Computer Vision

Sampling, Quantization and Image Enhancement

Part I : Sampling & quantization

1. Discretization of continuous signals
2. Signal representation in the frequency domain
3. Effects of sampling and quantization

Part II : Image enhancement

1. Noise suppression
2. De-blurring
3. Contrast enhancement

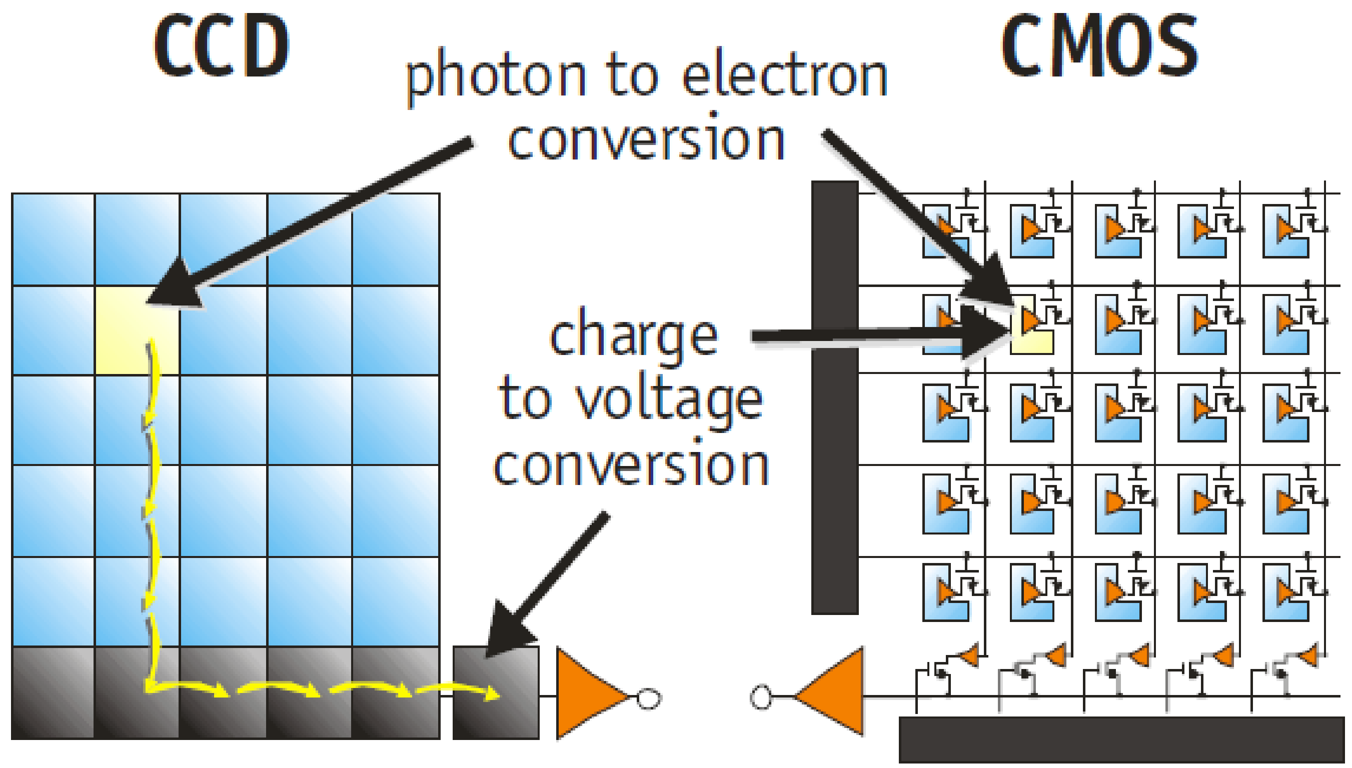
Part I : Sampling & quantization

1. Discretization of continuous signals
2. Signal representation in the frequency domain
3. Effects of sampling and quantization

Part II : Image enhancement

1. Noise suppression
2. De-blurring
3. Contrast enhancement

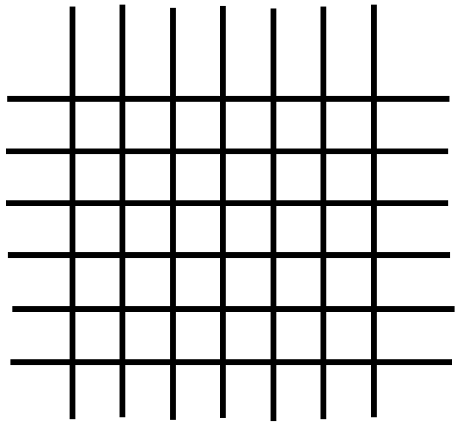
Recall – photons to digital signal

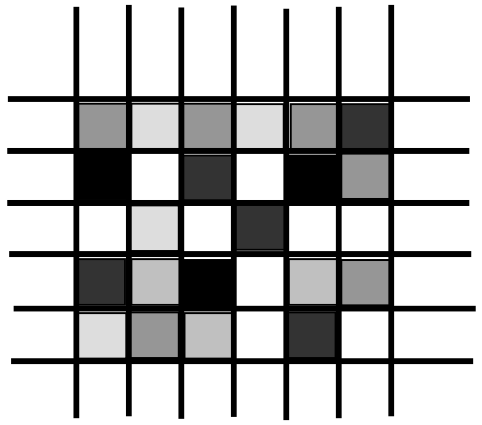


- CCD : Charge-coupled device
- CMOS : Complementary Metal Oxide Semiconductor
- We will study the effects of the digitization / discretization.

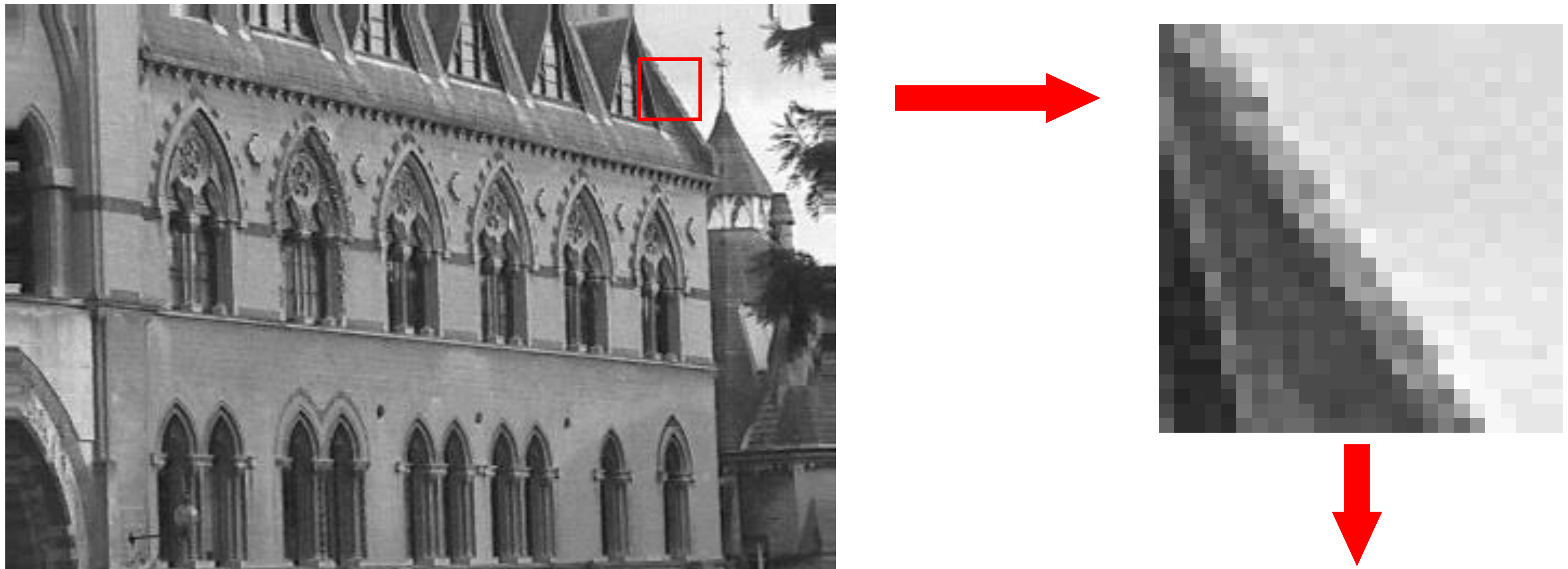
Discretization / Digitization

- Necessary computer to process an image
- Includes two parts
  1. Sampling – spatial discretization, creates “pixels”
  2. Quantization – intensity discretization, creates “grey levels”





Sampling & Quantization

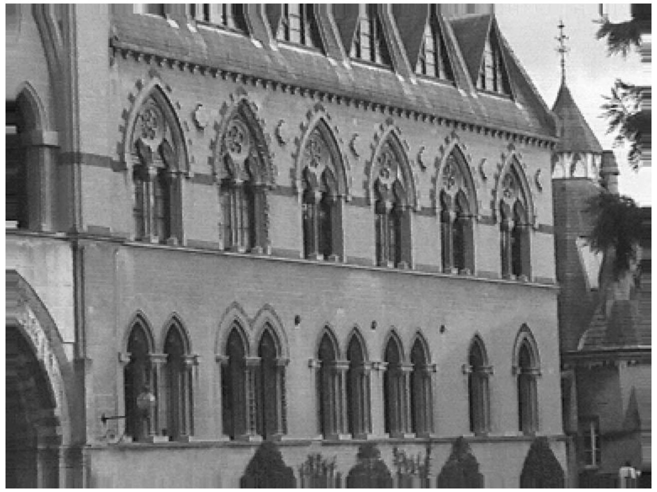


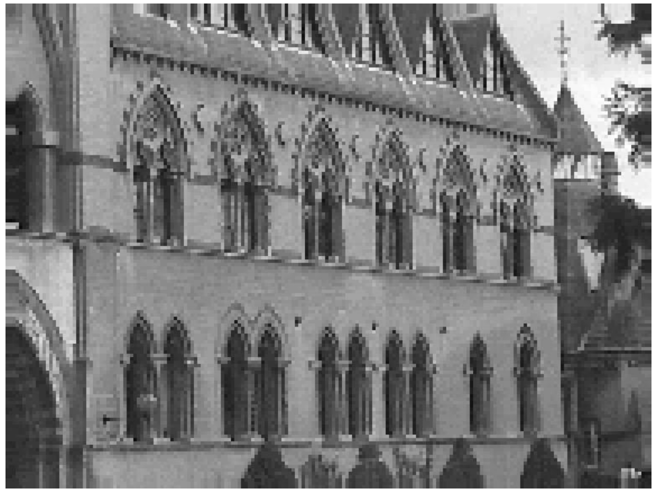
84	133	226	212	218	218	222	212	218	222	226	218
75	156	177	218	212	218	218	218	218	222	218	218
96	84	133	203	218	218	218	222	212	218	222	218
123	75	111	156	212	218	212	212	218	218	218	226
93	75	71	133	185	231	226	226	222	212	218	218
51	75	75	75	156	206	218	218	218	222	212	222
44	110	75	65	143	194	231	218	218	218	218	218
52	123	69	84	60	156	199	231	231	222	226	226
52	75	84	81	65	69	150	231	231	226	231	231
36	36	84	93	84	71	156	160	240	240	231	231
36	40	113	75	69	75	71	133	194	240	240	240
52	52	105	85	69	75	75	123	111	222	231	231
69	44	69	93	81	75	75	69	150	177	247	240
73	44	40	96	101	75	75	75	84	133	231	240

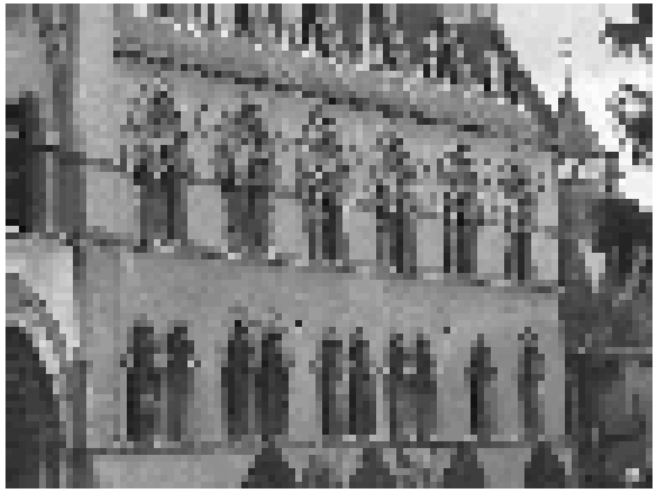
Creating finite number of points in space in a grid, i.e. pixels, and intensity value in each pixel is represented with finite number of bits in the computer.

