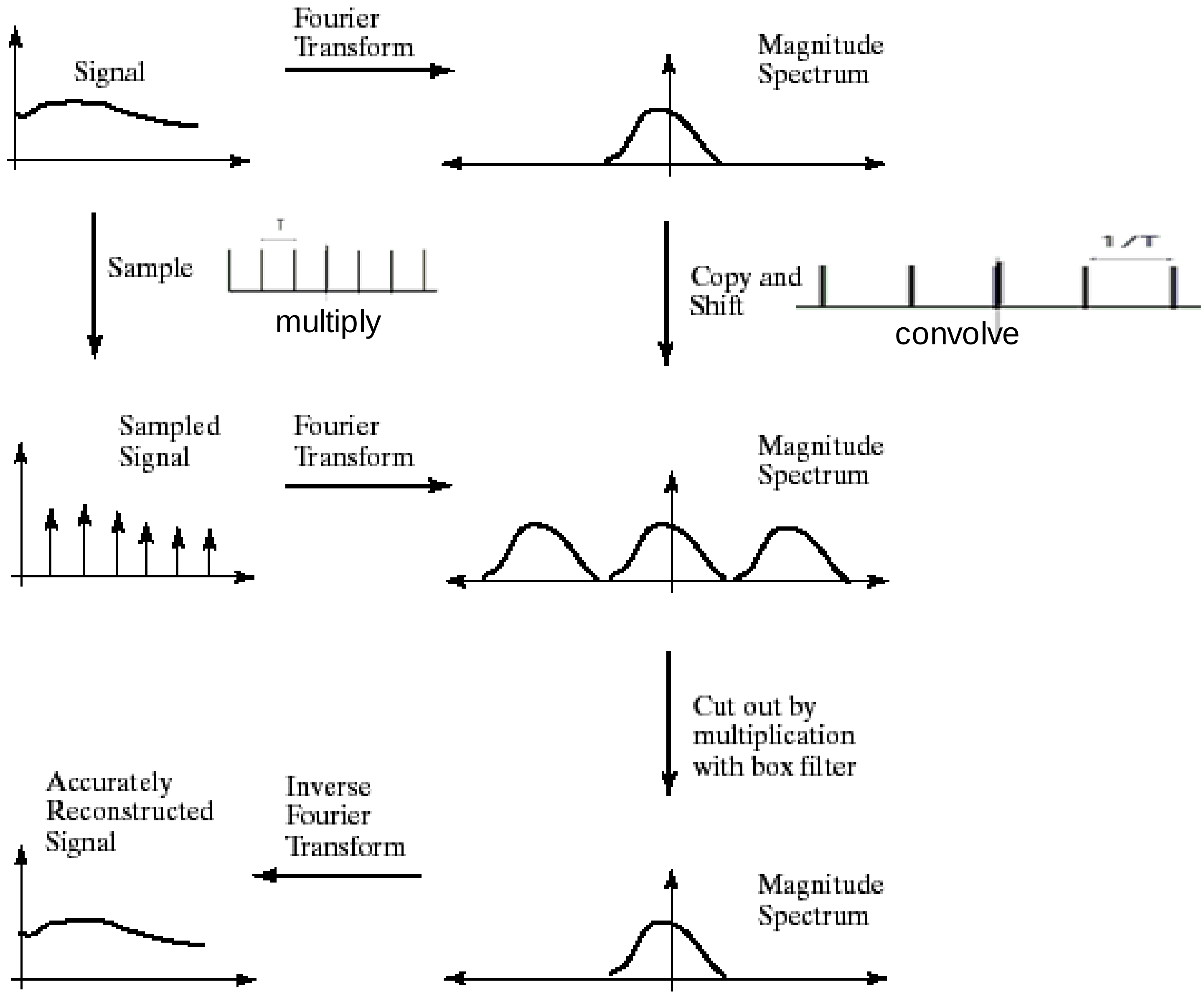
The Fourier transform of a sampled signal

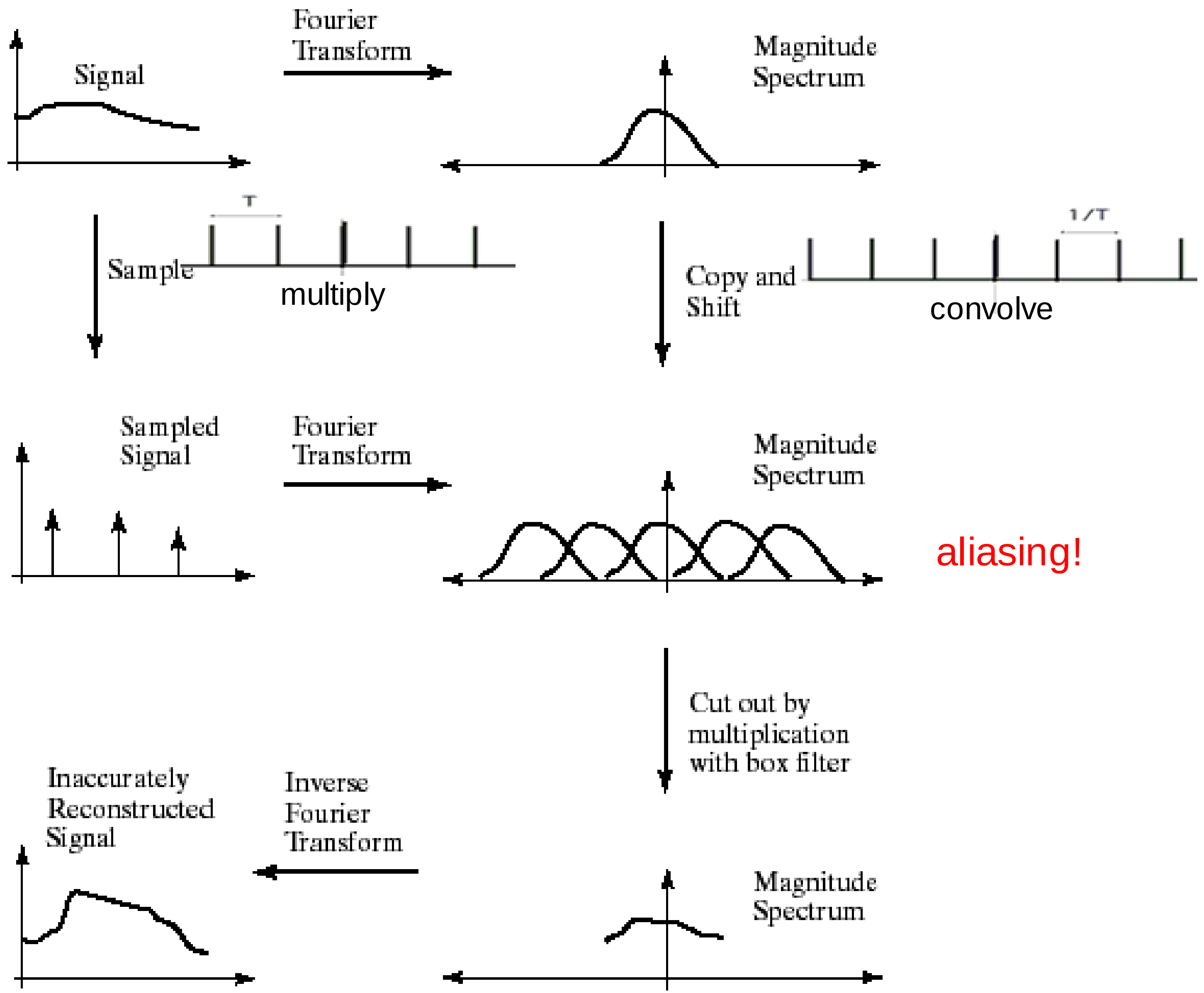
$$F\left(\mathrm{Sample}_{2\mathrm{D}}(f(x,y))\right) = F\left(f(x,y) \sum_{i=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} \delta(x-i, y-j) \right)$$

$$= F\left(f(x,y)\right) ** F\left(\sum_{i=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} \delta(x-i, y-j) \right)$$

