Deep Learning

5. Overview of classical computer vision - 1. Basic image processing, filtering, Fourier transform, image deblurring, edge detection

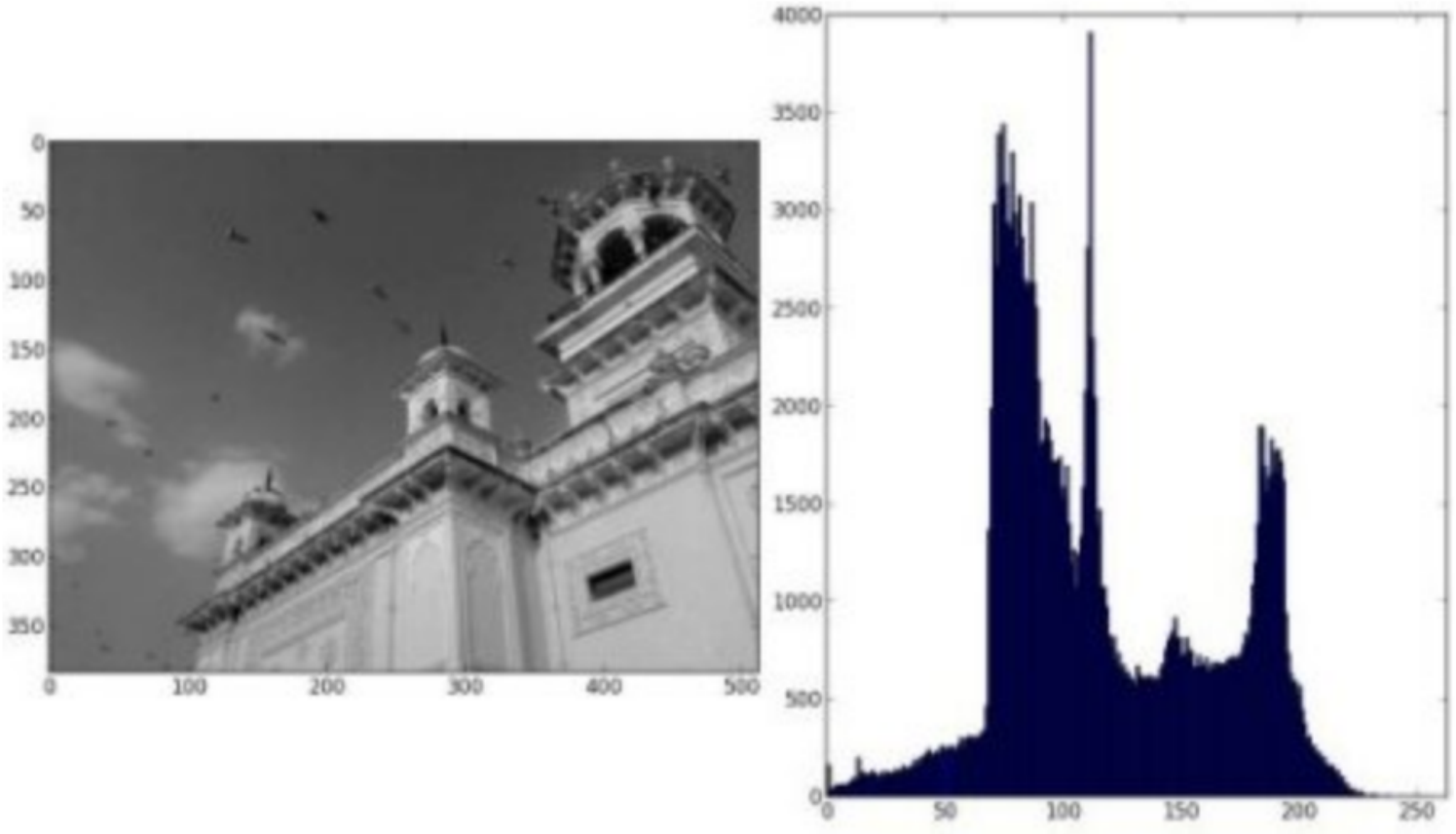
Viacheslav Dudar

Taras Shevchenko National University of Kyiv

2018

Histogram of the image

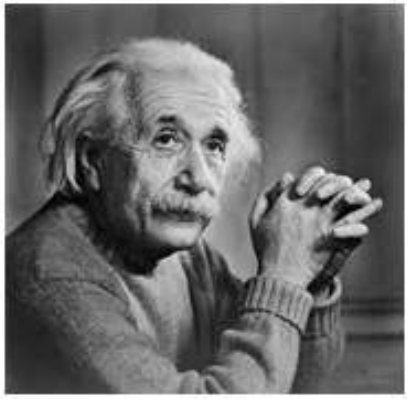
Histogram of the image: shows distribution of pixel intensities.



Point operations

Apply pixelwise transformation to the input image:
$J(x, y) = F(I(x, y))$
Example 1: digital negative
$F(x) = 1 - x$

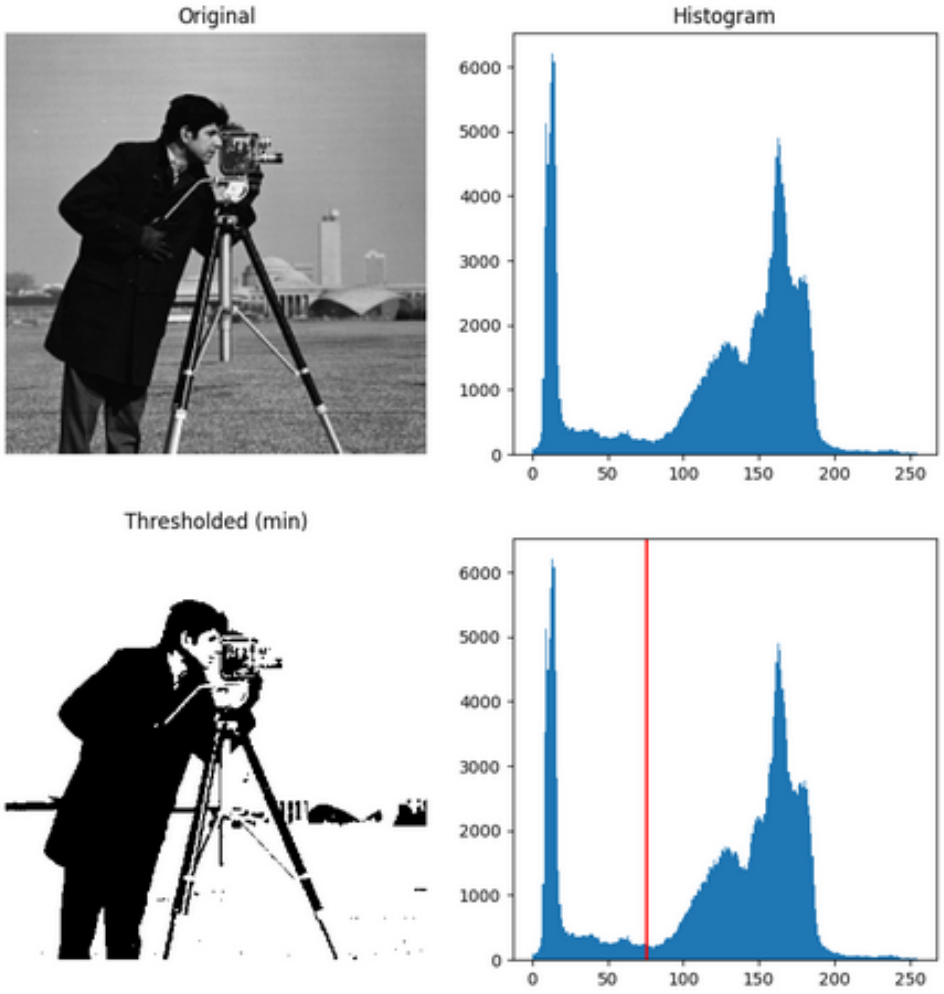
Input Image



Output Image

Point operations: thresholding

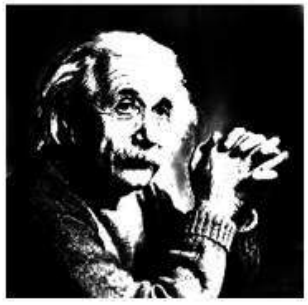
Example 2: image thresholding: $F(x) = (x > \theta)$