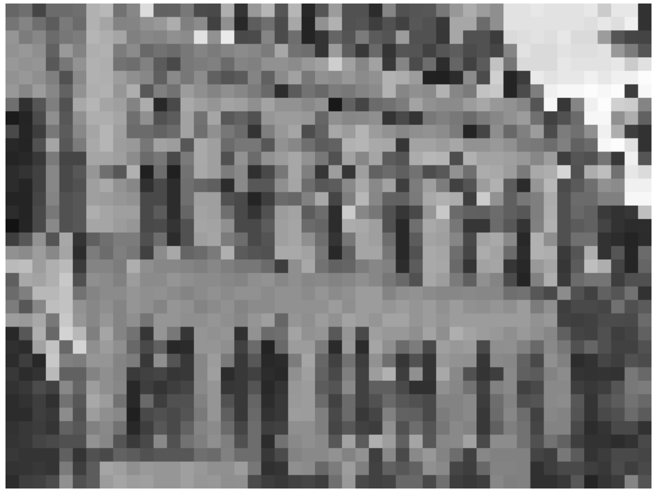
The original scene is continuous in space and intensity value

Example of sampling

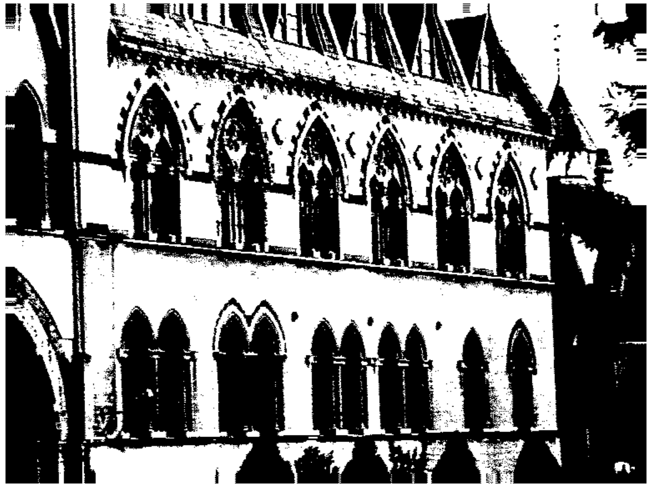

384 x 288 pixels

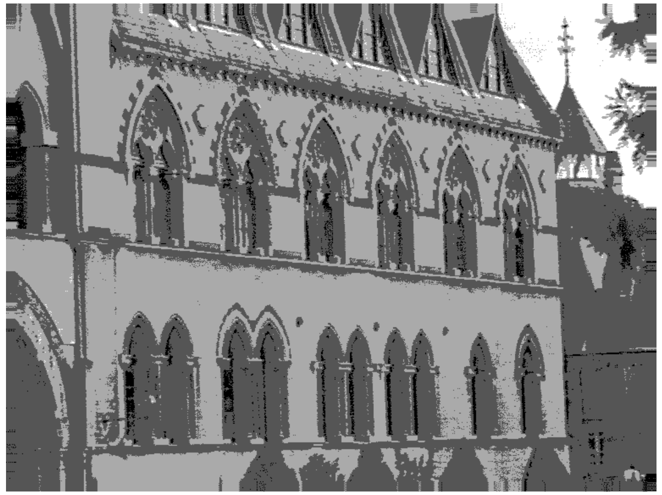

192 x 144 pixels

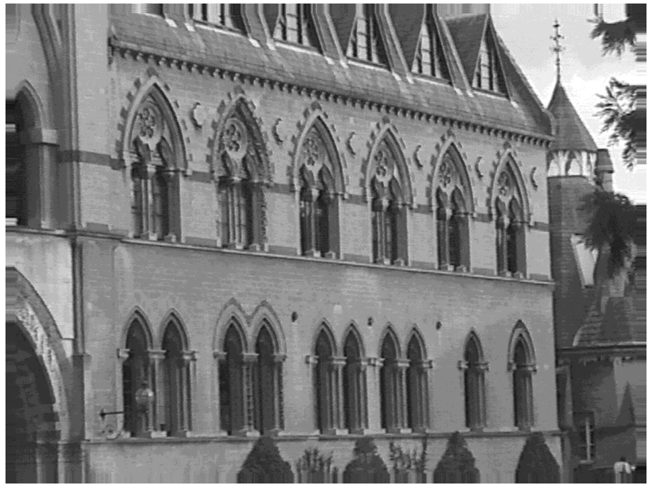

92 x 72 pixels


48 x 36 pixels

Example of quantization


2 levels - binary


4 levels


8 levels

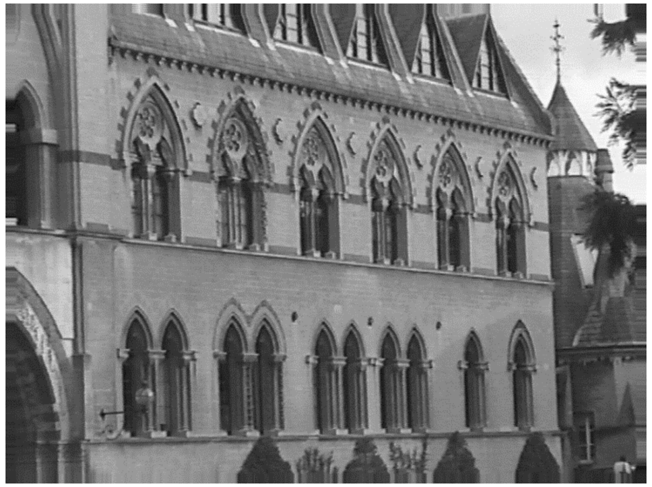

256 levels – 1 byte

Image distortion through sampling

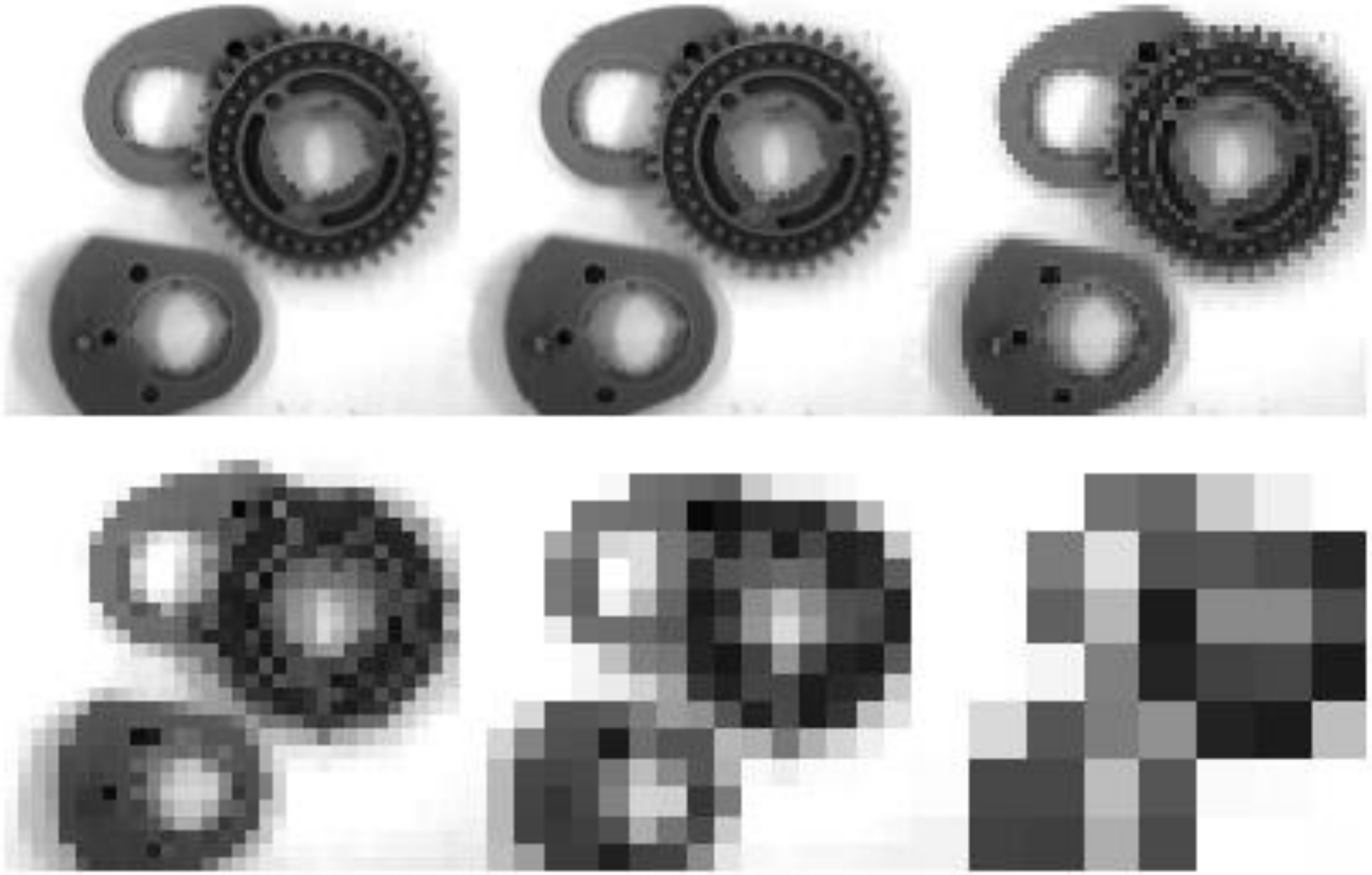
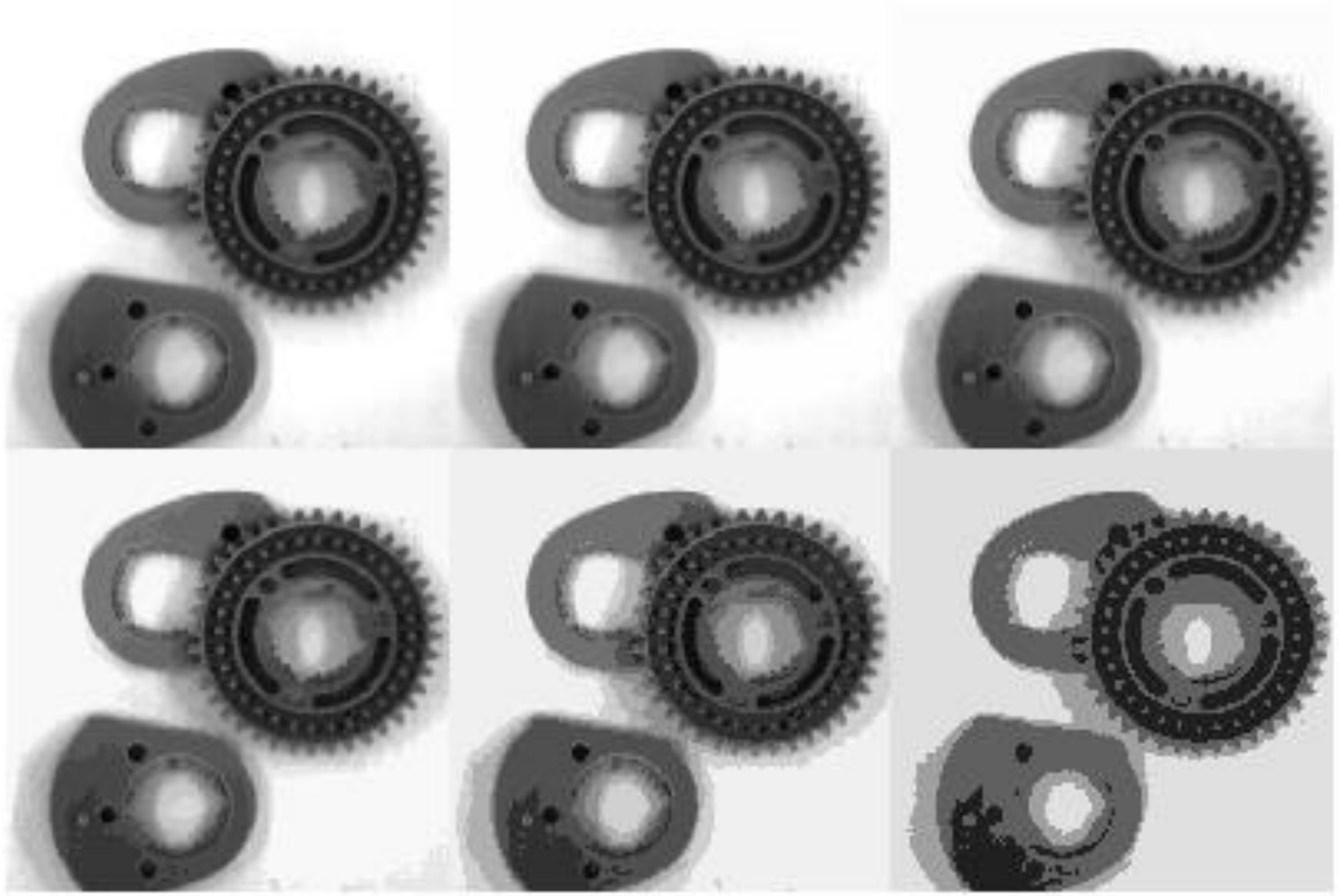