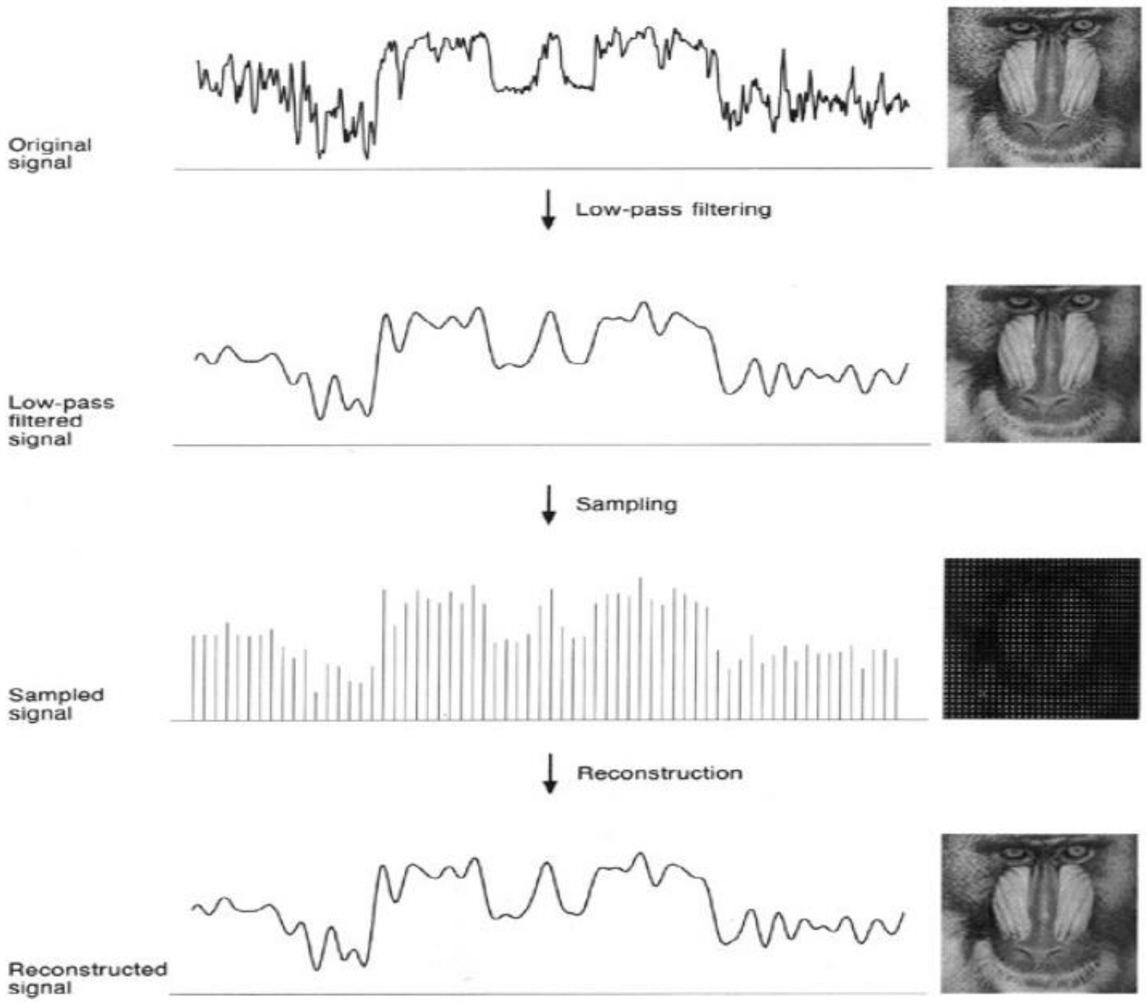
$$= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} F(u-i, v-j)$$

(notice f(x,y) is a continuous function and we use the continuous form of the Fourier transform)


Signal
Fourier Transform
Magnitude Spectrum
Sample
T
multiply
Copy and Shift
1/T
convolve
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Accurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum


Signal
Fourier Transform
Magnitude Spectrum
T
Sample
multiply
1/T
Copy and Shift
convolve
Sampled Signal
Fourier Transform
Magnitude Spectrum
aliasing!
Cut out by multiplication with box filter
Inaccurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