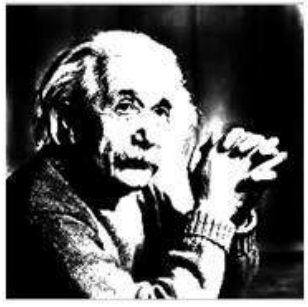
Point operations: power-law

Example 3: Gamma correction: $F(x) = x^{\gamma}$

Gamma = 10



Gamma = 8



Gamma = 6

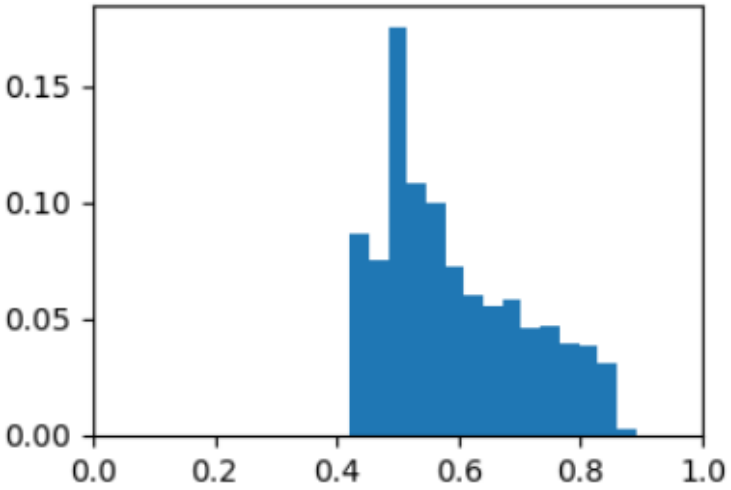
Point operations: contrast stretching

$$g = \frac{f - f_{\min}}{f_{\max} - f_{\min}}$$

Example 4: contrast stretching:

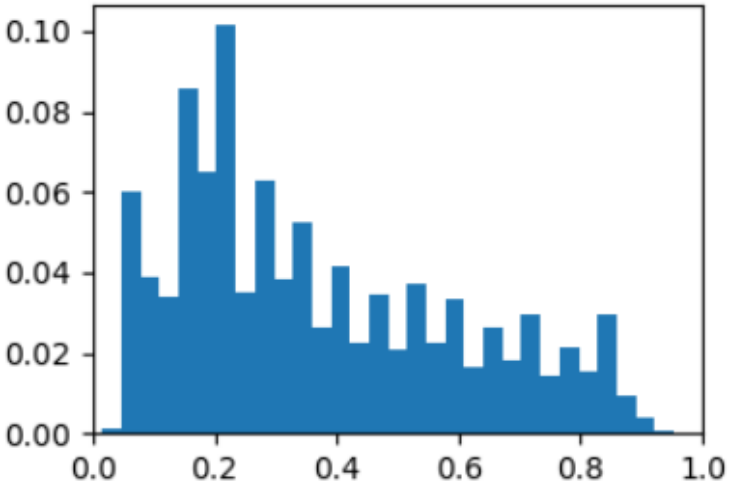
Low contrast orginal

Histogram of low contrast image



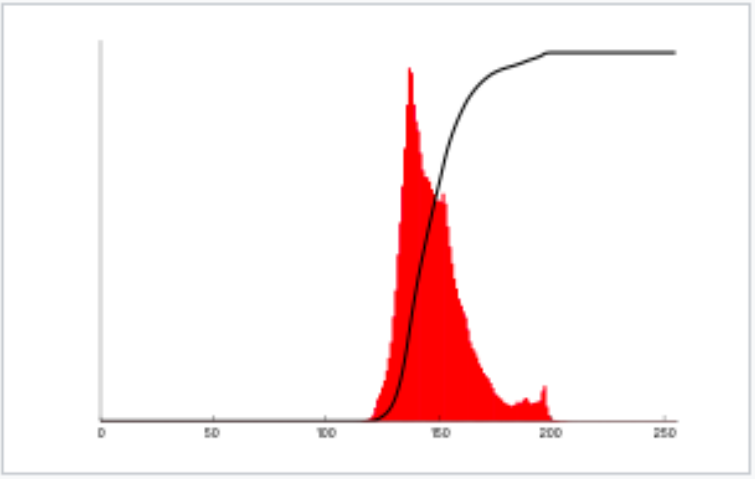
Contrast Stretched

Histogram of contrast stretched image



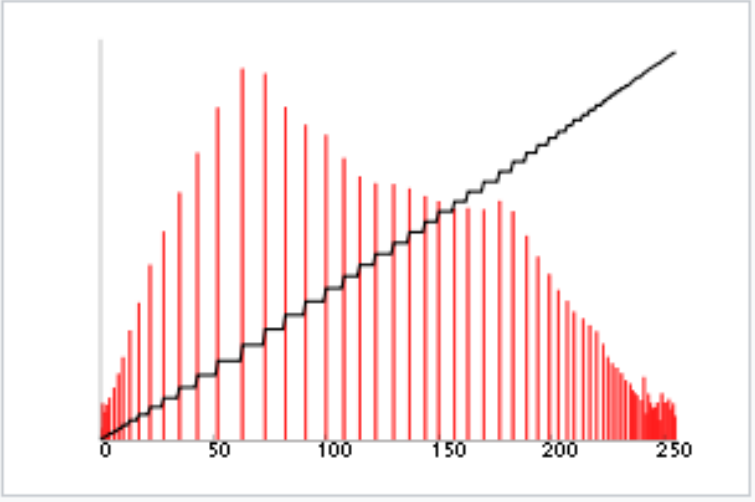
Point operations: Histogram equalization

Before Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

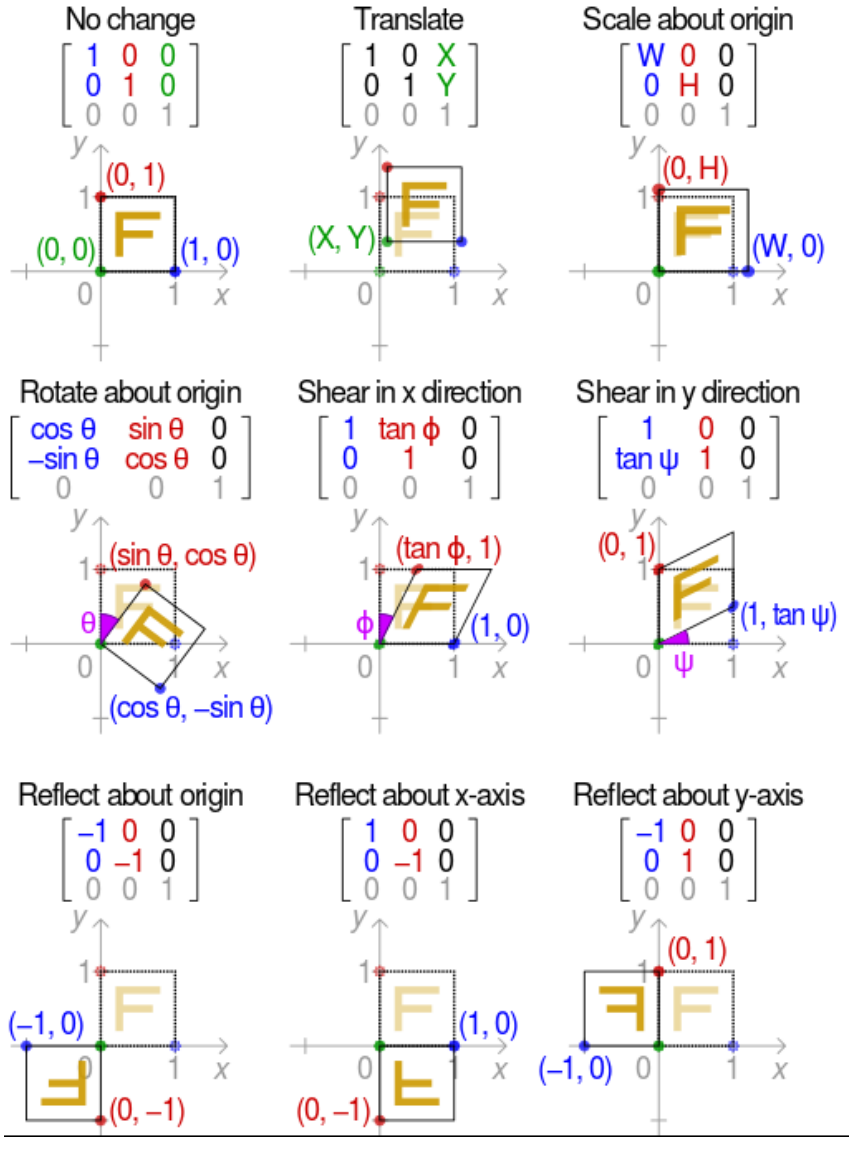
After Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