Image distortion through quantization

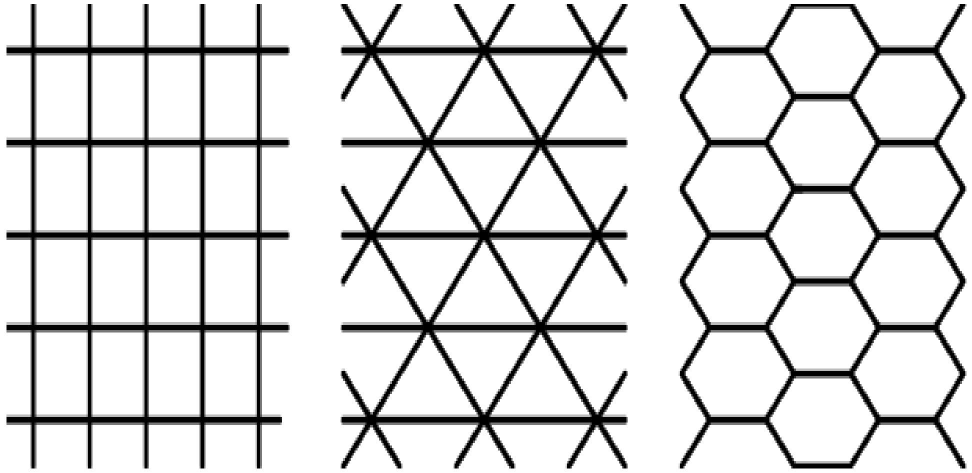


Remarks

1. Binary images – 1-bit quantization – are useful in industrial applications. They usually have control over imaging conditions, e.g. background color, lighting conditions, …
2. Non-uniform sampling and/or quantization is sometimes used for specialized applications
   a. Fine sampling to capture details
   b. Fine quantization for homogeneous regions
3. Different sampling strategies than square grids exist

Different sampling schemes

- You need regular, image covering tessellation
- There are 11 polygons to achieve this. If you want to use the same polygon across the image then only 3, shown on the right.
- Rectangular (square) is the most popular
- Hexagonal has advantages (more isotropic, no connectivity ambiguities). Similar structure is seen in the retina of various species.

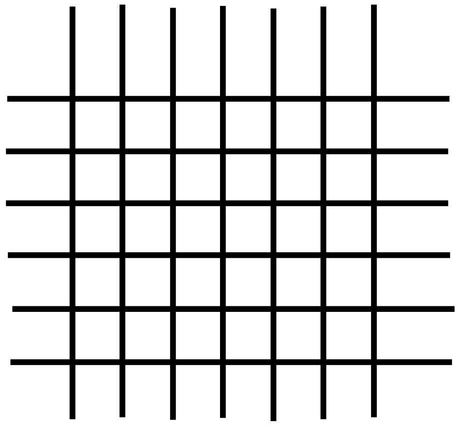


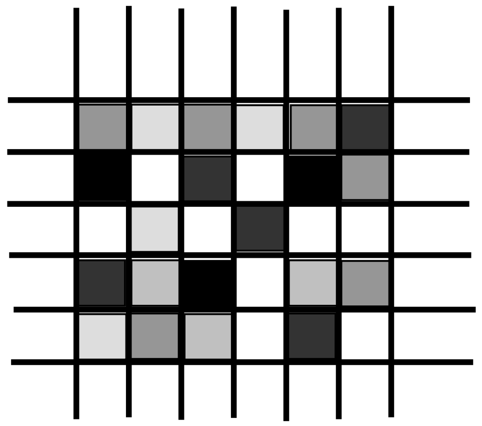
Computer Vision

Part I - Intro

Discretization / Digitization

- Necessary computer to process an image
- Includes two parts
    1. Sampling – spatial discretization, creates “pixels”
    2. Quantization – intensity discretization, creates “grey levels”





A model for sampling

- There are two essential steps

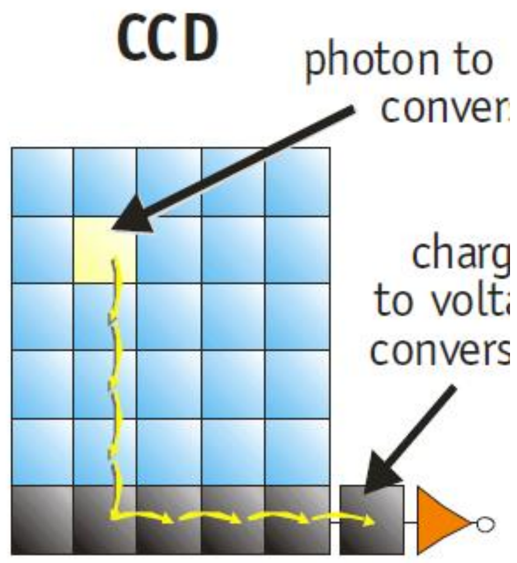
1. Integrate brightness over a cell window
   Leads to blurring type degradation

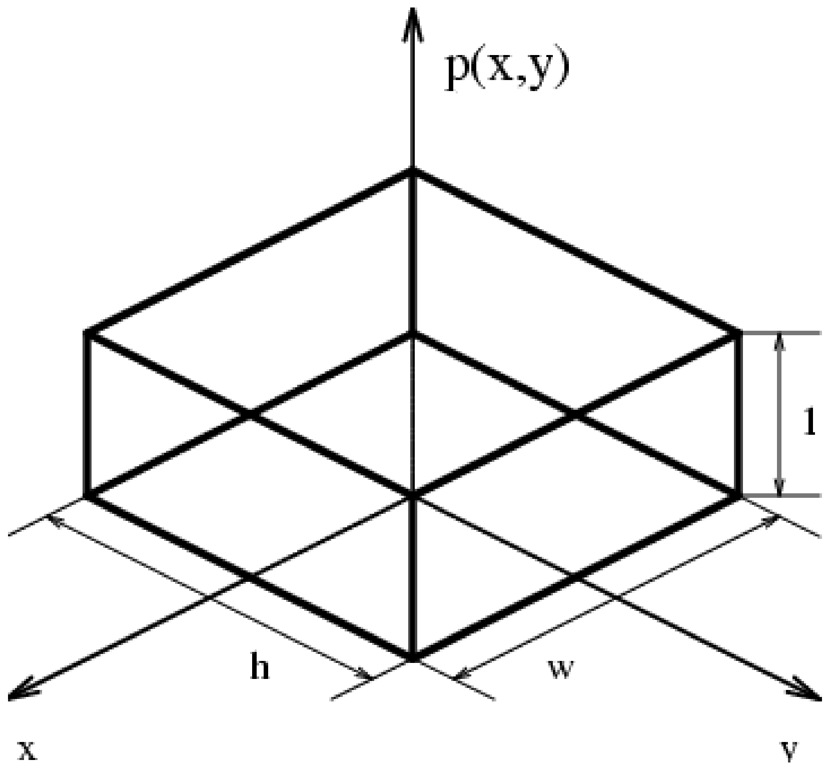
2. Read out values only at the pixel centers
   Leads to aliasing and leakage, frequency domain issues

Computer Vision

Part I – Intro
Part I – Sampling

STEP I: integrating over a cell window



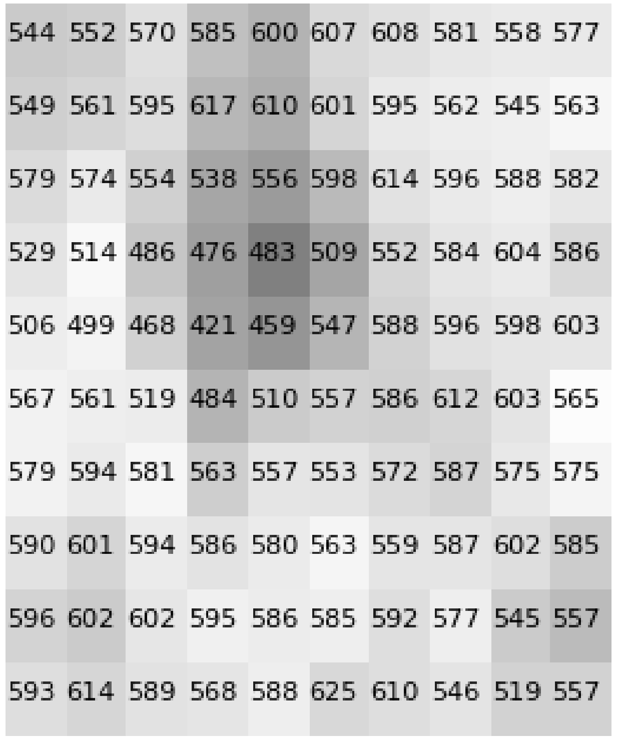


$$o(x',y') = \int\int i(x,y)p(x-x',y-y')dxdy$$

This is a *convolution*: $i(x,y) * p(-x,-y)$

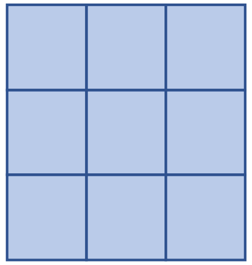
Convolution

- While the previous convolution was in continuous domain, we’ll look at discrete convolution to get an intuition.