Proper sampling

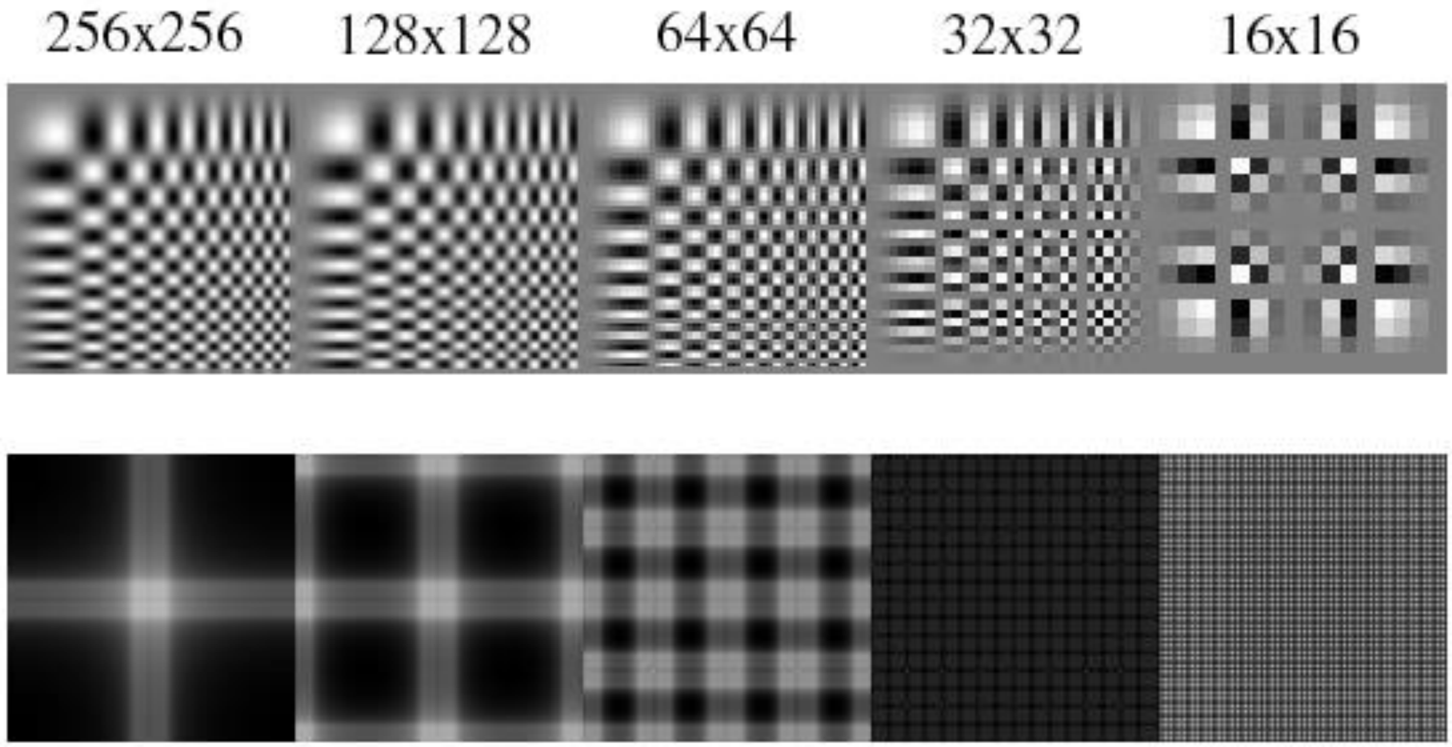


Slide from Dani Lischinski

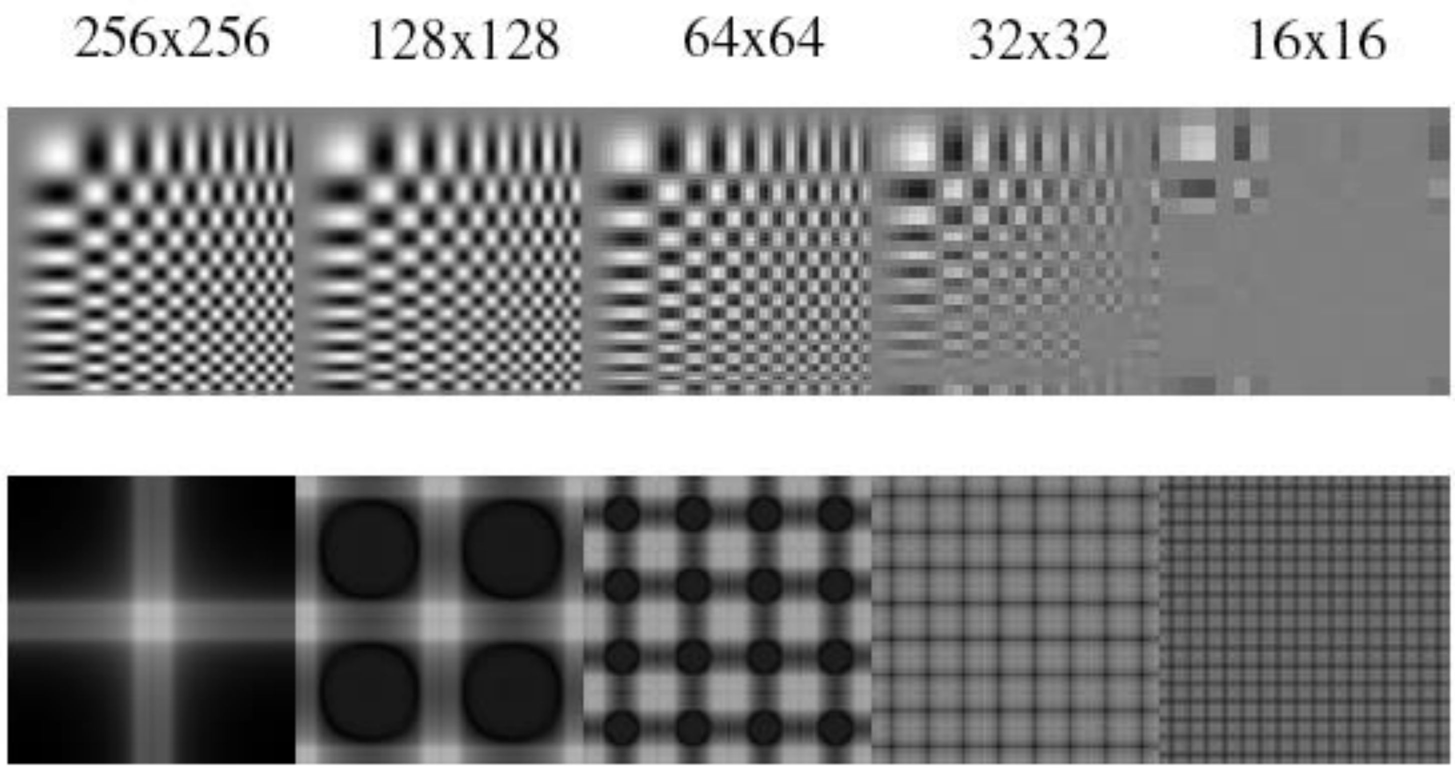
Smoothing as low-pass filtering

- The message of the FT is that high frequencies lead to trouble with sampling.
- Solution: suppress high frequencies before sampling
  - multiply the FT of the signal with something that suppresses high frequencies
  - or convolve with a low-pass filter
- A filter whose FT is a box is bad, because the filter kernel has infinite support
- Common solution: use a Gaussian
  - multiplying FT by Gaussian is equivalent to convolving image with Gaussian.

Sampling without smoothing.
Top row shows the images, sampled at every second pixel to get the next;
bottom row shows the magnitude spectrum of these images.

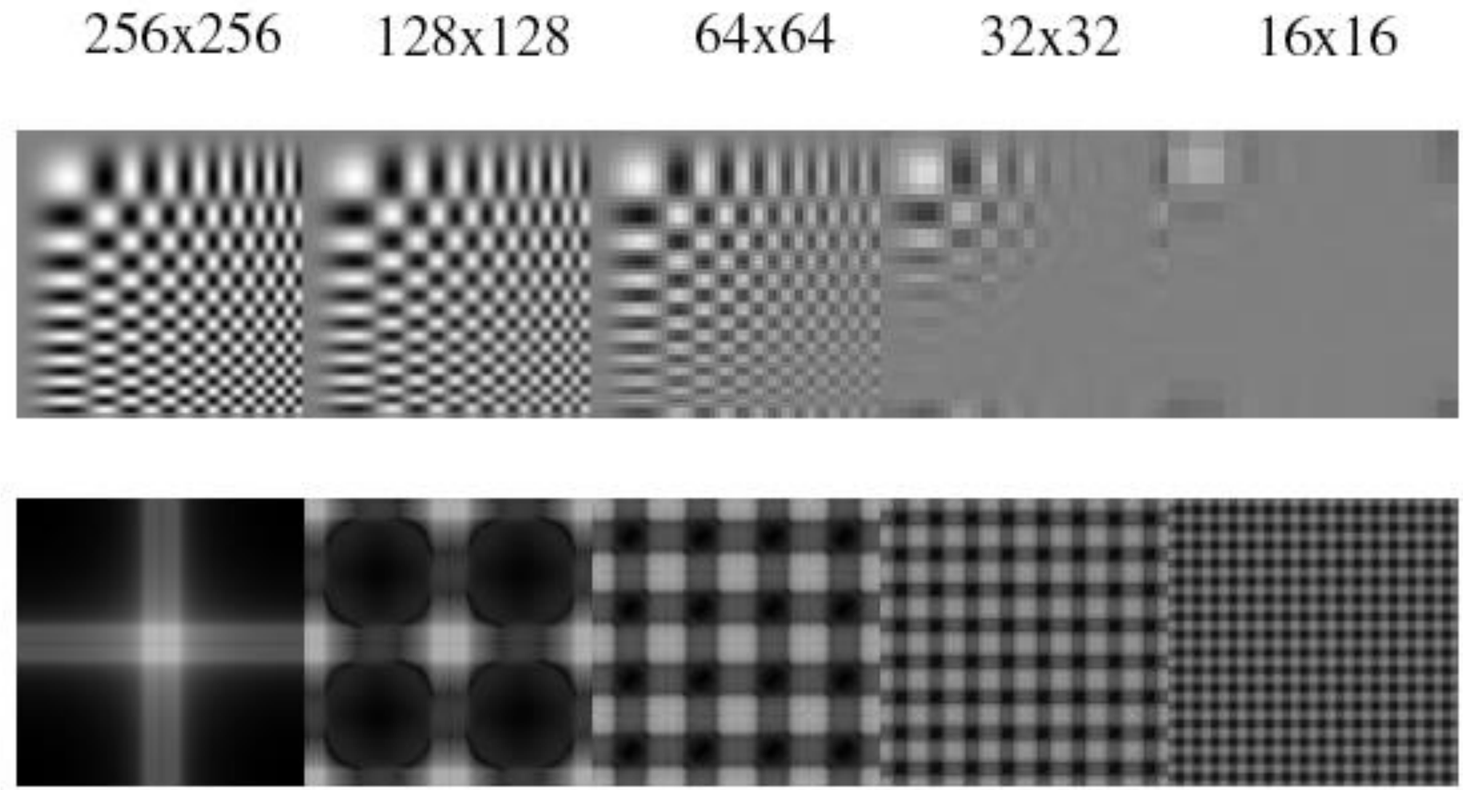


Sampling with smoothing.
Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1 pixel, then sampling at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.



Sampling with smoothing.

Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1.4 pixels, then sampling at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.

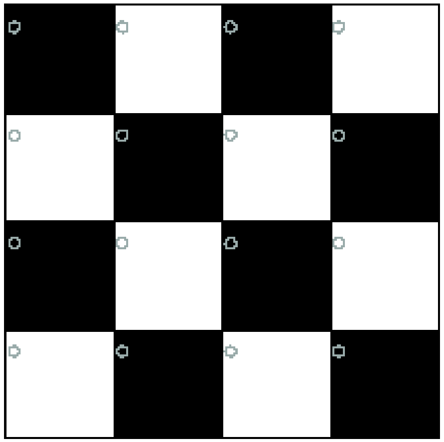


Nyquist sampling theorem

- Nyquist theorem: The sampling frequency must be at least twice the highest frequency

$$\omega_s \geq 2\omega$$

- If this is not the case the signal needs to be bandlimited before sampling, e.g. with a low-pass filter