Geometric transformations

General formula for geometric transformation: $J(x, y) = I(T(x, y))$



Affine tranformation:

$T(x, y) = \begin{pmatrix} ax+by+c \\ dx+ey+f \end{pmatrix}$

Combination of

- scaling
- rotation
- shear
- translation

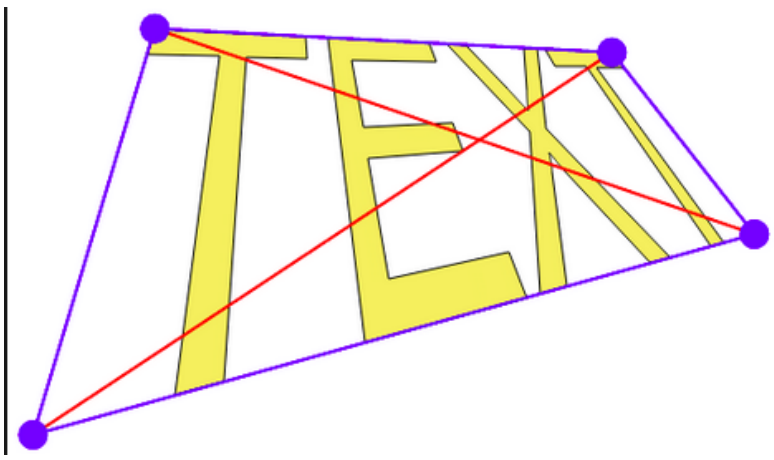
Inverse is also affine

Parallel lines remain parallel

Can concert any to any triangle

Needs 3 points to be found

Geometric transformations



Perspective tranformation:

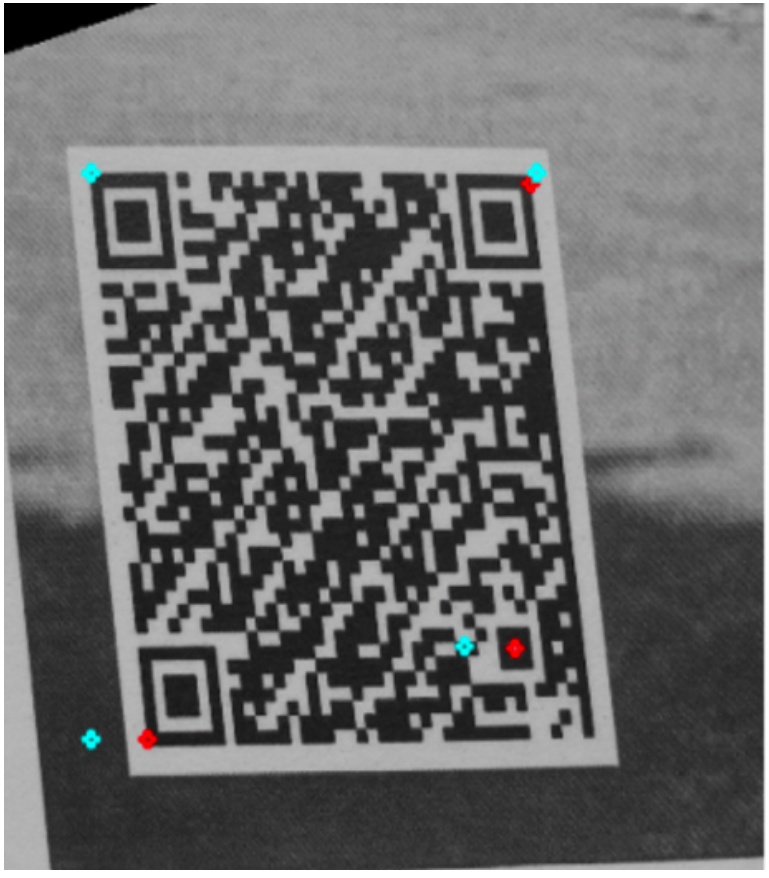
$$T(x,y) = \left(\frac{ax+by+c}{gx+hy+1}\right)$$
$$\frac{dx+ey+f}{gx+hy+1}$$