544	552	570	585	600	607	608	581	558	577
549	561	595	617	610	601	595	562	545	563
579	574	554	538	556	598	614	596	588	582
529	514	486	476	483	509	552	584	604	586
506	499	468	421	459	547	588	596	598	603
567	561	519	484	510	557	586	612	603	565
579	594	581	563	557	553	572	587	575	575
590	601	594	586	580	563	559	587	602	585
596	602	602	595	586	585	592	577	545	557
593	614	589	568	588	625	610	546	519	557

Image: x(i,j)

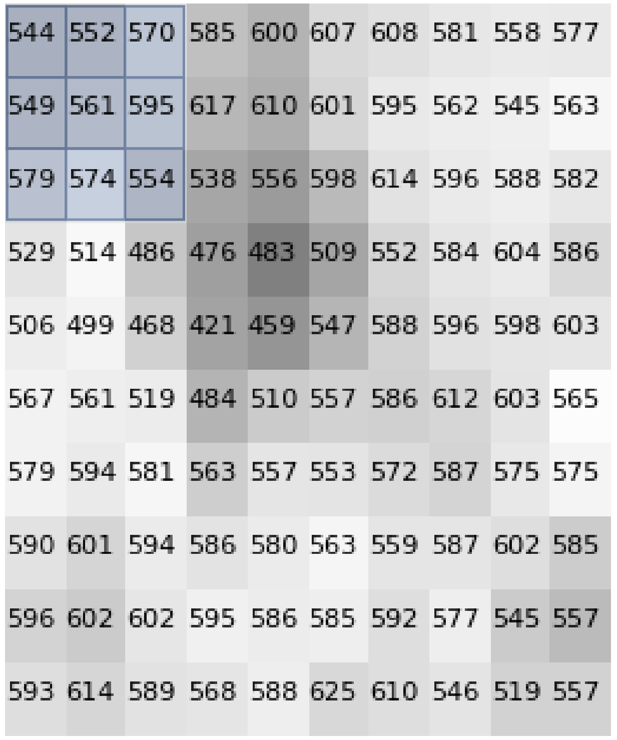


Convolutional kernel: w(i,j)

$$w * x$$

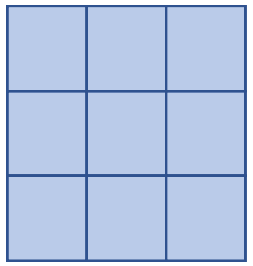
Convolution

- While the previous convolution was in continuous domain, we'll look at discrete convolution to get an intuition.



544	552	570	585	600	607	608	581	558	577
549	561	595	617	610	601	595	562	545	563
579	574	554	538	556	598	614	596	588	582
529	514	486	476	483	509	552	584	604	586
506	499	468	421	459	547	588	596	598	603
567	561	519	484	510	557	586	612	603	565
579	594	581	563	557	553	572	587	575	575
590	601	594	586	580	563	559	587	602	585
596	602	602	595	586	585	592	577	545	557
593	614	589	568	588	625	610	546	519	557

Image: x(i,j)



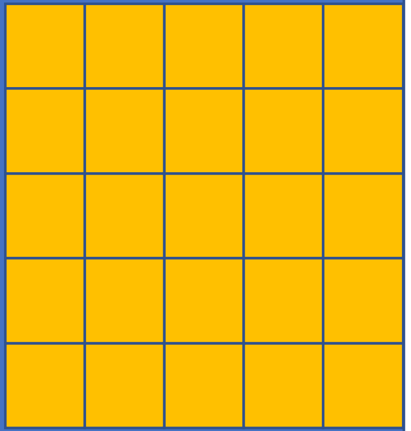
Convolutional kernel: w(i,j)

$$w * x$$

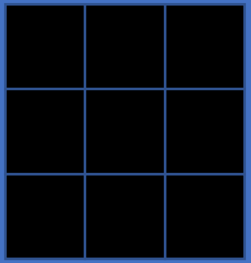
$$a_{ij} = \sum_{p} \sum_{q} x_{(i-p)(j-q)} w_{(p)(q)}$$

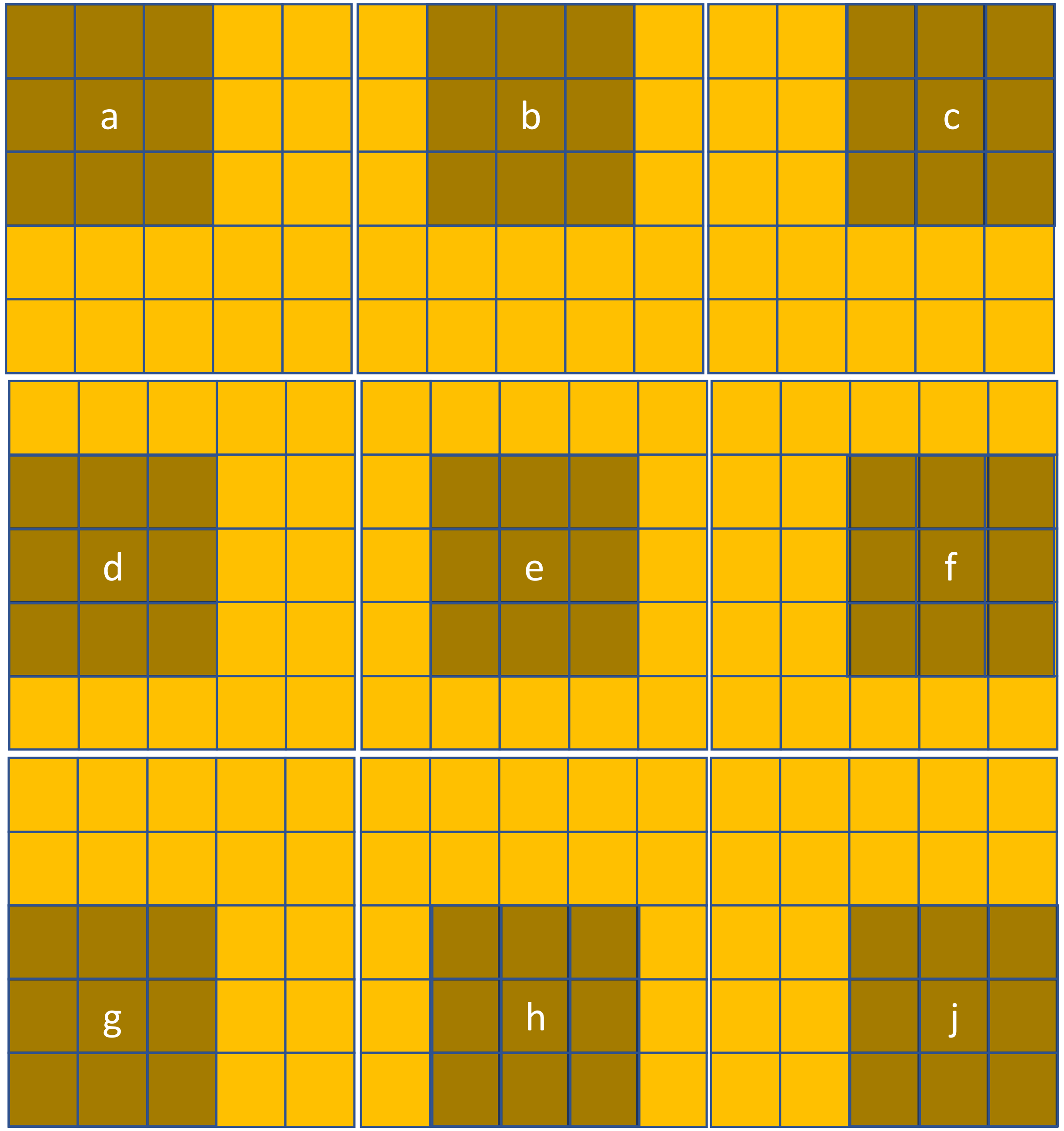
Computer Vision

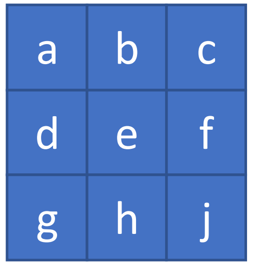
Part I – Intro
Part I – Sampling



Image


kernel



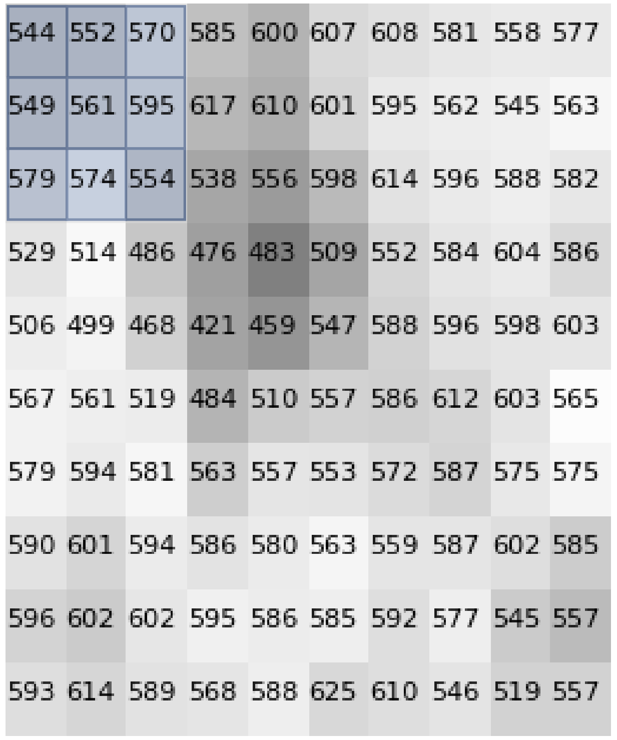


Computer Vision

Part I – Intro
Part I – Sampling

Convolution

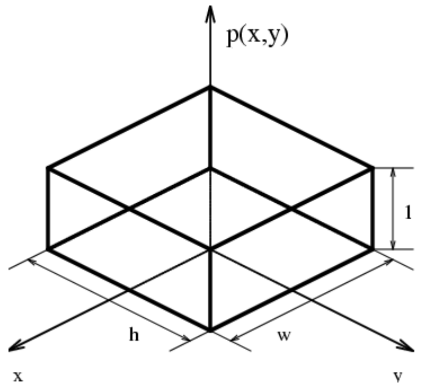
- While the previous convolution was in continuous domain, we'll look at discrete convolution to get an intuition.