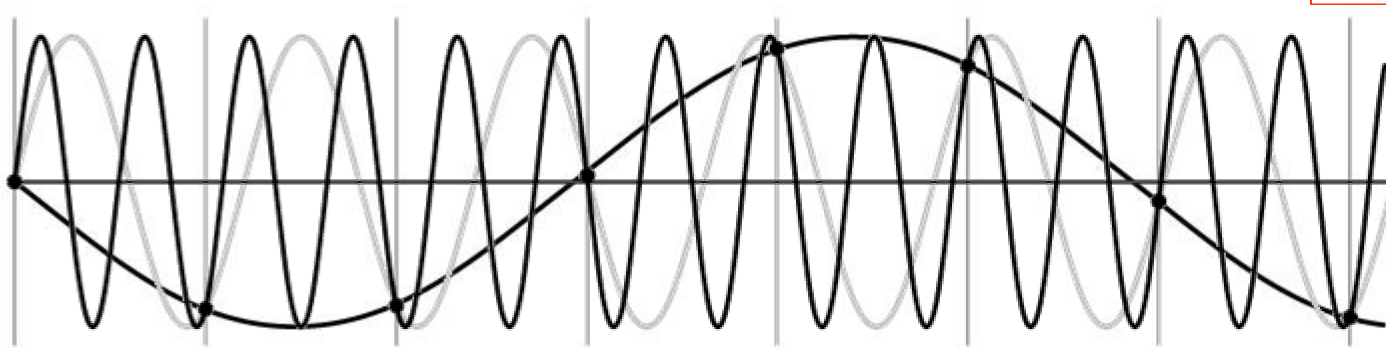
Computation of 2D Fourier Transform

- 2D Fourier Transform can be compute as sequence of 1D Fourier transforms

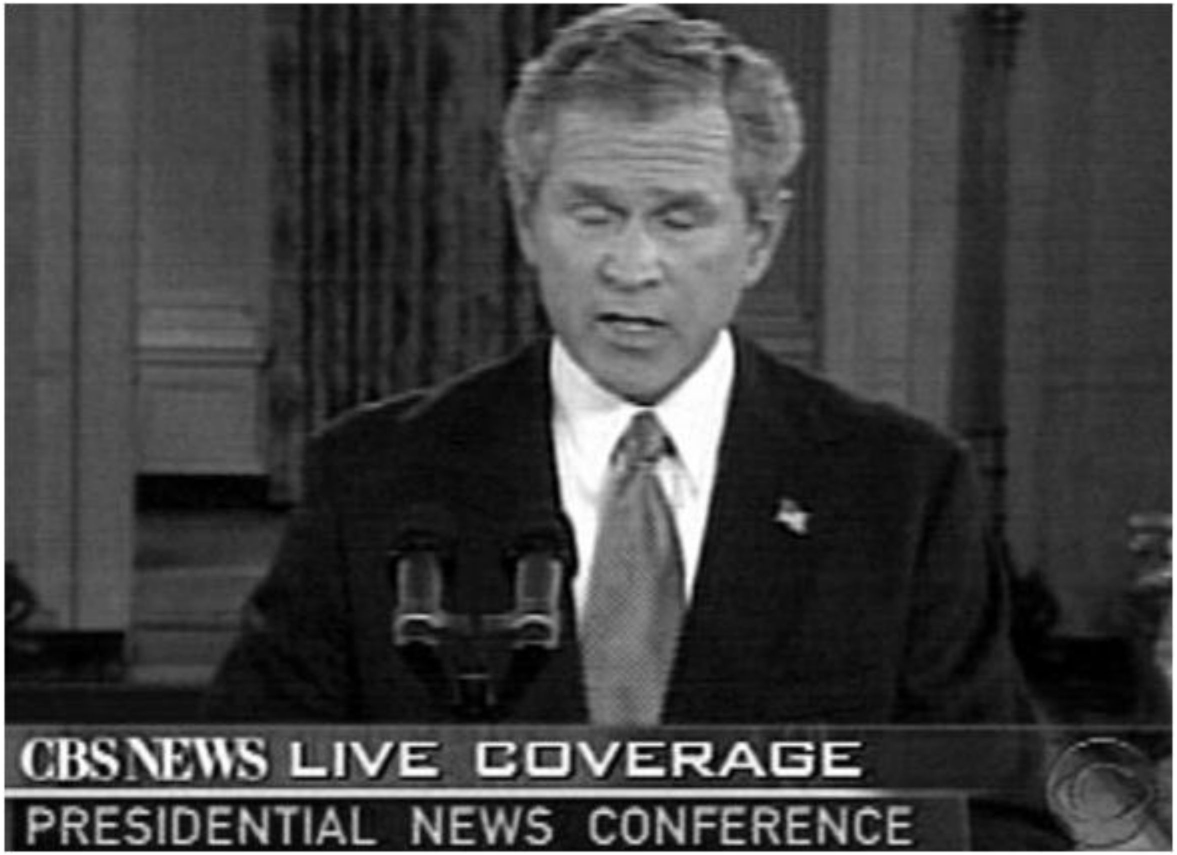
$$F(g(x,y))(u,v) = \iint g(x,y) e^{-i2\pi(ux+vy)}\, dx\, dy$$
$$= \int \left(\int g(x,y) e^{-i2\pi(ux)}\, dx \right) e^{-i2\pi(vy)} dy$$

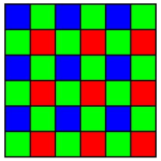
- Fast Fourier Transform (FFT) can compute Discrete Fourier Transform very fast (use symmetries)

$$\boxed{F = \mathbf{U}f}$$



What went wrong?





color sampled at half-resolution!

Signal reconstruction

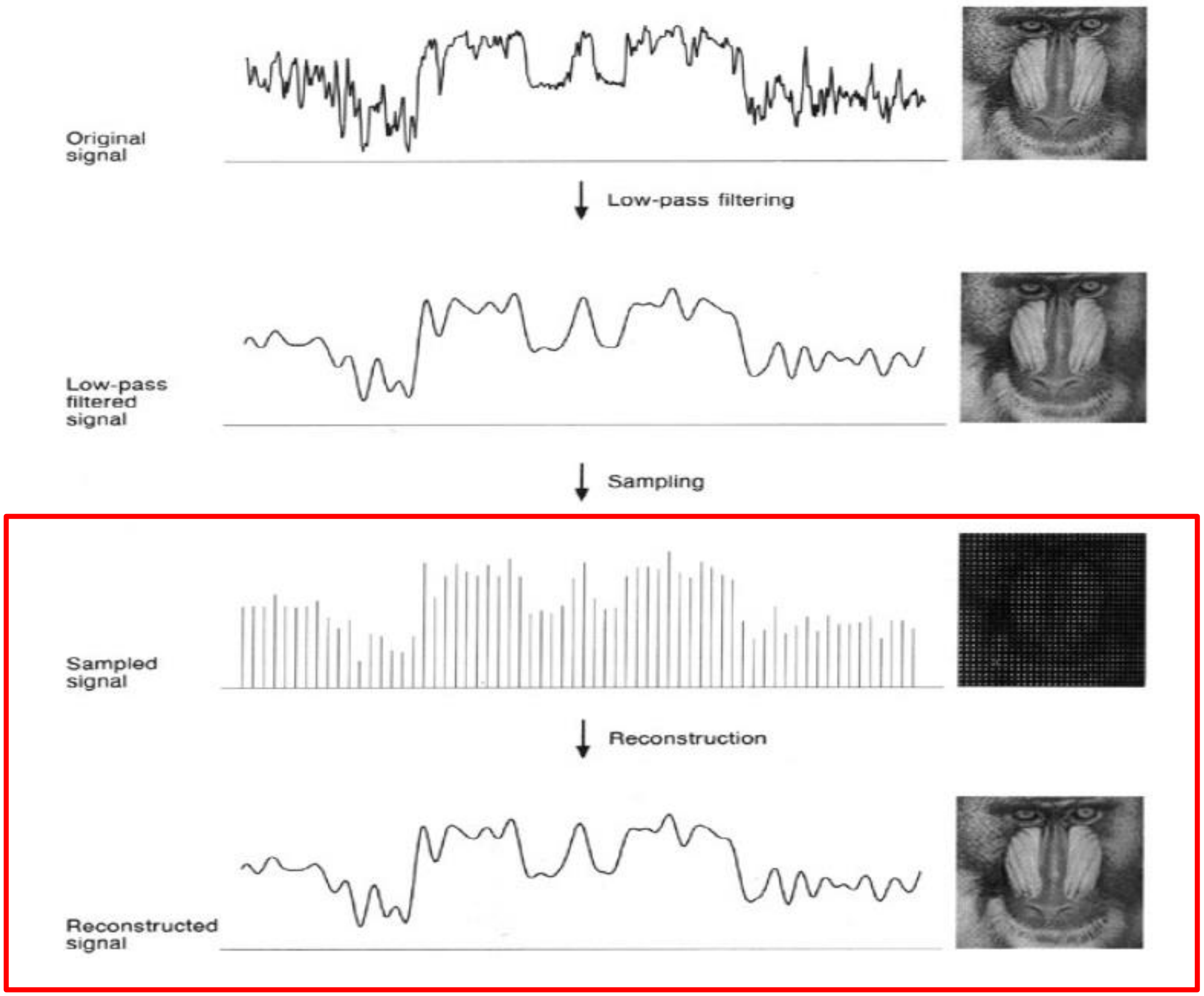
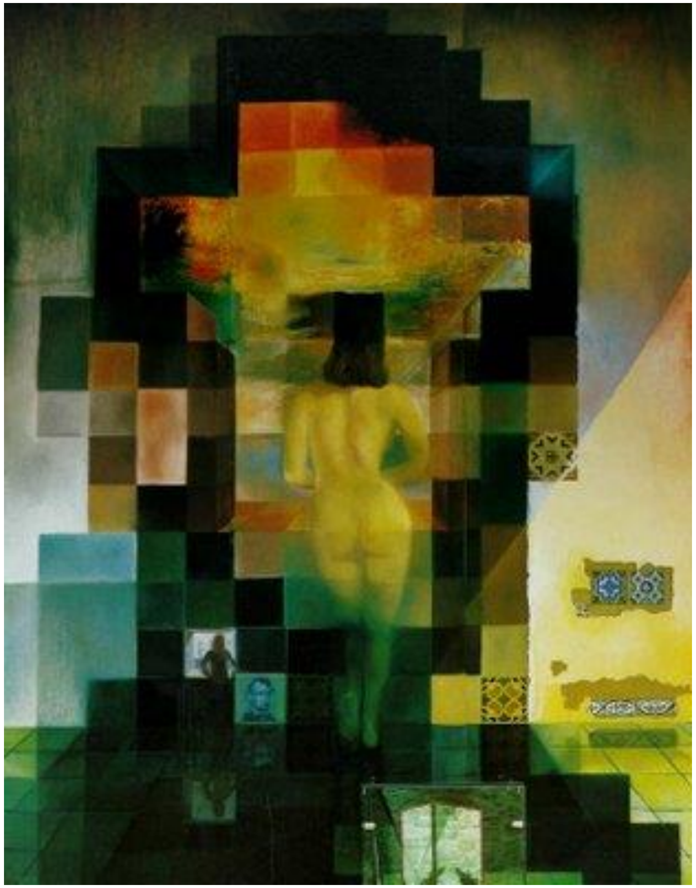


Image reconstruction: pixelization

- Who is this?