Inverse is also perspective

Can convert any to any quadrangle

Needs 4 points to be found

Example: QR codes scan



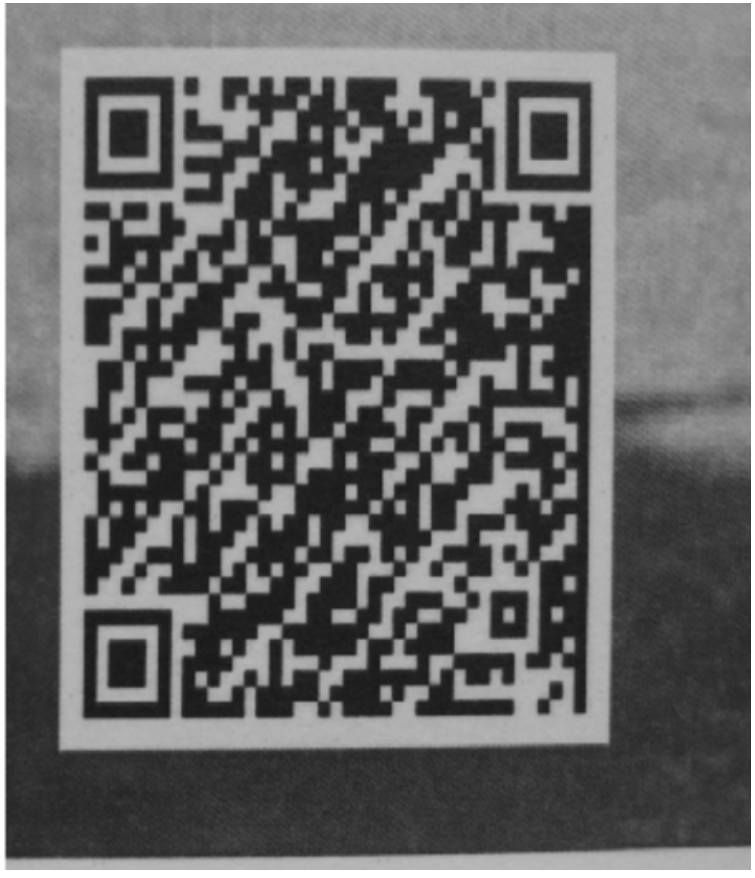
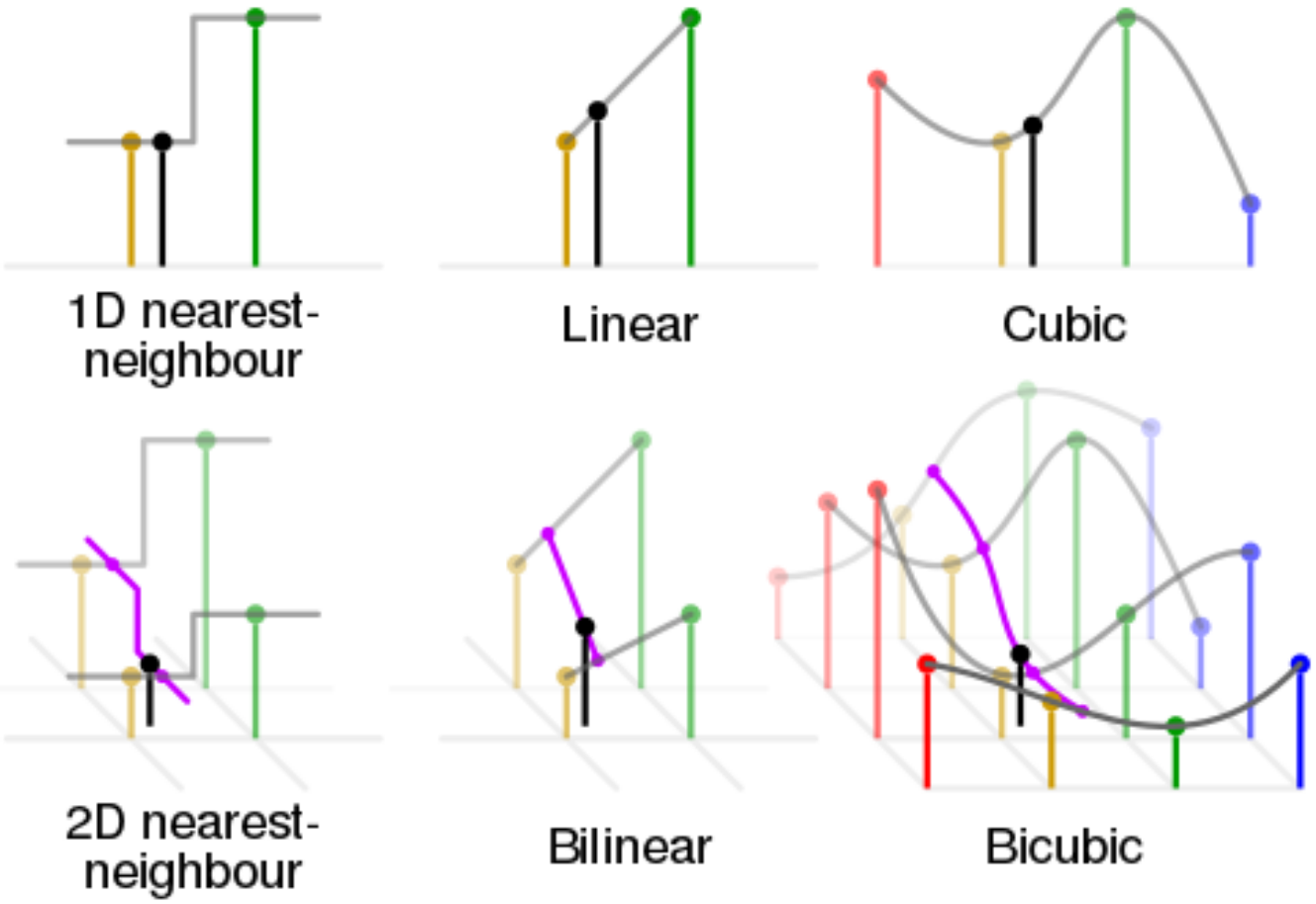


Image interpolations



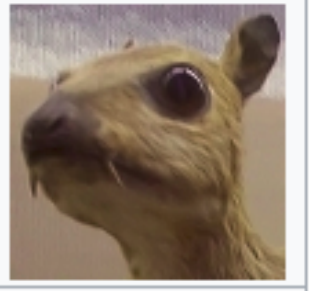
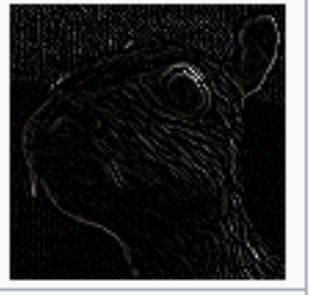
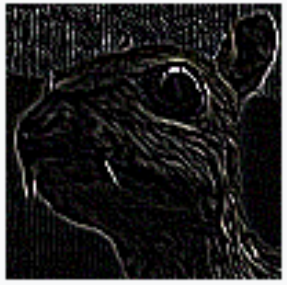
2D convolution

Image: $I[i,j]$,
Convolution filter: $F[i,j]$
Convolution:

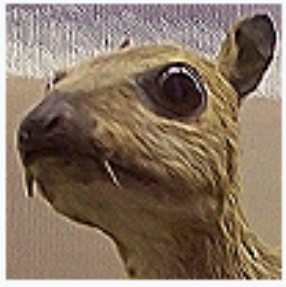
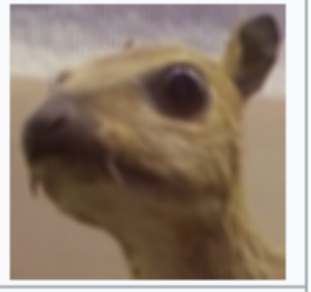
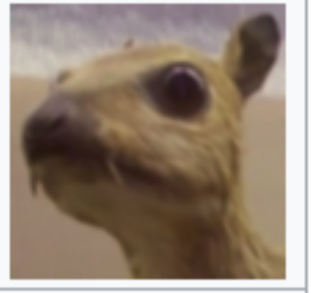
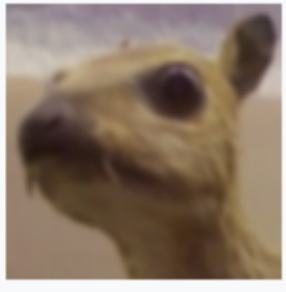
$$I[i,j] * F[i,j] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} I[n_1][n_2] \cdot F[i-n_1][j-n_2]$$