544	552	570	585	600	607	608	581	558	577
549	561	595	617	610	601	595	562	545	563
579	574	554	538	556	598	614	596	588	582
529	514	486	476	483	509	552	584	604	586
506	499	468	421	459	547	588	596	598	603
567	561	519	484	510	557	586	612	603	565
579	594	581	563	557	553	572	587	575	575
590	601	594	586	580	563	559	587	602	585
596	602	602	595	586	585	592	577	545	557
593	614	589	568	588	625	610	546	519	557

$$o(x', y') = \int\int i(x, y)p(x - x', y - y')dxdy$$

Consider the continuous case as the limit where pixels are very small as well as the convolutional kernel is formed to correspond to that with many very small elements.



The kernel for this case is a rectangular box.

Properties of convolution

Commutative

$$f * g = g * f$$

Associative

$$\begin{aligned} k &= h * f \\ &= (h_1 * h_2) * f \\ &= h_1 * (h_2 * f) \end{aligned}$$

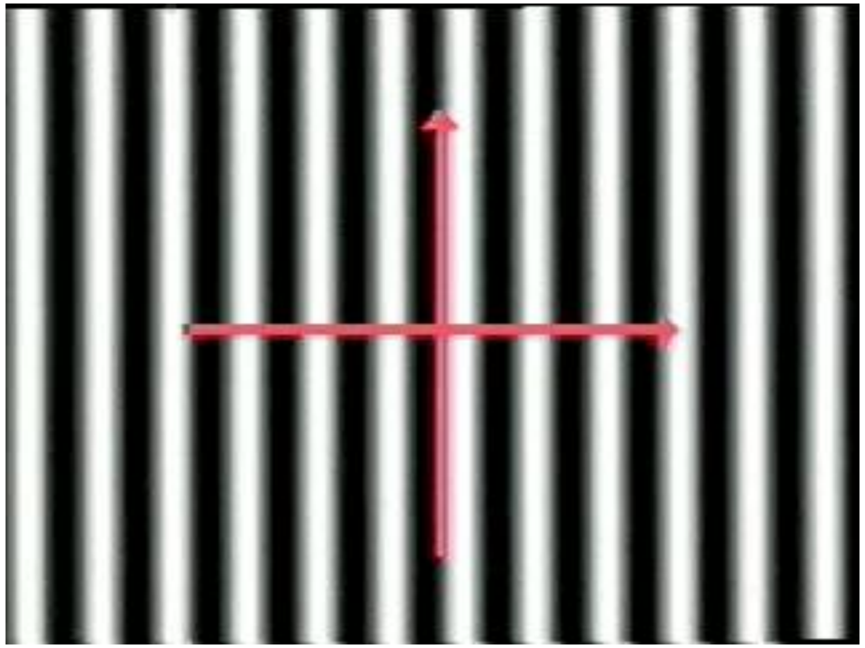
The Fourier Transform

- An important tool we should remind ourselves is the Fourier Transform (FT).
- This is crucial to understand the effects of STEPI as well as STEPII taken in sampling.
- Particularly, it is difficult to understand what type of information we lose when we convolve an image with a kernel with a box shape.
- Using FT, this becomes much easier!

Characterization of functions in the frequency domain

- Represent any signal as a linear combination of orthonormal basis functions

$$e^{i2\pi(ux+vy)} = \cos 2\pi(ux+vy) + i\sin 2\pi(ux+vy)$$



- Waves with wavelength orthogonal to the stripes of

$$\lambda = \frac{1}{\sqrt{u^2+v^2}}$$

The Fourier Transform: definition

Linear decomposition of functions in the new basis
Scaling factor for basis function (*u,v*)

The Fourier Transform

$$\mathcal{F}[f(x,y)] = F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-i2\pi(ux+vy)}dxdy$$

Reconstruction of the original function in the spatial domain: weighted sum of the basis functions

The Inverse Fourier Transform

$$\mathcal{F}^{-1}[F(u,v)] = f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{i2\pi(ux+vy)}dxdy$$

Fourier Coefficients

Complex function

$$F(u,v) = \underbrace{F_R(u,v)}_{\text{real part}} + \underbrace{iF_I(u,v)}_{\text{imaginary part}}$$

Magnitude

$$|F(u,v)| = \sqrt{F_R(u,v)^2 + F_I(u,v)^2}$$

Phase - angle

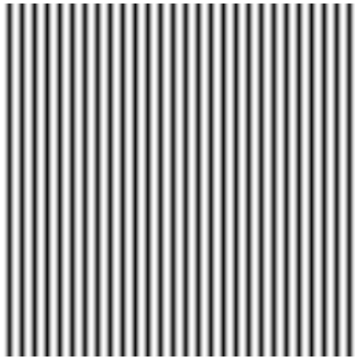
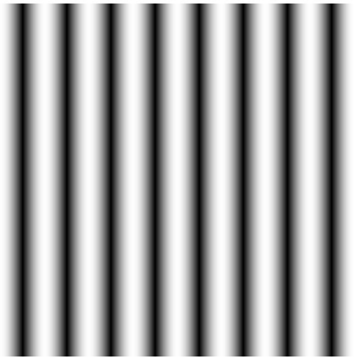
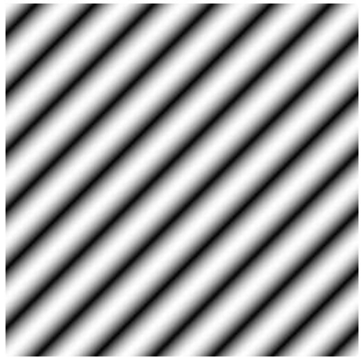
$$\phi(u,v) = \arg(F_R(u,v) + iF_I(u,v)) = \arctan \frac{F_I(u,v)}{F_R(u,v)}$$

Decomposition visually

$f(x, y) =$

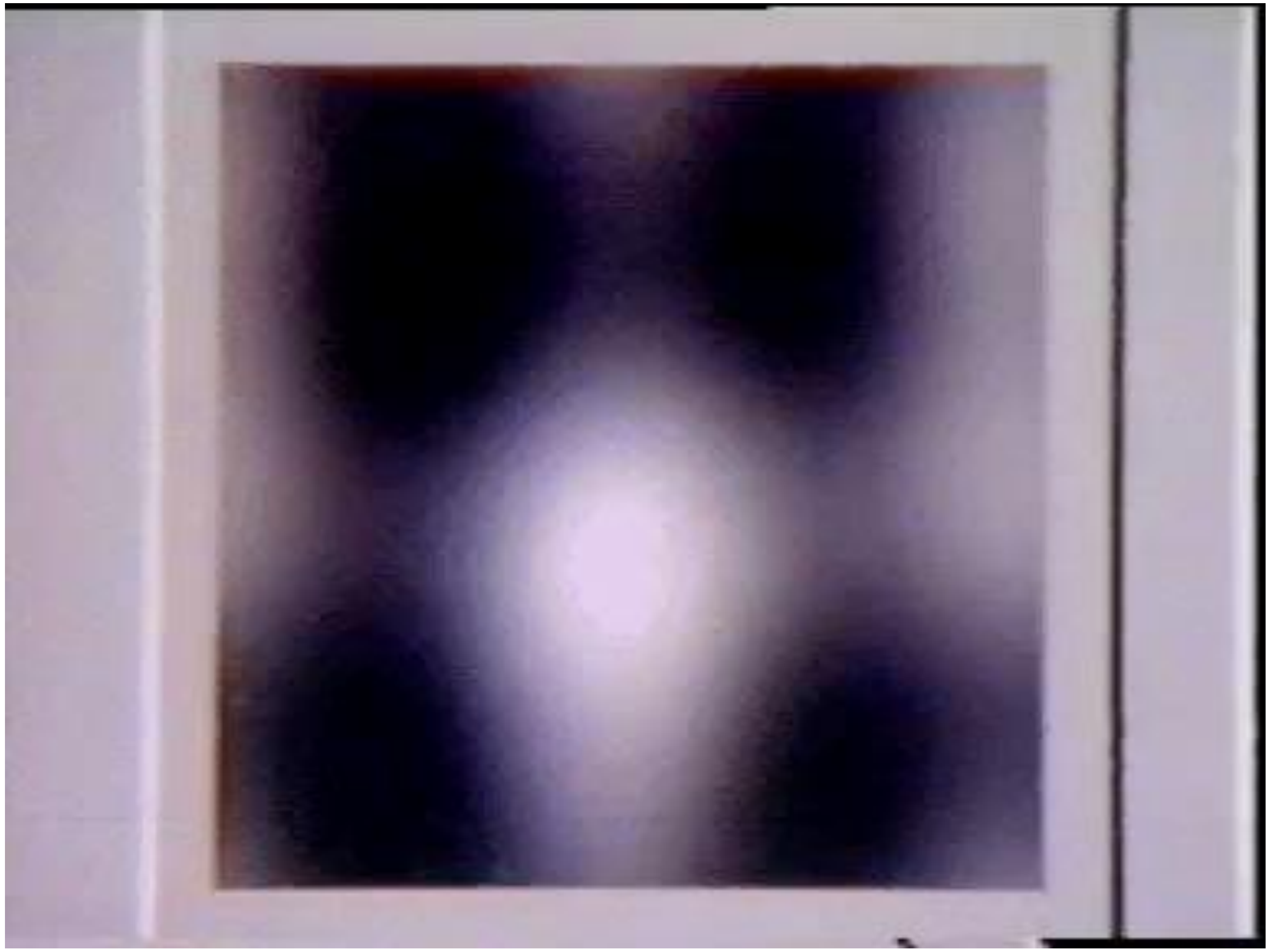
$= F(u, v) \quad + F(u', v') \quad + F(u'', v'') \cdots$