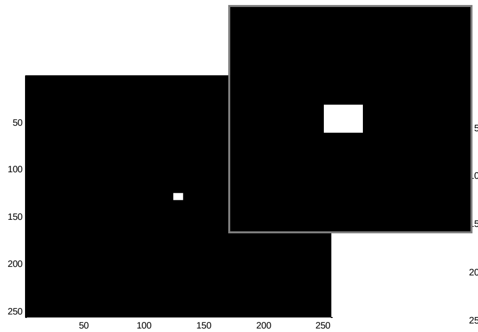
Harmon & Julesz 1973



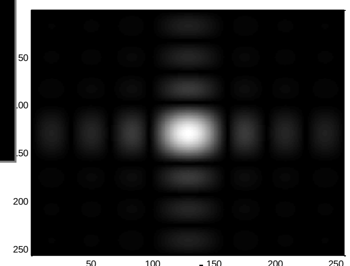
Dali 1976

Reconstruction filters

Square pixels

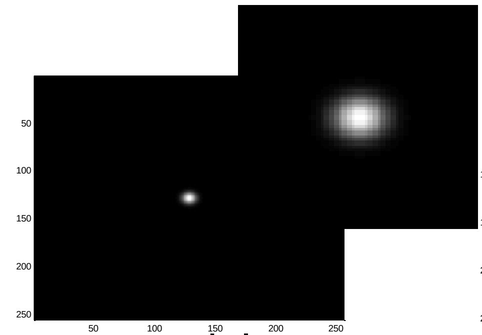


spatial

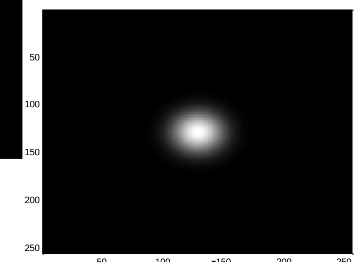


Fourier

Gaussian reconstruction filter

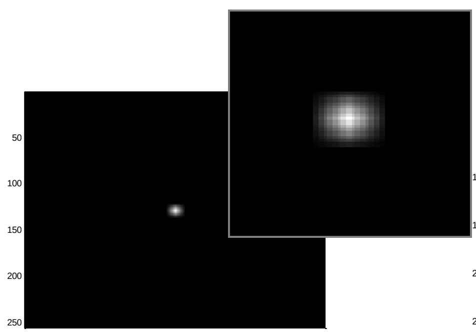


spatial

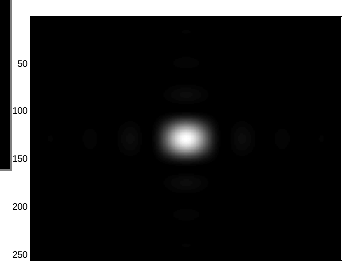


Fourier

Bilinear interpolation

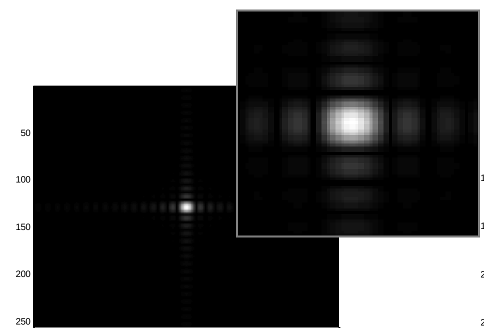


spatial

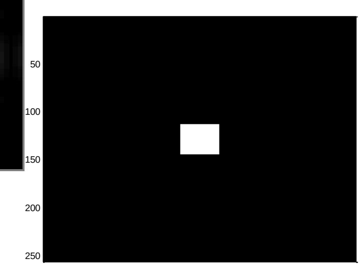


Fourier

Perfect reconstruction filter



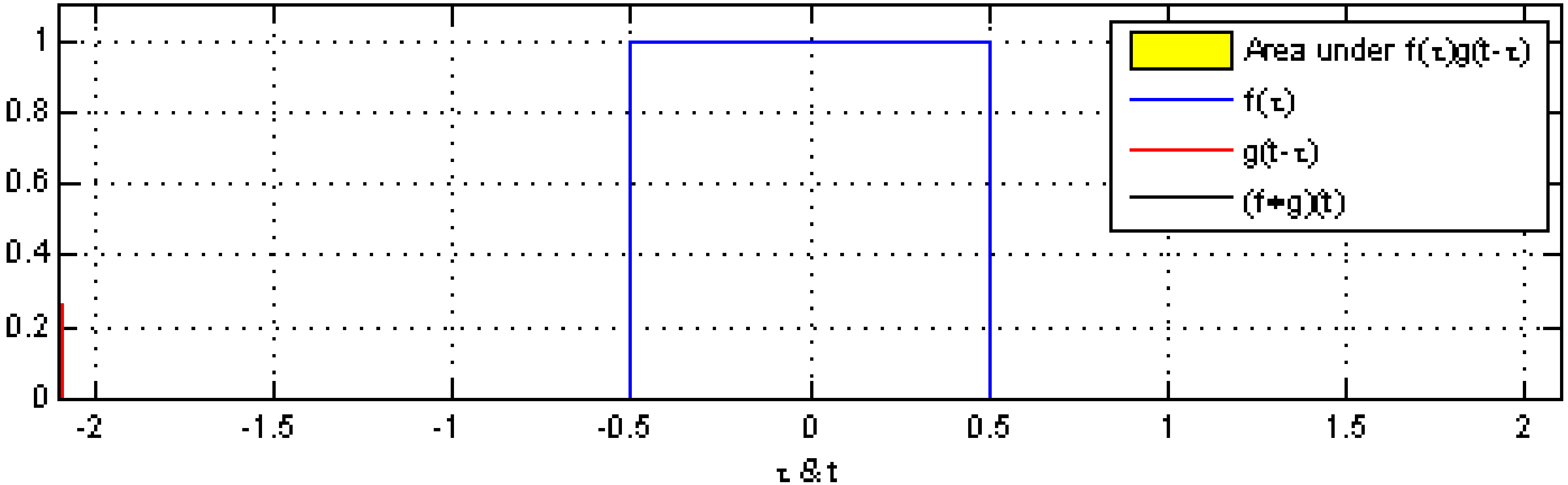
spatial



Fourier

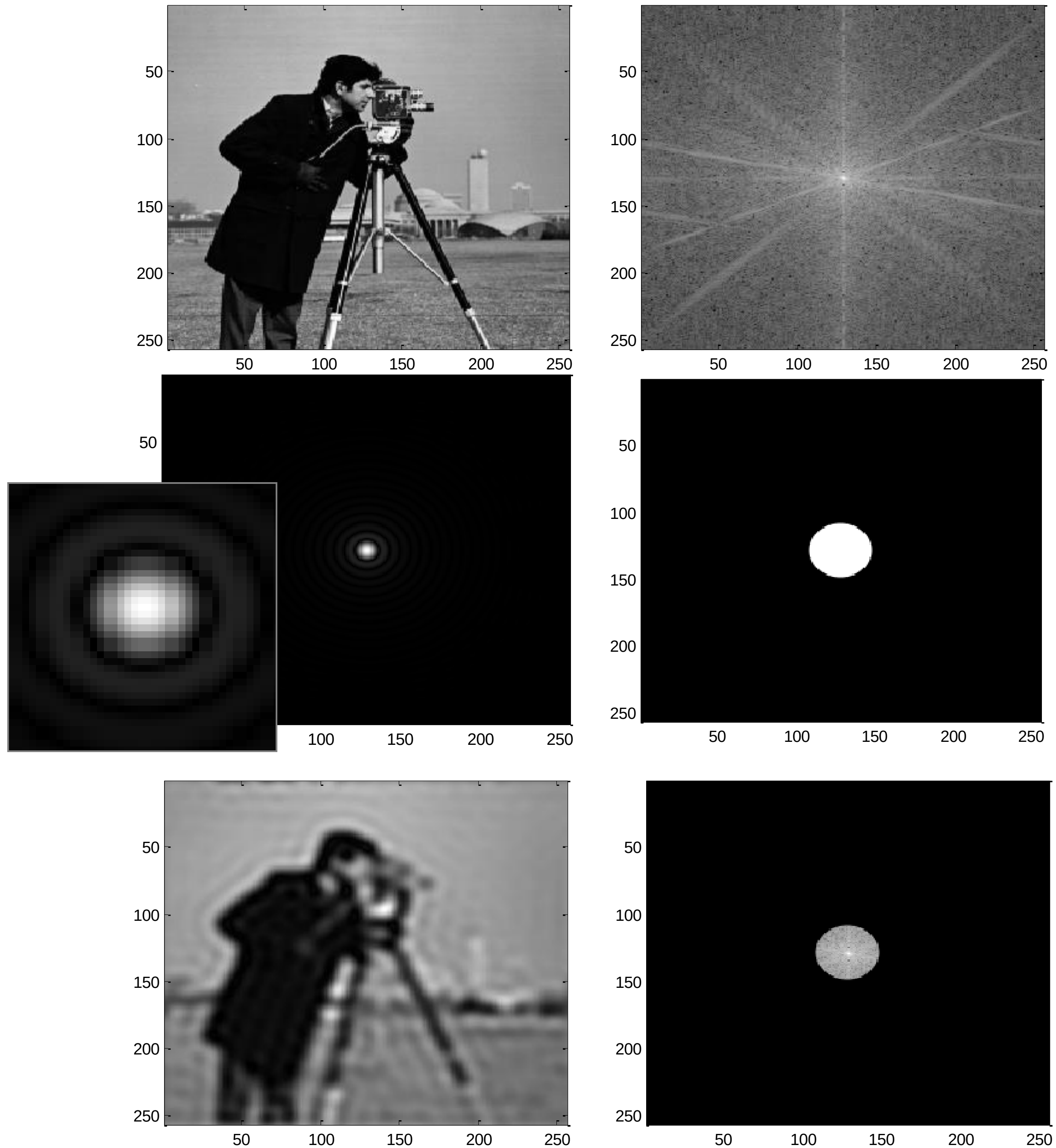
ETH

Convolution of Box with Box

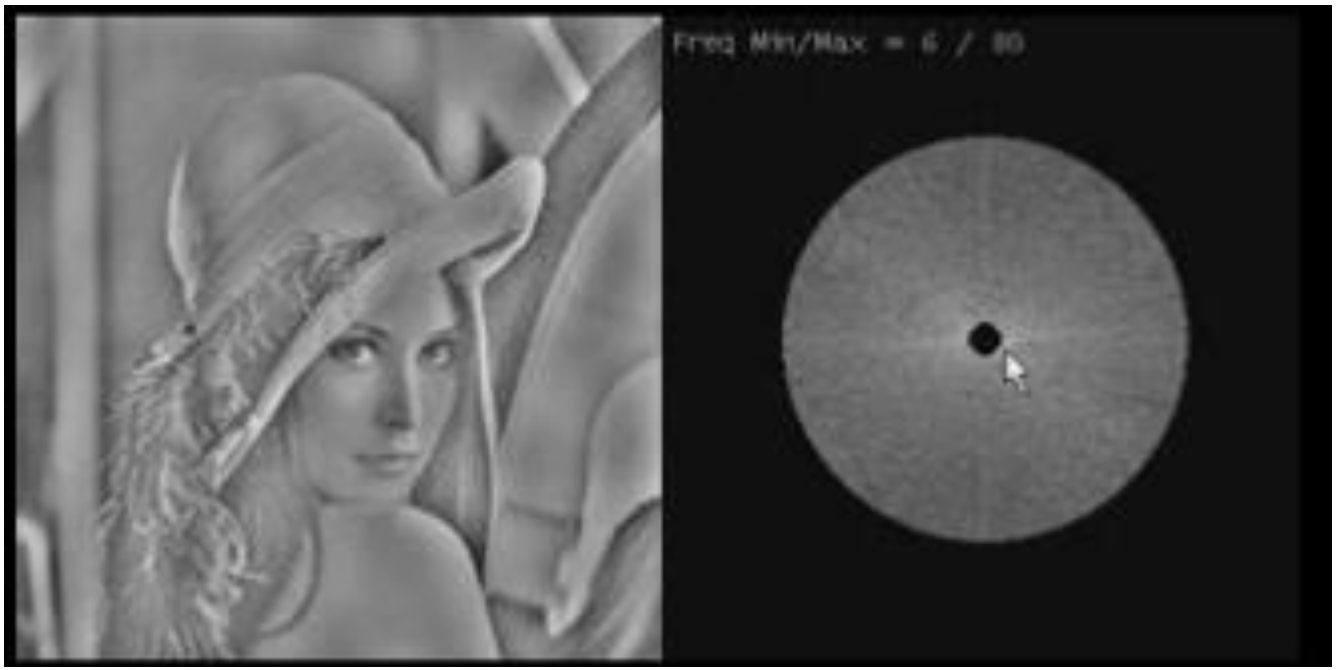


https://commons.wikimedia.org/wiki/File:Convolution_of_box_signal_with_itself2.gif

Designing the 'perfect' low-pass filter

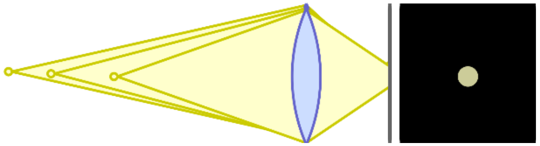


Filtering in Fourier domain

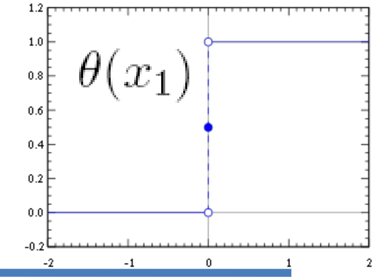


ETH

Defocus blurring



Motion blurring



Each light dot is transformed into a short line along the x_1-axis:

$$h(x_1, x_2) = \frac{1}{2l}\left[\theta(x_1 + l) - \theta(x_1 - l)\right]\delta(x_2)$$

Image restoration problem

$$f(\mathbf{x}) \longrightarrow \boxed{h(\mathbf{x})} \longrightarrow g(\mathbf{x}) \longrightarrow \boxed{\tilde{h}(\mathbf{x})} \longrightarrow f(\mathbf{x})$$

The 'inverse' kernel $\tilde{h}(\mathbf{x})$ should compensate the effect of the image degradation $h(\mathbf{x})$, i.e.,

$$(\tilde{h} * h)(\mathbf{x}) = \delta(\mathbf{x})$$

$\tilde{h}$ may be determined more easily in Fourier space:

$$\mathcal{F}[\tilde{h}](u,v) \cdot \mathcal{F}[h](u,v) = 1$$

To determine $\mathcal{F}[\tilde{h}]$ we need to estimate

1. the distortion model $h(\mathbf{x})$ (point spread function) or $\mathcal{F}[h](u,v)$ (modulation transfer function)
2. the parameters of $h(\mathbf{x})$, e.g. r for defocussing.

Image Restoration: Motion Blur

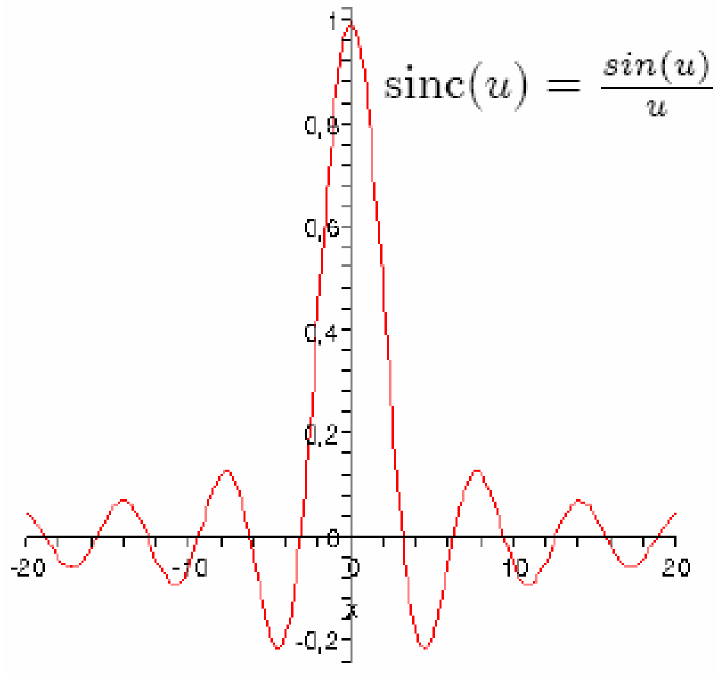
Kernel for motion blur $h(\mathbf{x}) = \frac{1}{2l}\big(\theta(x_1 + l) - \theta(x_1 - l)\big)\delta(x_2)$

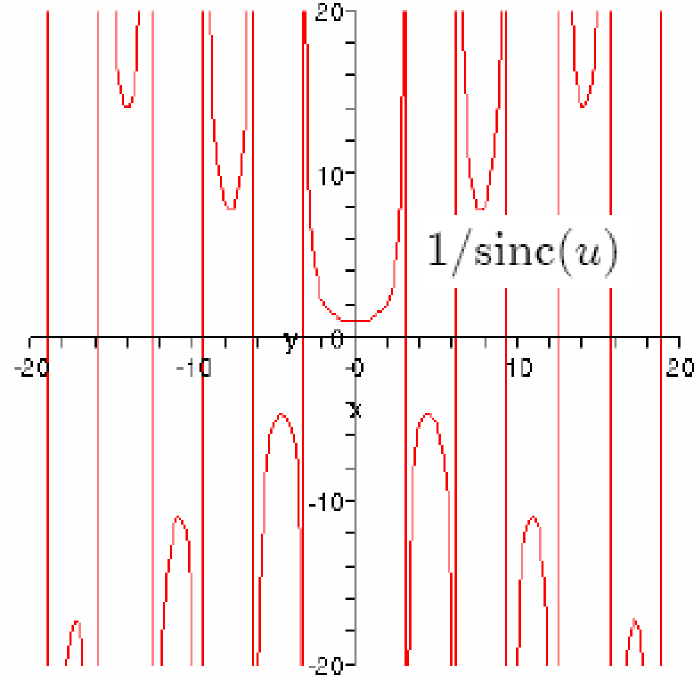
(a light dot is transformed into a small line in x_1 direction).