Properties: translation equivariant

Example kernels 1

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

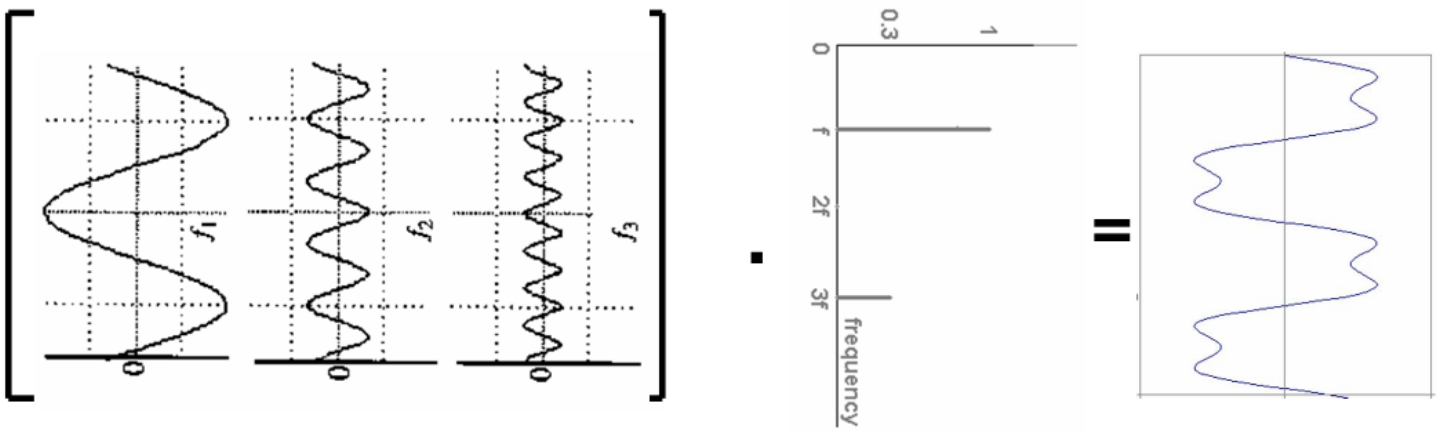
Example kernels 2

Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Fourier transform

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux}\, dx,$$

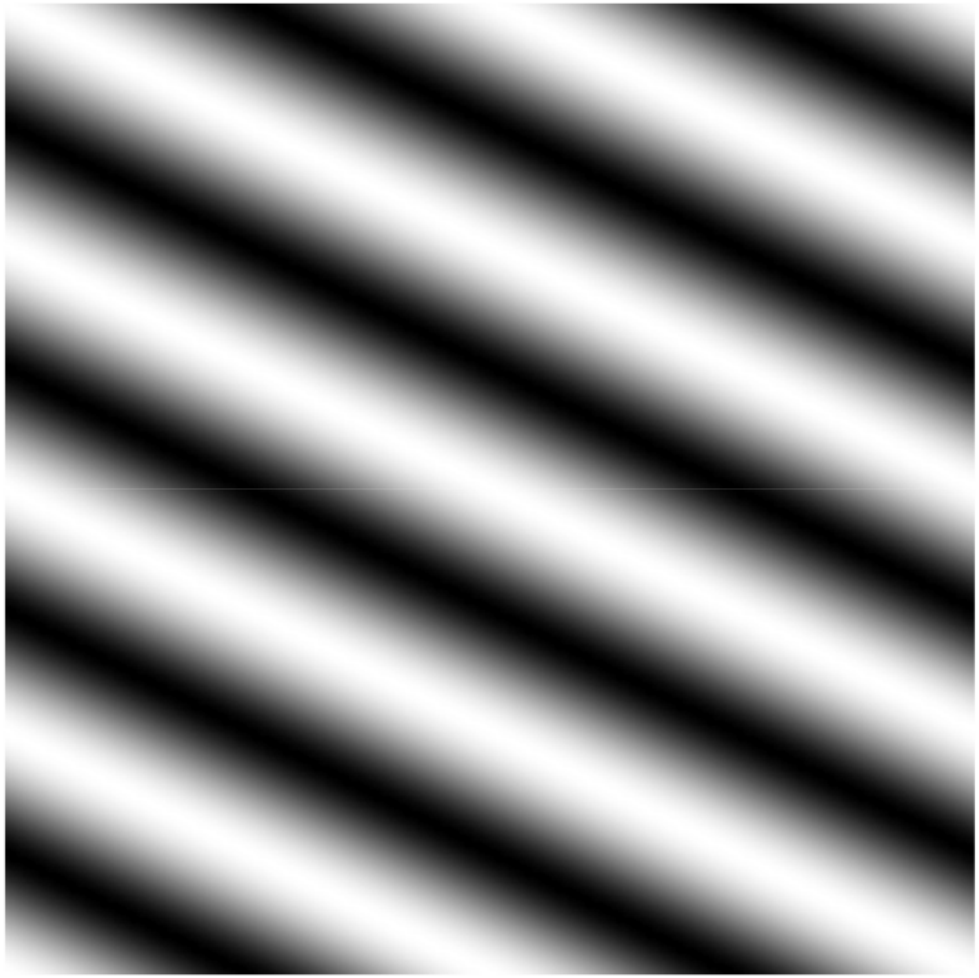
$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux}\, du$$

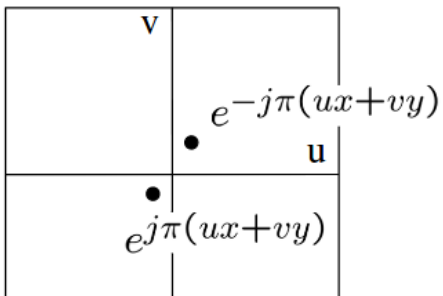


2D Fourier

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}\,dx\,dy,$$
$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}\,du\,dv$$

To get some sense of what basis elements look like, we plot a basis element --- or rather, its real part --- as a function of x,y for some fixed u, v. We get a function that is constant when (ux+vy) is constant. The magnitude of the vector (u, v) gives a frequency, and its direction gives an orientation. The function is a sinusoid with this frequency along the direction, and constant perpendicular to the direction.

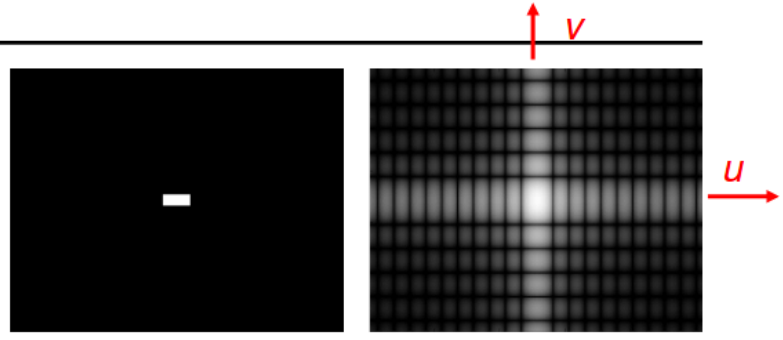




Fourier example 1

FT pair example 1

rectangle centred at origin
with sides of length X and Y



$$F(u,v) = \int\int f(x,y)e^{-j2\pi(ux+vy)}dxdy,$$

$$= \int_{-X/2}^{X/2} e^{-j2\pi ux}dx \int_{-Y/2}^{Y/2} e^{-j2\pi vy}dy,$$

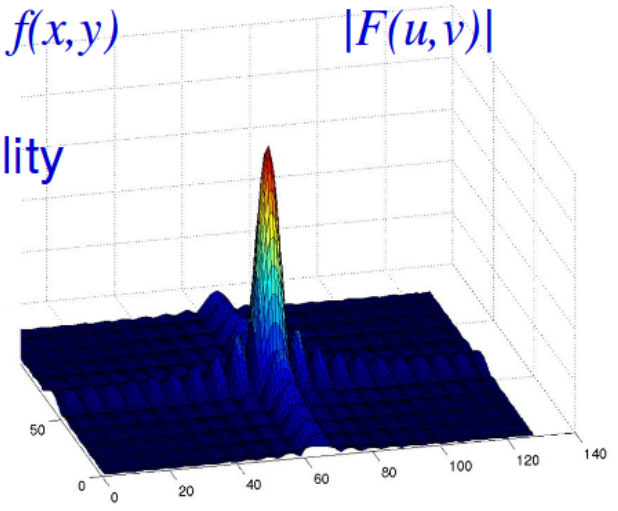
$$= \left[\frac{e^{-j2\pi ux}}{-j2\pi u}\right]_{-X/2}^{X/2} \left[\frac{e^{-j2\pi vy}}{-j2\pi v}\right]_{-Y/2}^{Y/2},$$

$$= \frac{1}{-j2\pi u}\left[e^{-juX} - e^{juX}\right]\frac{1}{-j2\pi v}\left[e^{-jvY} - e^{jvY}\right],$$

$$= XY\left[\frac{sin(\pi Xu)}{\pi Xu}\right]\left[\frac{sin(2\pi Yv)}{\pi Yv}\right]$$