$\times \quad \times \quad \times$

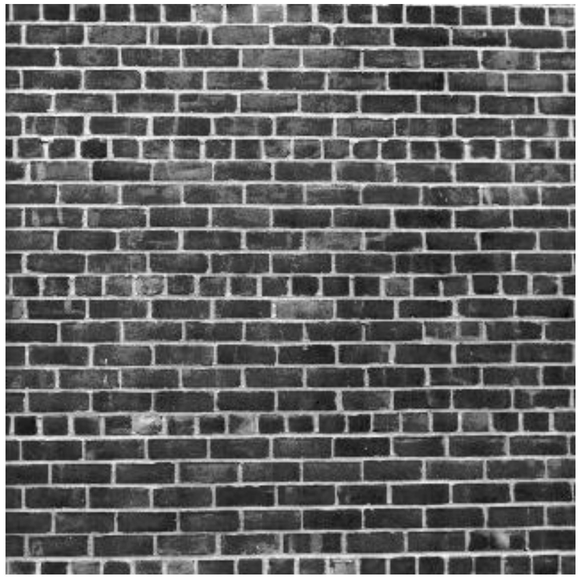
Example of FT

Effect of additional components

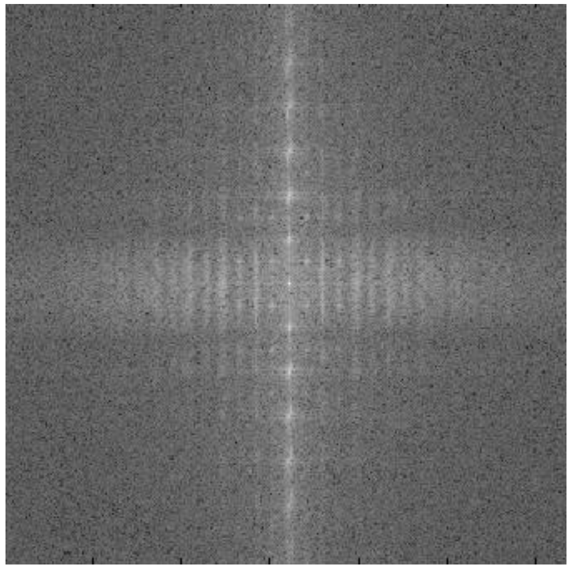


Importance of the magnitude in FT

- Image with periodic structure



$f(x,y)$



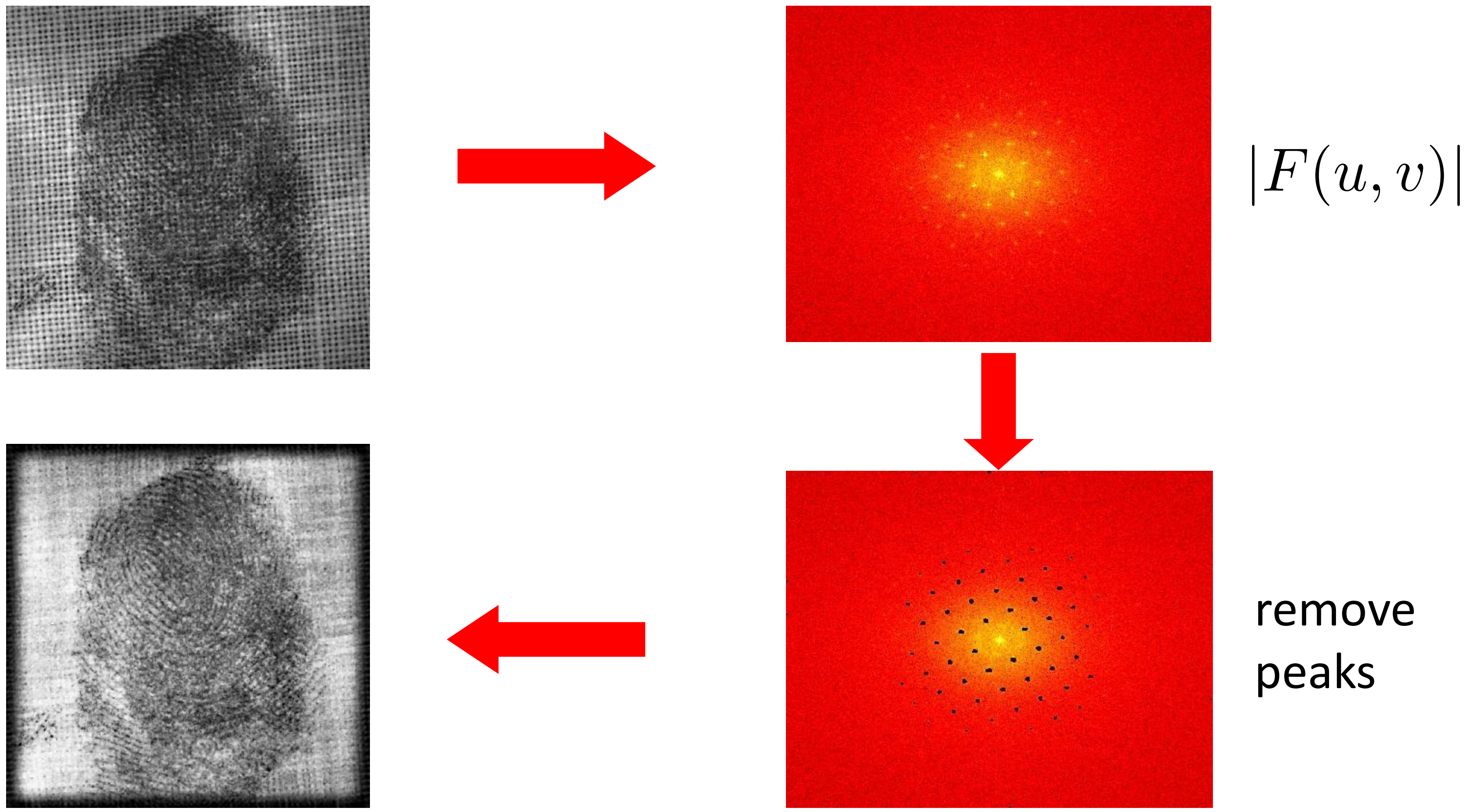
$|F(u,v)|$

FT has peaks at spatial frequencies of repeated texture

Computer Vision

Part I – Intro
Part I – Sampling

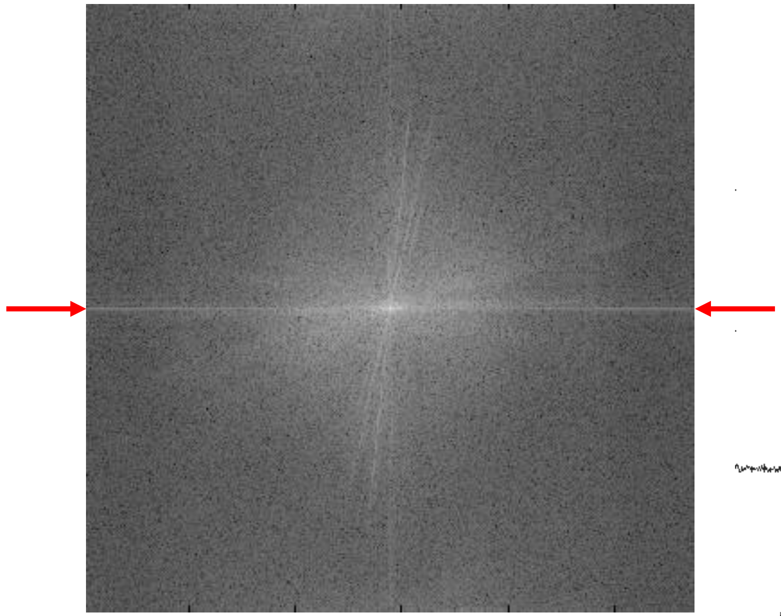
Importance of the magnitude in FT



Periodic background removed

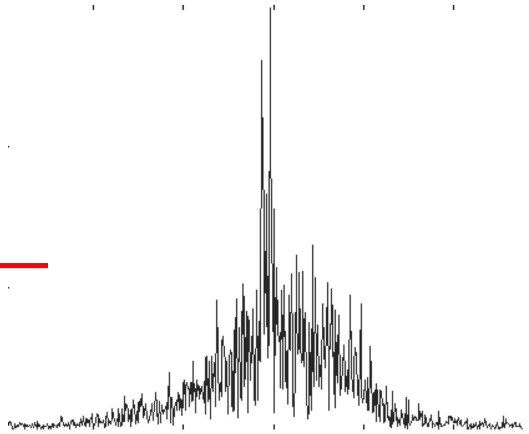
General structure of the magnitude

$f(x,y)$

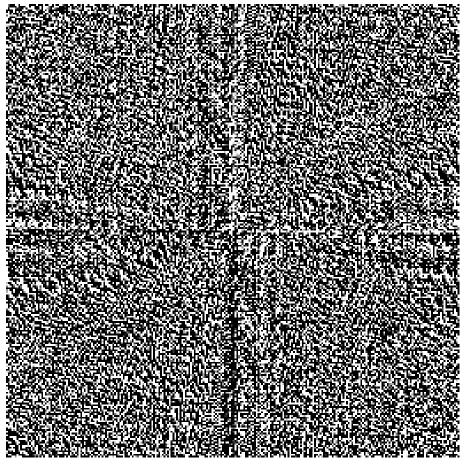


$|F(u,v)|$

cross-section

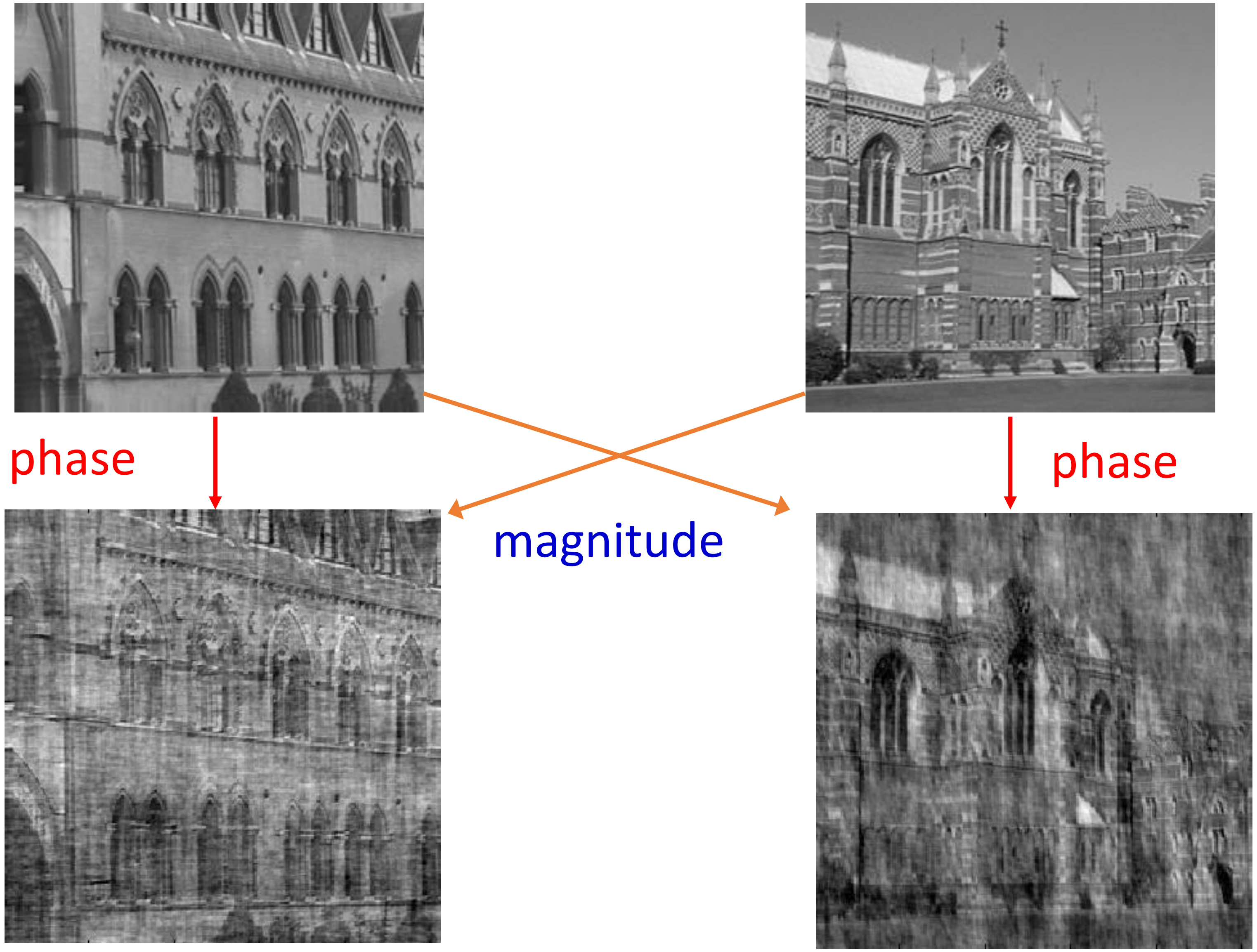


Phase



- Magnitude generally decreases with higher spatial frequencies
- phase appears less informative