Fourier transformation:

$$\begin{aligned}\mathcal{F}[h](u,v) &= \frac{1}{2l}\int_{-l}^{+l} \exp(-\imath 2\pi u x_1) \underbrace{\int_{-\infty}^{+\infty} \delta(x_2)\exp(-\imath 2\pi v x_2) dx_2}_{=1} \, dx_1 \\ &= \frac{\sin(2\pi u l)}{2\pi u l} =: \operatorname{sinc}(2\pi u l)\end{aligned}$$

ETH





$\hat{h}(u) = \mathcal{F}[h](u) = \mathrm{sinc}(2\pi u l)$

$\mathcal{F}[\tilde{h}](u) = 1/\hat{h}(u)$

Problems:

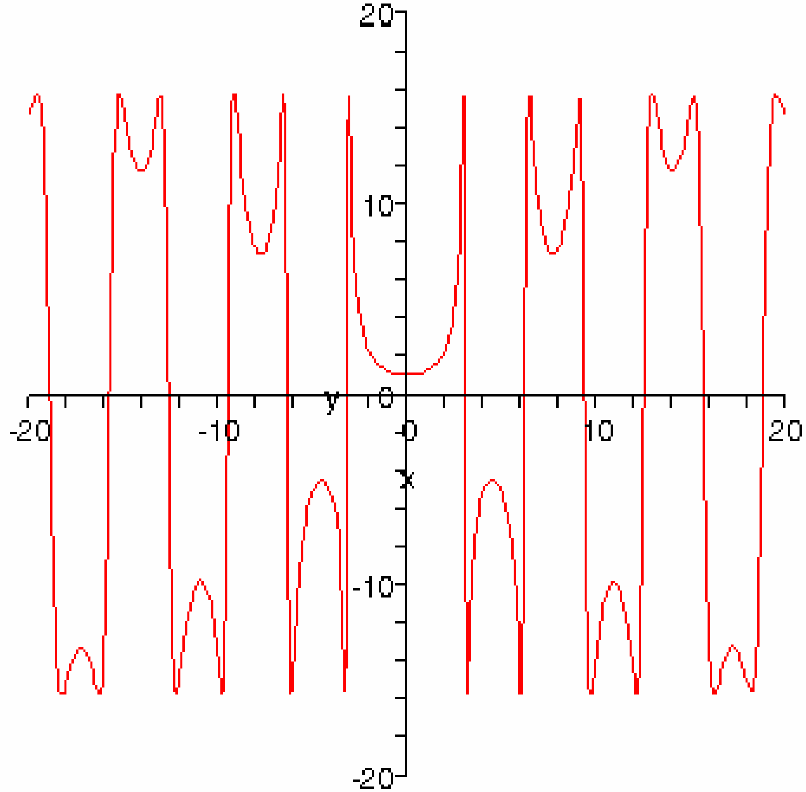
- Convolution with the kernel h completely cancels the frequencies $\frac{\nu}{2l}$ for $\nu \in \mathcal{Z}$. Vanishing frequencies cannot be recovered!
- Noise amplification for $\mathcal{F}[h](u, v) \ll 1$.

Avoiding noise amplification

Regularized
reconstruction filter:

$$\tilde{\mathcal{F}}[\tilde{h}](u,v) = \frac{\mathcal{F}[h]}{|\mathcal{F}[h]|^2 + \epsilon}$$

Singularities are avoided by the regularization ϵ.



The size of ϵ implicitly determines an estimate of the noise level in the image, since we discard signals which are dampened below the size ϵ.

Coded Exposure Photography: Assisting Motion Deblurring using Fluttered Shutter

Raskar, Agrawal, Tumblin (Siggraph2006)