$$= XY\mathrm{sinc}(\pi \mathrm{Xu})\mathrm{sinc}(\pi \mathrm{Yv}).$$

separability



$|F(u,v)|$

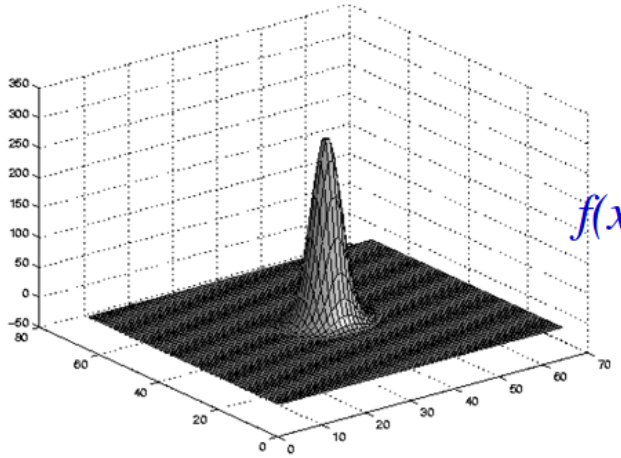
Fourier example 2

FT pair example 2

Gaussian centred on origin

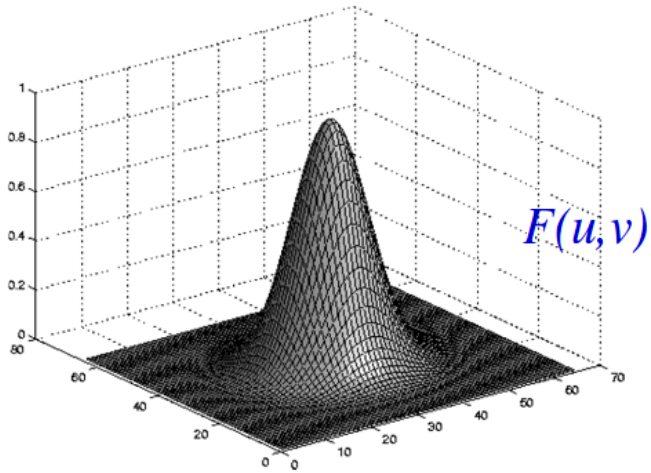
$$f(r) = \frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2}$$

where $r^2 = x^2 + y^2$.



$$F(u,v) = F(\rho) = e^{-2\pi^2\rho^2\sigma^2}$$

where $\rho^2 = u^2 + v^2$.



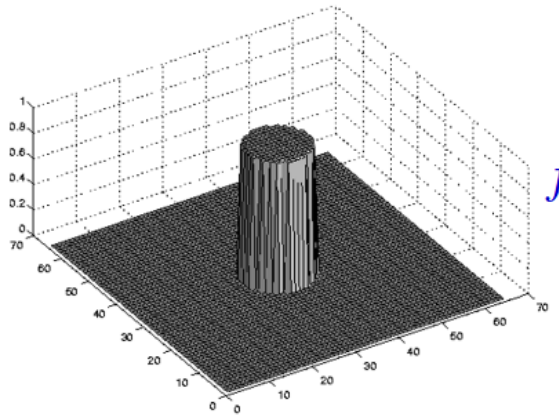
- FT of a Gaussian is a Gaussian
- Note inverse scale relation

Fourier example 3

FT pair example 3

Circular disk unit height and radius *a* centred on origin

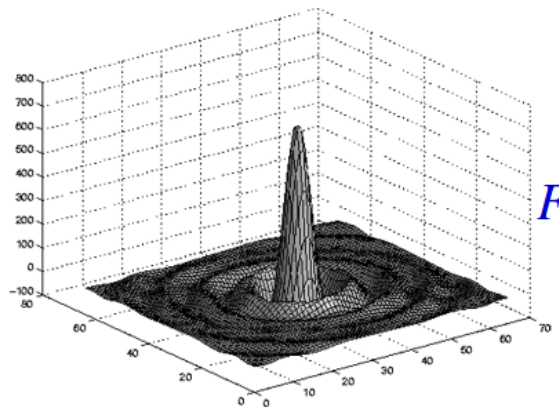
$$f(x,y) = \begin{cases} 1, & |r| < a, \\ 0, & |r| \geq a. \end{cases}$$



f(x,y)

$$F(u,v) = F(\rho) = aJ_1(\pi a\rho)/\rho$$

where $J_1(x)$ is a Bessel function.



F(u,v)

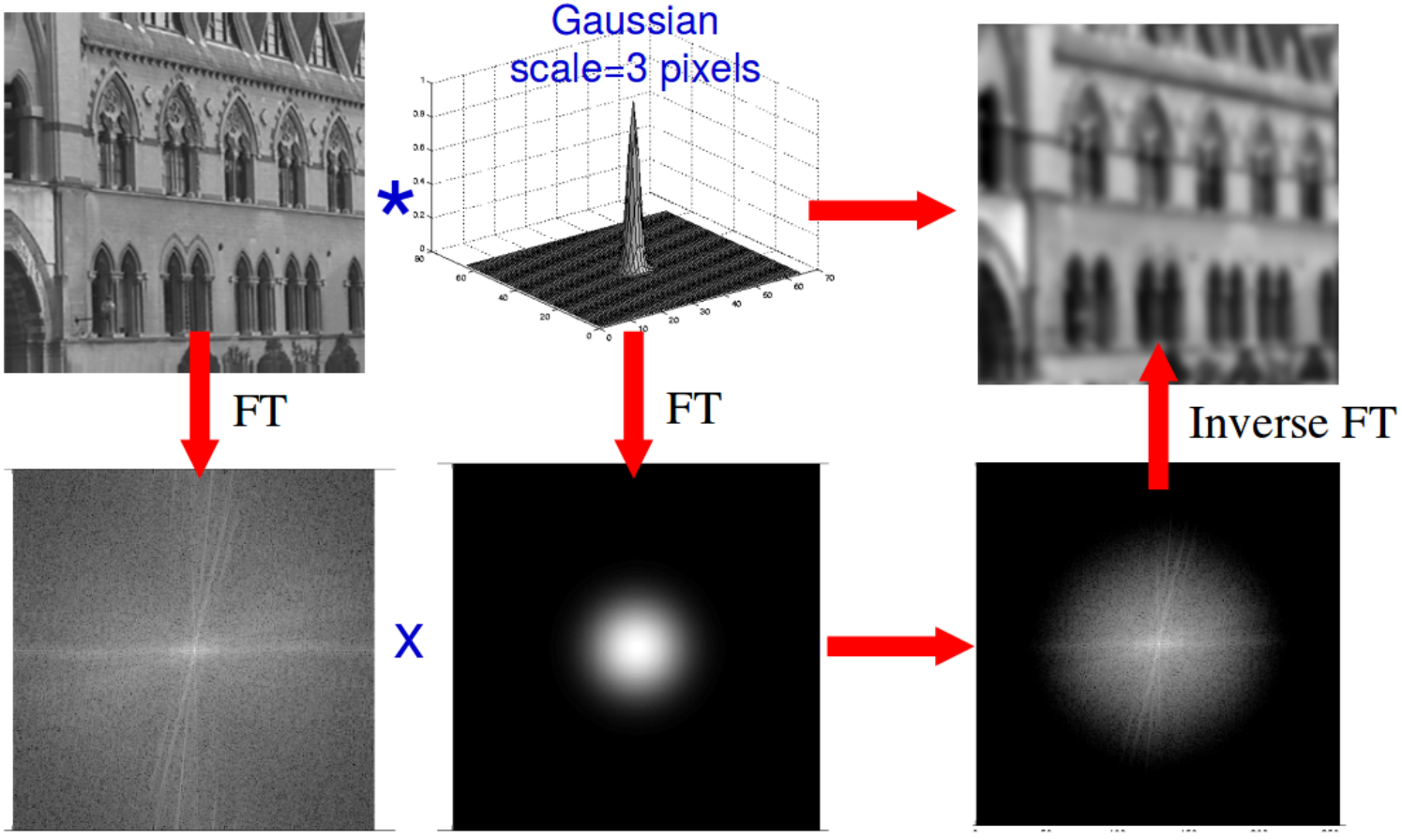
- rotational symmetry
- a '2D' version of a sinc

Phase importance

Fourier properties

- Fourier is a linear transform
- Fourier of convolution is equal to product of Fourier transforms
- Fourier of Gaussian is Gaussian
- FFT (Fast Fourier transform) computes it in $nlog(n)$