Importance of the phase in FT



Computer Vision

Part I – Intro
Part I – Sampling

The convolution theorem

$$c(x,y) = a(x,y) * b(x,y)$$

What is the FT of a convolution?

$$\begin{aligned} C(u,v) &= \int\int [a(x,y) * b(x,y)]\, e^{-i2\pi(ux+vy)} dxdy \\ &= \int\int \left[\int\int a(x-\alpha, y-\beta) b(\alpha,\beta) d\alpha d\beta\right] e^{-i2\pi(ux+vy)} dxdy \\ &= \int\int \left[\int\int a(x-\alpha, y-\beta) e^{-i2\pi(ux+vy)} dxdy\right] b(\alpha,\beta) d\alpha d\beta \end{aligned}$$

Computer Vision

Part I – Intro
Part I – Sampling

The convolution theorem

$$\begin{aligned} C(u,v) &= \int\int\left[\int\int a(x-\alpha,y-\beta)e^{-i2\pi(ux+vy)}dxdy\right]b(\alpha,\beta)d\alpha d\beta \\ &= \int\int\left[\int\int a(x',y')e^{-i2\pi(u(x'+\alpha)+v(y'+\beta))}dx'dy'\right]b(\alpha,\beta)d\alpha d\beta \\ &= \int\int\left[\int\int a(x',y')e^{-i2\pi(ux'+vy')}dx'dy'\right]b(\alpha,\beta)e^{-i2\pi(u\alpha+v\beta)}d\alpha d\beta \end{aligned}$$

Noticing the two separate FT in this four integral term leads to the main result

$$\begin{aligned} C(u,v) &= \int\int A(u,v)b(\alpha,\beta)e^{-i2\pi(u\alpha+v\beta)}d\alpha d\beta \\ &= A(u,v)B(u,v) \end{aligned}$$

Space convolution = frequency multiplication

Reciprocity in convolution theorem

$$\begin{aligned} C(u,v) &= A(u,v)B(u,v) \\ c(x,y) &= a(x,y) * b(x,y) \end{aligned}$$

$$\begin{aligned} C(u,v) &= A(u,v) * B(u,v) \\ c(x,y) &= a(x,y)b(x,y) \end{aligned}$$

Space multiplication = frequency convolution

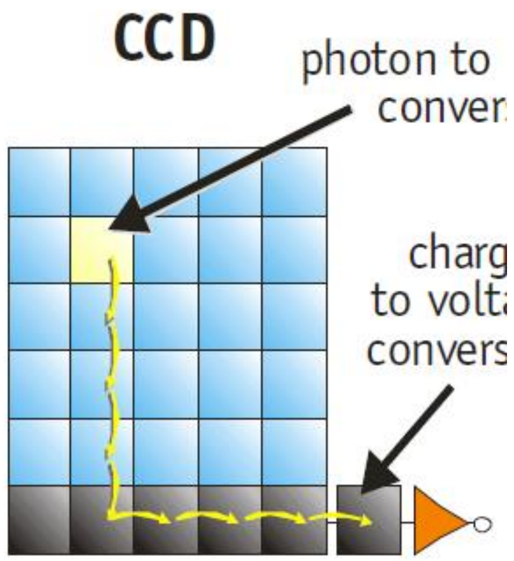
Point spread function and Modulation transfer function

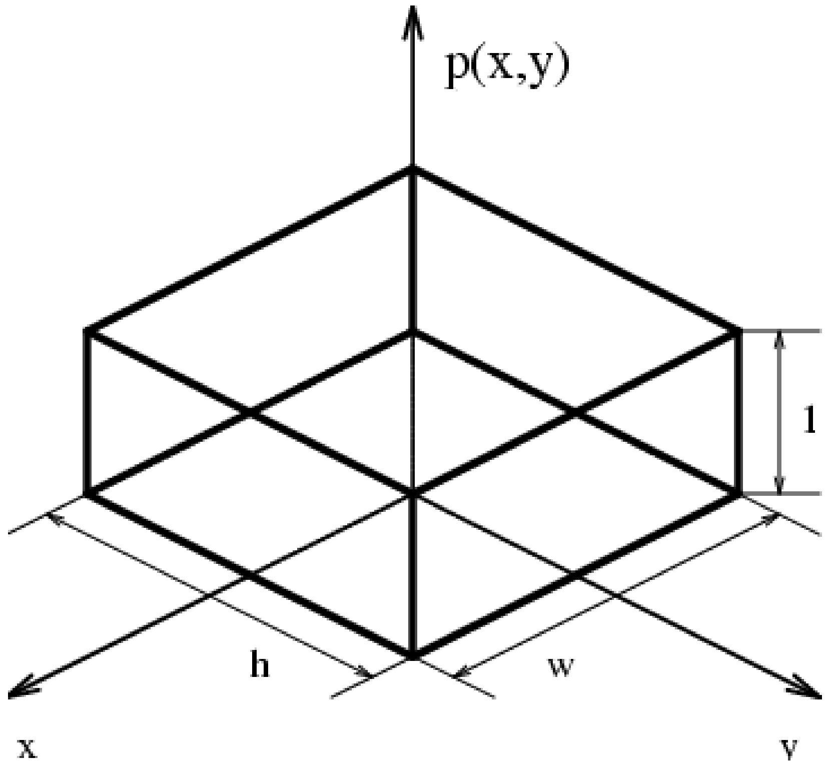
- When we talk about an imaging system where there is an image i(x,y) and a kernel r(x,y) that convolves the image, it is common to call the kernel the point spread function
- The convolution spreads the intensities to adjacent pixels based on r(x,y)
- Widely used terminology in microscopic imaging

$$\begin{aligned} O(u,v) &= \mathcal{F}\{o(x,y)\} \\ &= \mathcal{F}\{i(x,y) * r(x,y)\} \\ &= I(u,v)R(u,v) \\ R(u,v) &= \mathcal{F}\{r(x,y)\} \\ &= \mathcal{F}\{\text{point spread function}\} \end{aligned}$$

= **modulation transfer function**

STEP I: integrating over a cell window

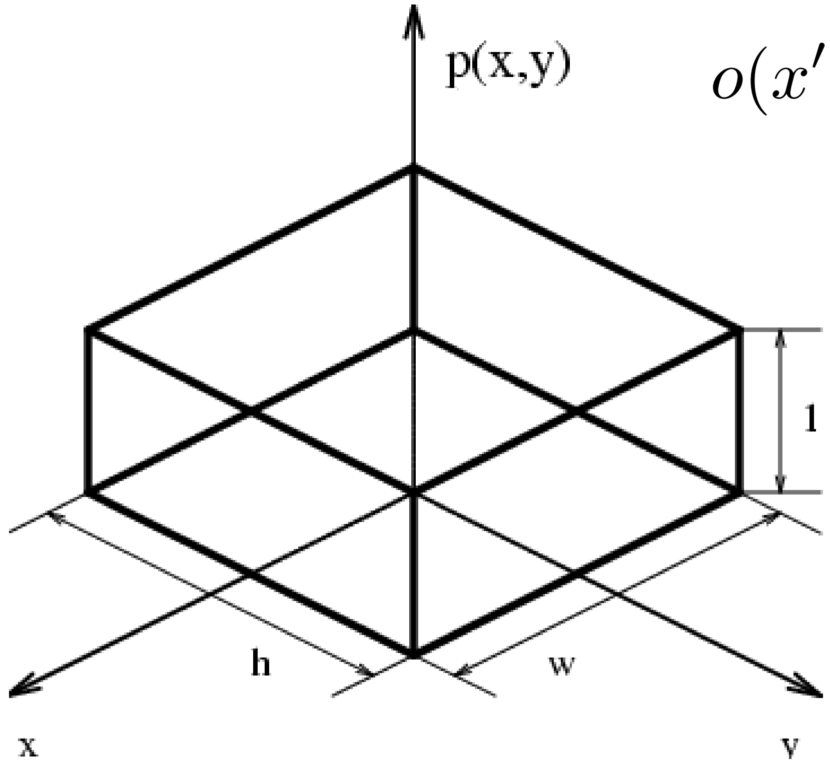




$$o(x',y') = \int\int i(x,y)p(x-x',y-y')dxdy$$

This is a *convolution*: $i(x,y) * p(-x,-y)$

Integrating over a cell window



$$o(x',y') = \int\int i(x,y)p(x-x',y-y')dxdy$$

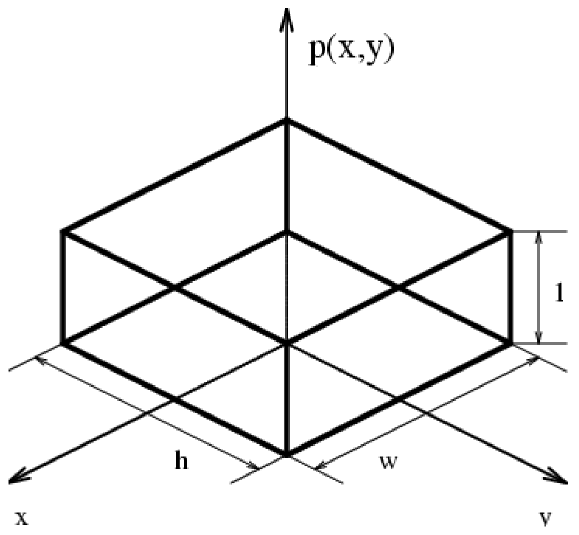
Assuming p(x,y) is symmetric around the origin
From convolution theorem

$$O(u,v) = I(u,v)P(u,v)$$

Computer Vision

Modulation transfer function of the window function

Fourier transform of window :



$$
\begin{aligned}
P(u,v) &= \int\int e^{-i2\pi(ux+vy)} p(x,y)\,dx\,dy \\
&= \int_{-w/2}^{w/2} e^{-i2\pi ux}\,dx \int_{-h/2}^{h/2} e^{-i2\pi vy}\,dy \\
&= wh\left(\frac{\sin(\pi wu)}{\pi wu}\right)\left(\frac{\sin(\pi hv)}{\pi hv}\right)
\end{aligned}
$$

2D sinc function

Modulation transfer function – 2D sinc

predominantly low-pass

real ☑ no phase shifts

however, phase reversals !