Short Exposure | **Traditional** | **Coded**

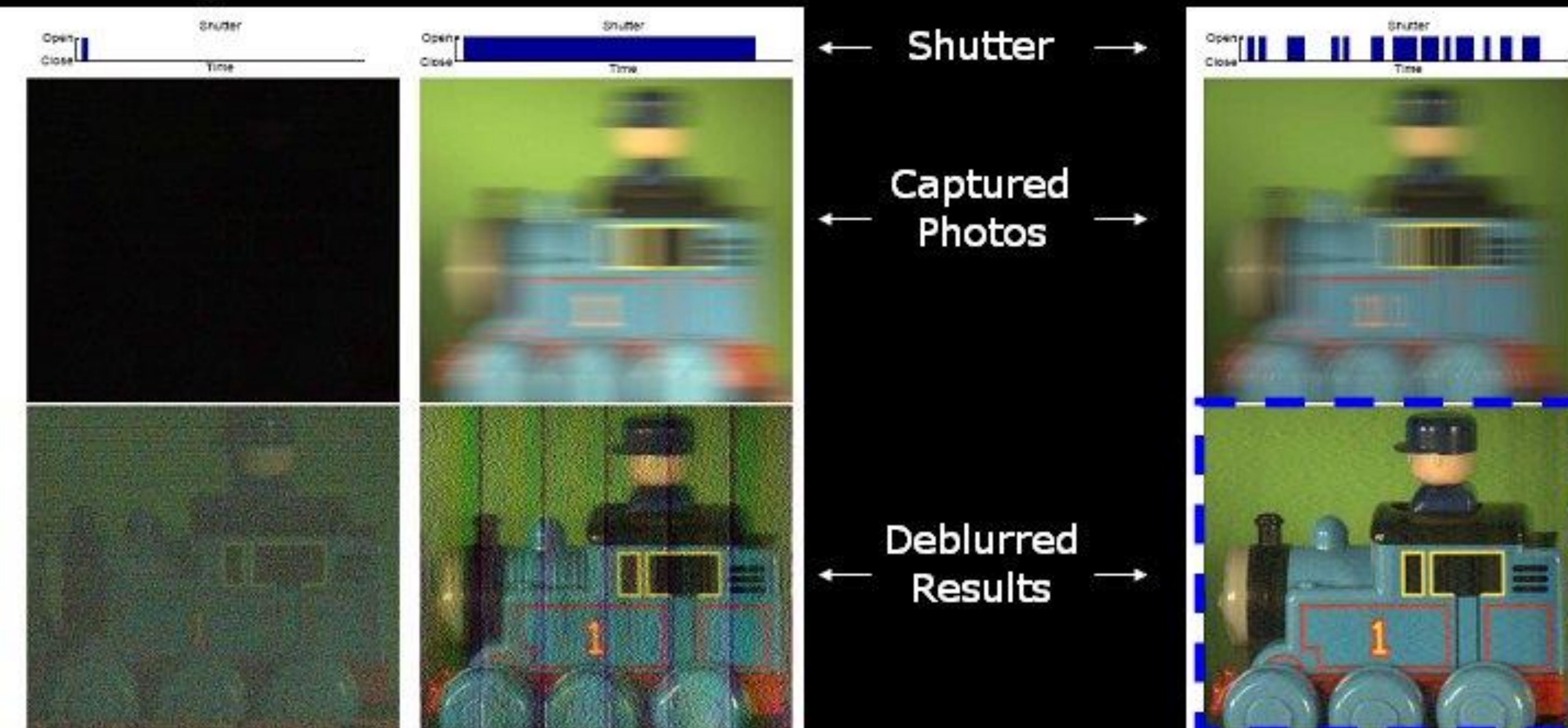


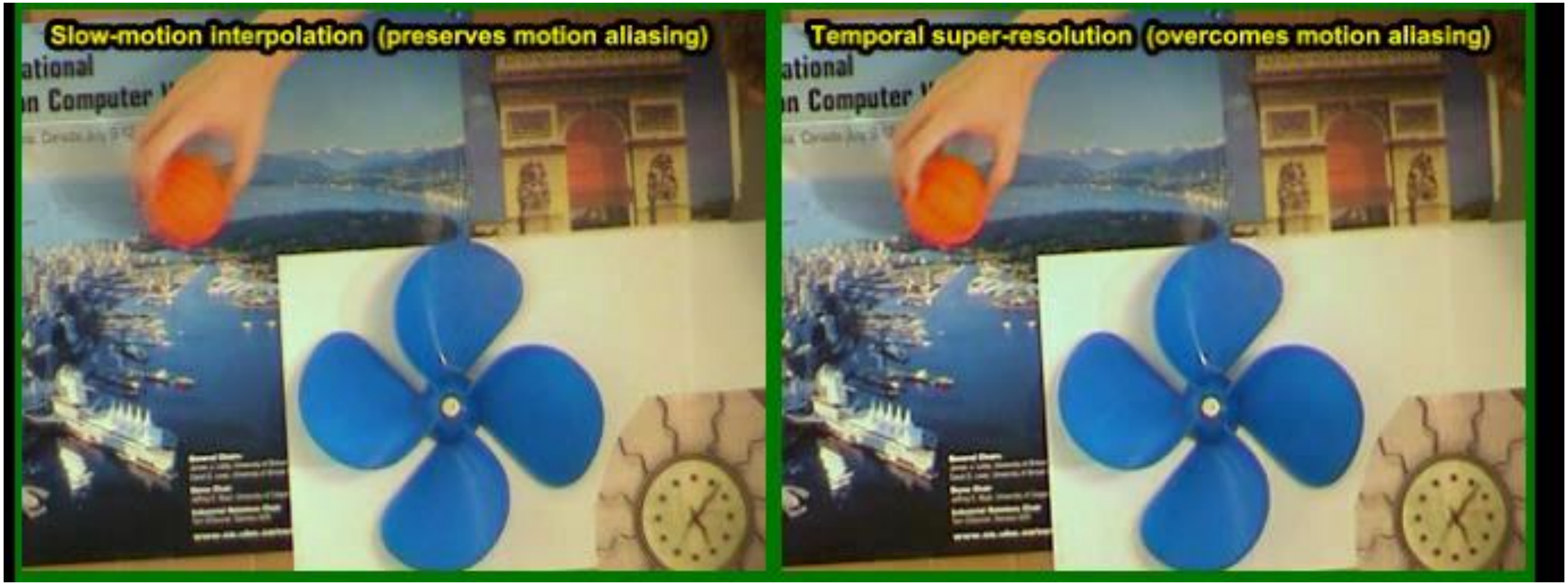
Image is dark and noisy

Result has Banding Artifacts and some spatial frequencies are lost

Decoded image is as good as image of a static scene

Space-time super-resolution

Shechtman et al. PAMI05

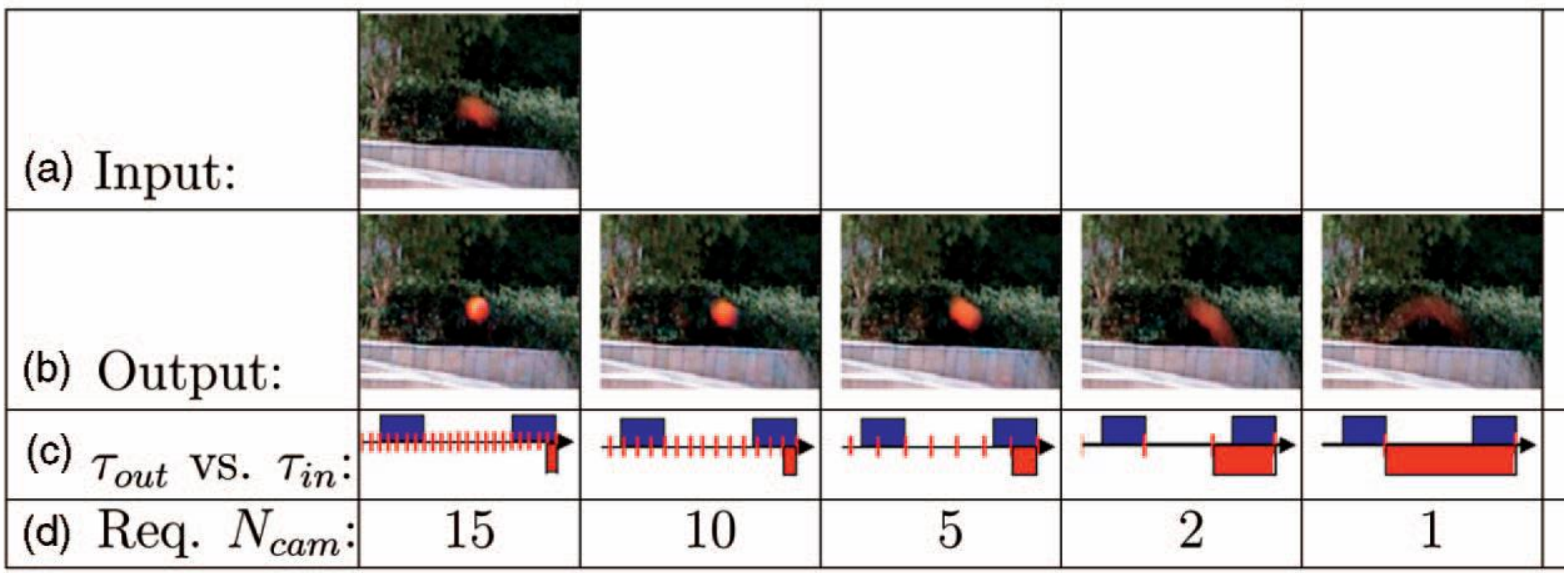


Space-time super-resolution

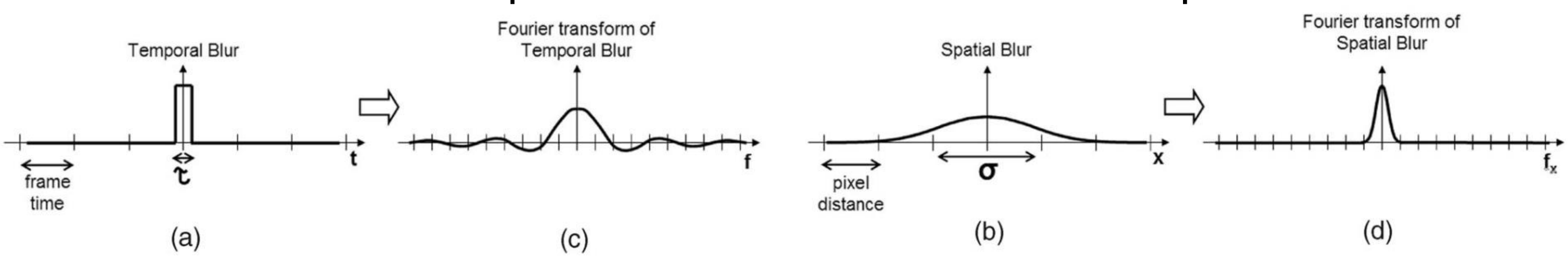
Shechtman et al. PAMI05

Space-time super-resolution

Shechtman et al. PAMI05

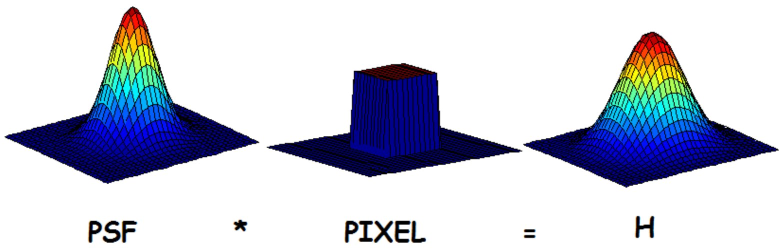


time super-resolution works better than space



Spatial super-resolution

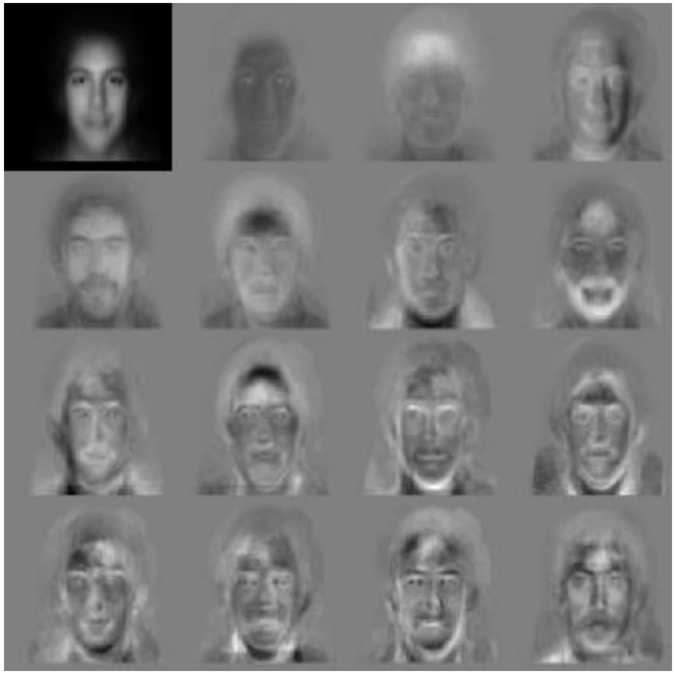
- lens+pixel=low-pass filter (desired to avoid aliasing)



- Low-res images = D*H*G*(desired high-res image)
  - D: decimate, H:lens+pixel, G: Geometric warp
- Simplified case for translation: LR=(D*G)*(H*HR)
  - G is shift-invariant and commutes with H
  - First compute H*HR, then deconvolve HR with H
- Super-resolution needs to restore attenuated frequencies
  - Many images improve S/N ratio (~sqrt(n)), which helps
  - Eventually Gaussian’s double exponential always dominates

Next week:
More transforms…

Eigenfaces



Wavelets