Example: image restoration



Blurring acts as a low pass filter and attenuates higher spatial frequencies

Problems with direct deblur

Deblurring with an inverse filter

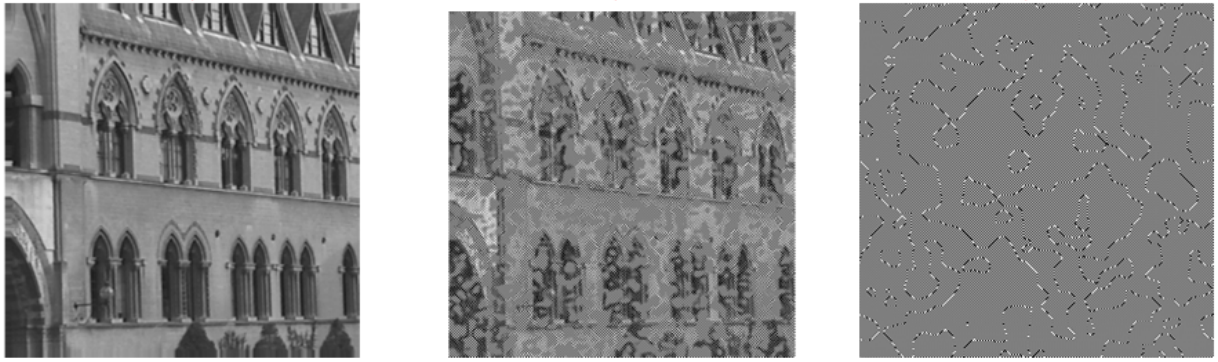
noise σ = 0.3 grey levels

$\hat{F}(u,v) = G(u,v) / H(u,v)$

blur σ = 0.5 pixels | blur σ = 1.0pixels | blur σ = 1.5pixels

$g(x,y)$

$\hat{f}(x,y)$



The Wiener filter

$$\widehat{F}(u,v) = W(u,v)\, G(u,v)$$

$$W(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K(u,v)}$$

where

$$K(u,v) = S_\eta(u,v)/S_f(u,v)$$

$S_f(u,v) = |F(u,v)|^2$ power spectral density of $f(x,y)$

$S_\eta(u,v) = |N(u,v)|^2$ power spectral density of $\eta(x,y)$

Wiener deblur

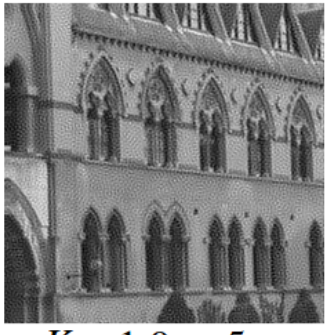
Example 1: Focus deblurring with a Wiener filter

blur σ = 1.5 pixels
noise σ = 0.3 grey levels

$\hat{F}(u,v) = W(u,v)\, G(u,v)$

$$W(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + K(u,v)}$$

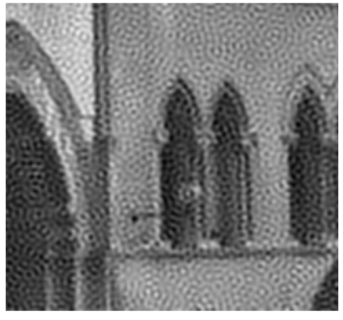
$g(x,y)$ | $\hat{f}(x,y)$



K = 1.0 e -5

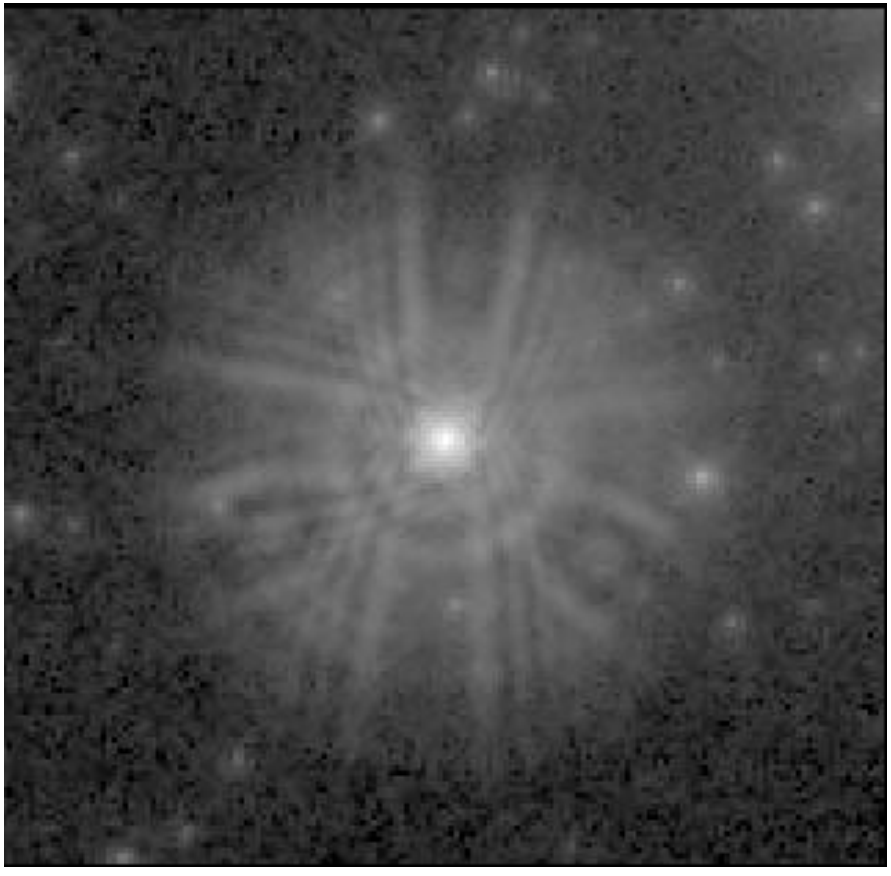
K = 1.0 e -3

K = 1.0 e -1



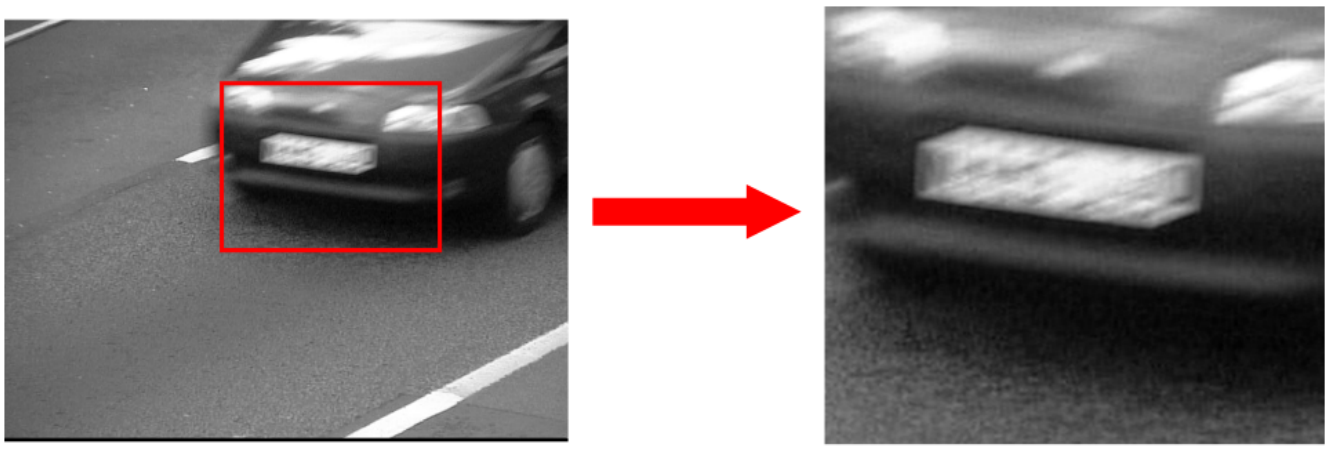
Point spread function

The point spread function (PSF) describes the response of an imaging system to a point source or point object.
Hubble telescope PSF:



Example: reading number plates

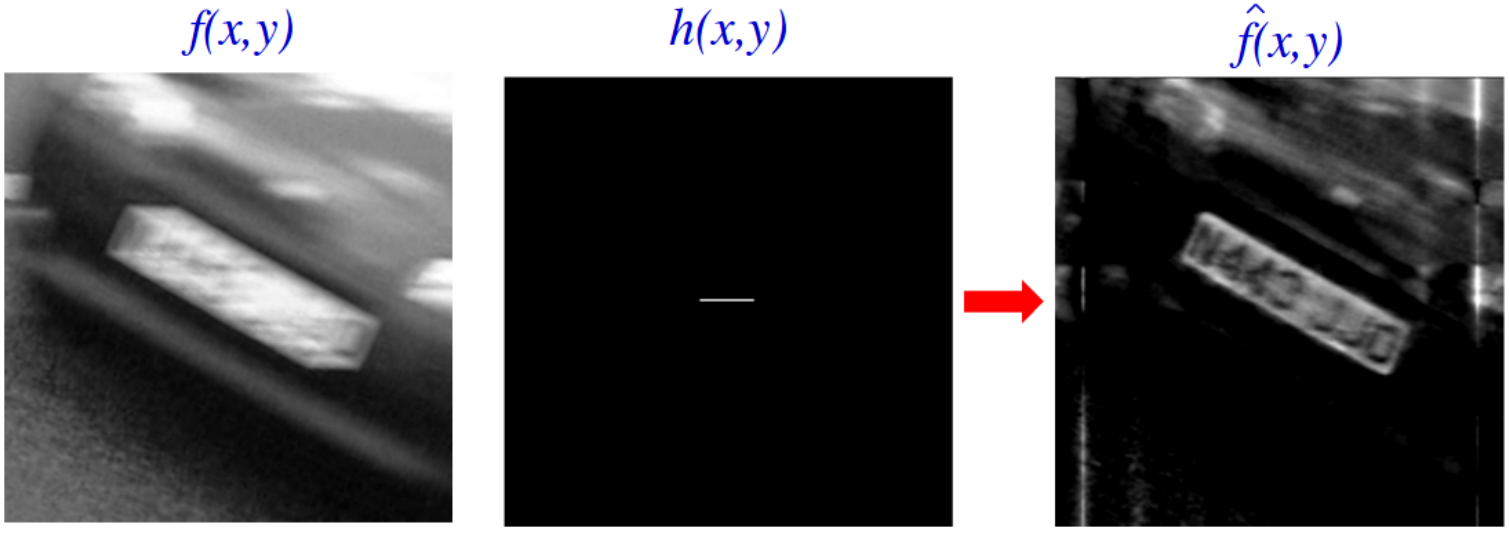
Application: Reading number plates



Algorithm

1. Rotate image so that blur is horizontal
2. Estimate length of blur
3. Construct a bar modelling the convolution
4. Compute and apply a Wiener filter
5. Optimize over values of K

Example: reading number plates



blur = 30 pixels

Optimization for deblurring

- Estimate $f(x,y)$ by optimizing a cost function:

observed image

generated image

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} \underbrace{(\mathbf{g} - \mathbf{A}\mathbf{f})^2}_{\text{Likelihood/ loss function}} + \underbrace{\lambda p(\mathbf{f})}_{\text{prior/ regularization}}$$

Example

$$p(f) = (\nabla \mathbf{f})^2$$

to suppress high frequency noise

Blind deblurring

So far we have assumed that we know the generative model, e.g.

$$\mathbf{g} = \mathbf{A}(\mathbf{h})\,\mathbf{f}$$
$$\mathbf{G} = \mathbf{H}\,\mathbf{F}$$

i.e. that $h(x,y)$ is known, so that given the observed image $g(x,y)$, then the original image $f(x,y)$ can be estimated (restored)

Consider if only the observed image $g(x,y)$ is known.
This is the problem of blind estimation.

Blind deblurring

- Estimate $f(x,y)$ and $h(x,y)$ by optimizing a cost function:

observed image

generated image

$$\min_{\mathbf{f},\mathbf{h}} (\mathbf{g} - \mathbf{A}(\mathbf{h})\,\mathbf{f})^2 + \lambda p_f(\mathbf{f}) + \mu p_h(\mathbf{h})$$

Likelihood/ loss function

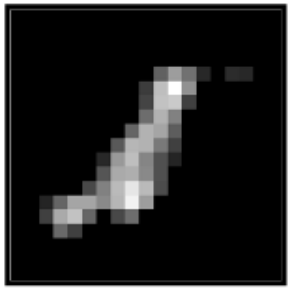
image prior

blur prior

Blind deblurring

blurred image

estimated blur filter



restored image

Blind deblurring

Sobel edge detector

Gradient in x an y direction:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Decomposition:

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

$$\mathbf{\Theta} = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Sobel example

Sobel gradient

LoG: Laplacian of Gaussian

- To reduce the noise effect, the image is first smoothed.
- When the filter chosen is a Gaussian, we call it the LoG edge detector.

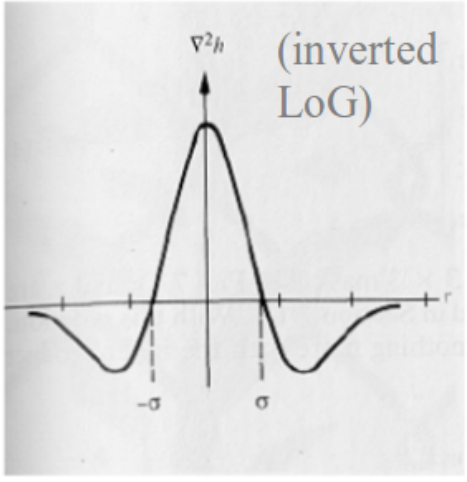
$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

σ controls smoothing

- It can be shown that:

$$\nabla^2[f(x,y) * G(x,y)] = \nabla^2 G(x,y) * f(x,y)$$

$$\nabla^2 G(x,y) = \left(\frac{r^2 - 2\sigma^2}{\sigma^4}\right) e^{-r^2/2\sigma^2}, \quad (r^2 = x^2 + y^2)$$



LoG: Laplacian of Gaussian

- The 2-D Laplacian of Gaussian (LoG) function centered on zero and with Gaussian standard deviation σ has the form:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ is the standard deviation

- The amount of smoothing can be controlled by varying the value of the standard deviation.

LoG: Laplacian of Gaussian

(inverted LoG)

5 × 5 Laplacian of Gaussian mask

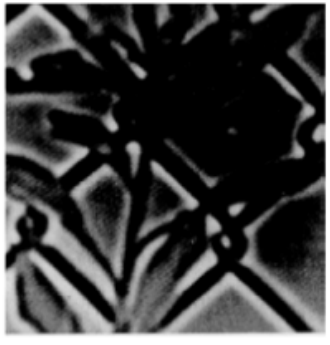
0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

(inverted LoG)

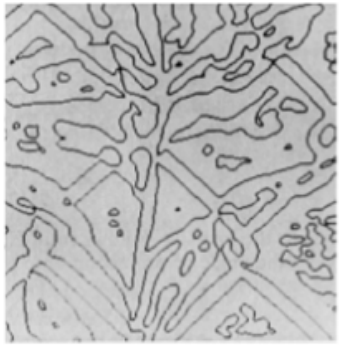
17 × 17 Laplacian of Gaussian mask

0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	4	12	21	24	21	12	4	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0

filtering



zero-crossings



Edge detection

Try out more edge detectors:

- Canny
- Deriche
- Prewitt
- Roberts cross