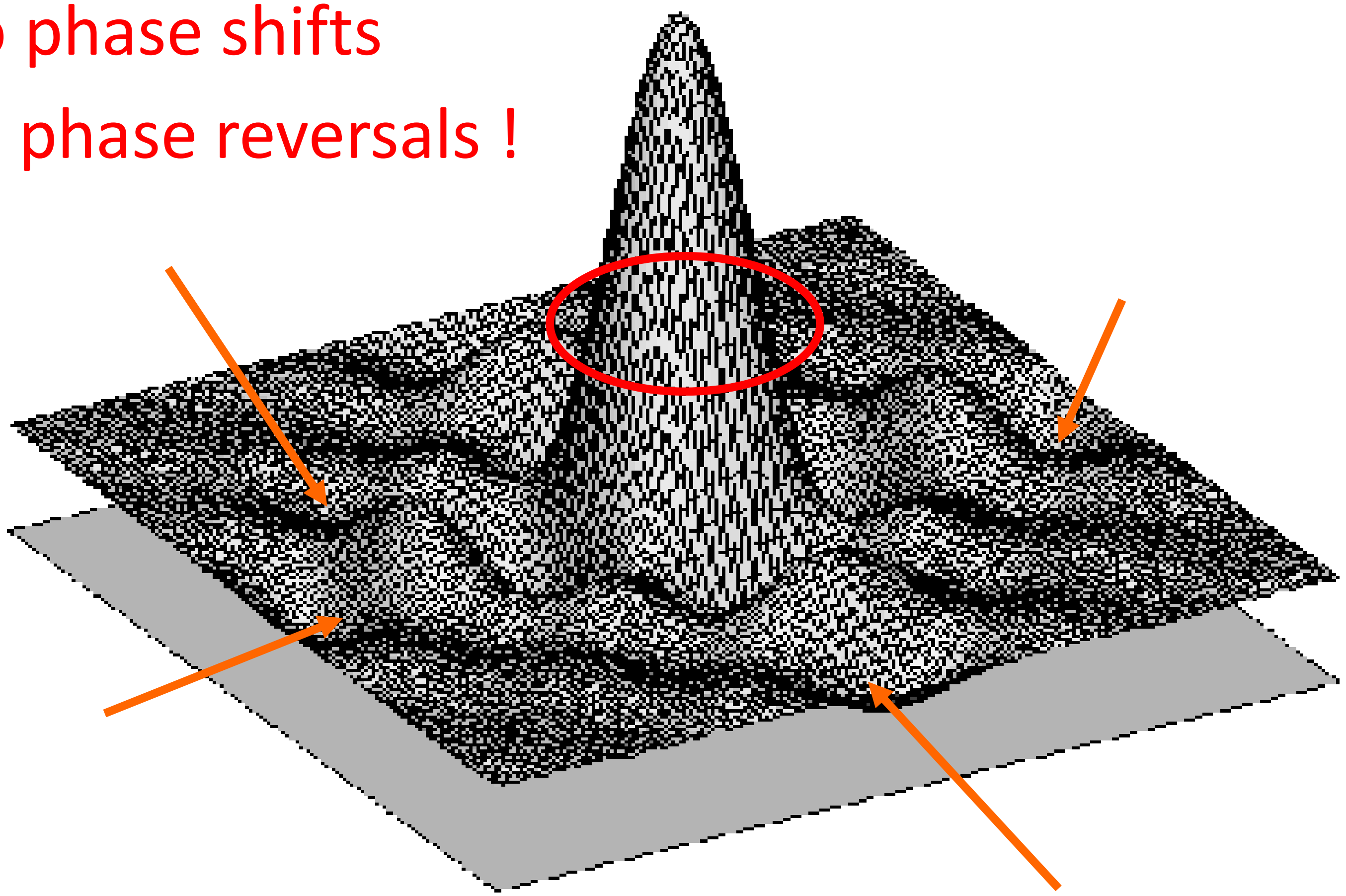
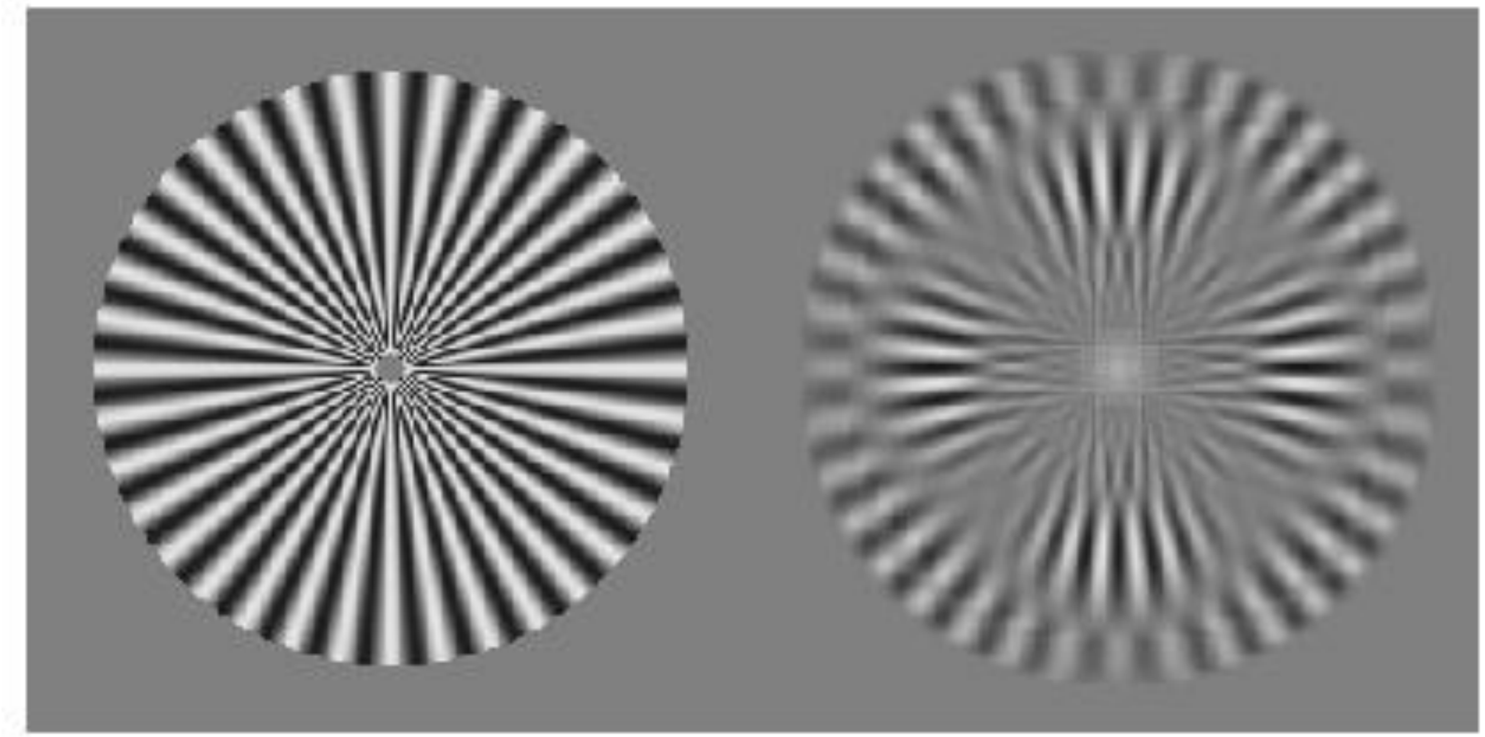


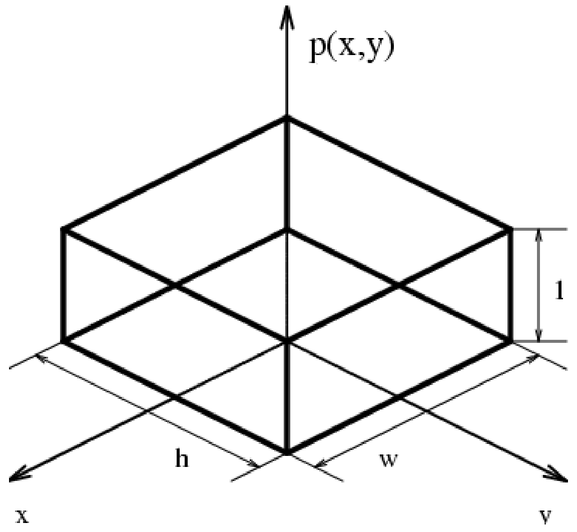
Illustration of the effect of 2D sinc

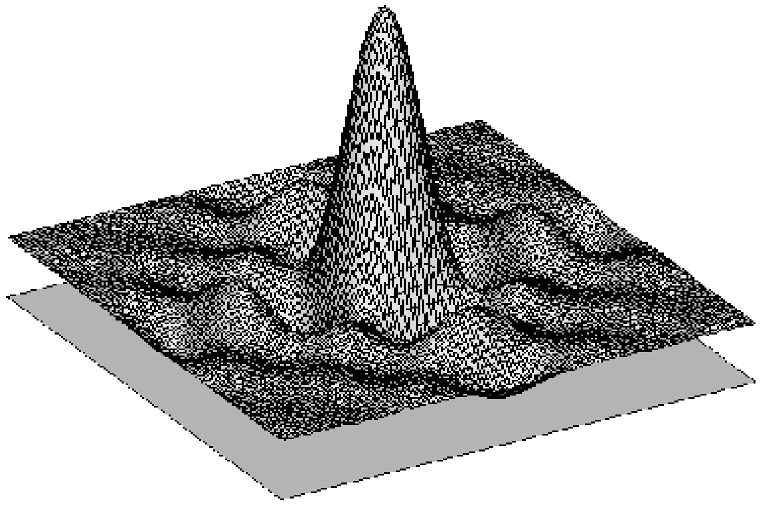


$$P(u,v) = wh\left(\frac{\sin \pi w u}{\pi w u}\right)\left(\frac{\sin \pi h v}{\pi h v}\right)$$

Summary for STEP I

- Convolve with a window function – rectangular box
- Blurs the image
- May cause phase reversals in certain frequencies – modify the image content





A model for sampling

- There are two essential steps

1. Integrate brightness over a cell window
   Leads to blurring type degradation

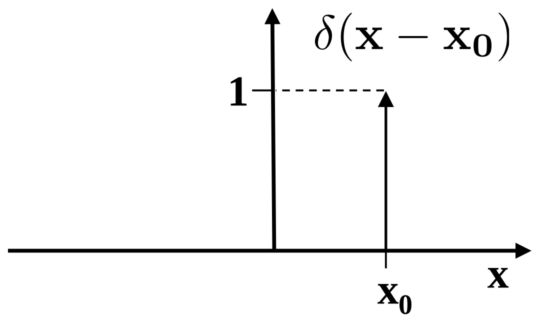
2. Read out values only at the pixel centers
   Leads to aliasing and leakage, frequency domain issues

Local probing of functions

To understand the effect of Step II, we need the probing function: Dirac pulse

$$\delta(\mathbf{x} - \mathbf{x}_0) = 0 \quad \mathbf{x} \neq \mathbf{x}_0$$

$$\int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} \delta(\mathbf{x} - \mathbf{x}_0) d\mathbf{x} = 1$$



Function probing (in 1D)

$$\int_{-\infty}^{\infty} \delta(x) f(x) dx = f(0)$$

$$\int_{-\infty}^{\infty} \delta(x - x_0) f(x) dx = f(x_0)$$

Discretization in the spatial domain is multiplication with a Dirac train

2D Dirac train / Dirac comb

$$\sum_{k=-\infty}^{\infty} \sum_{l=\infty}^{\infty} \delta(x - kw, y - lh)$$

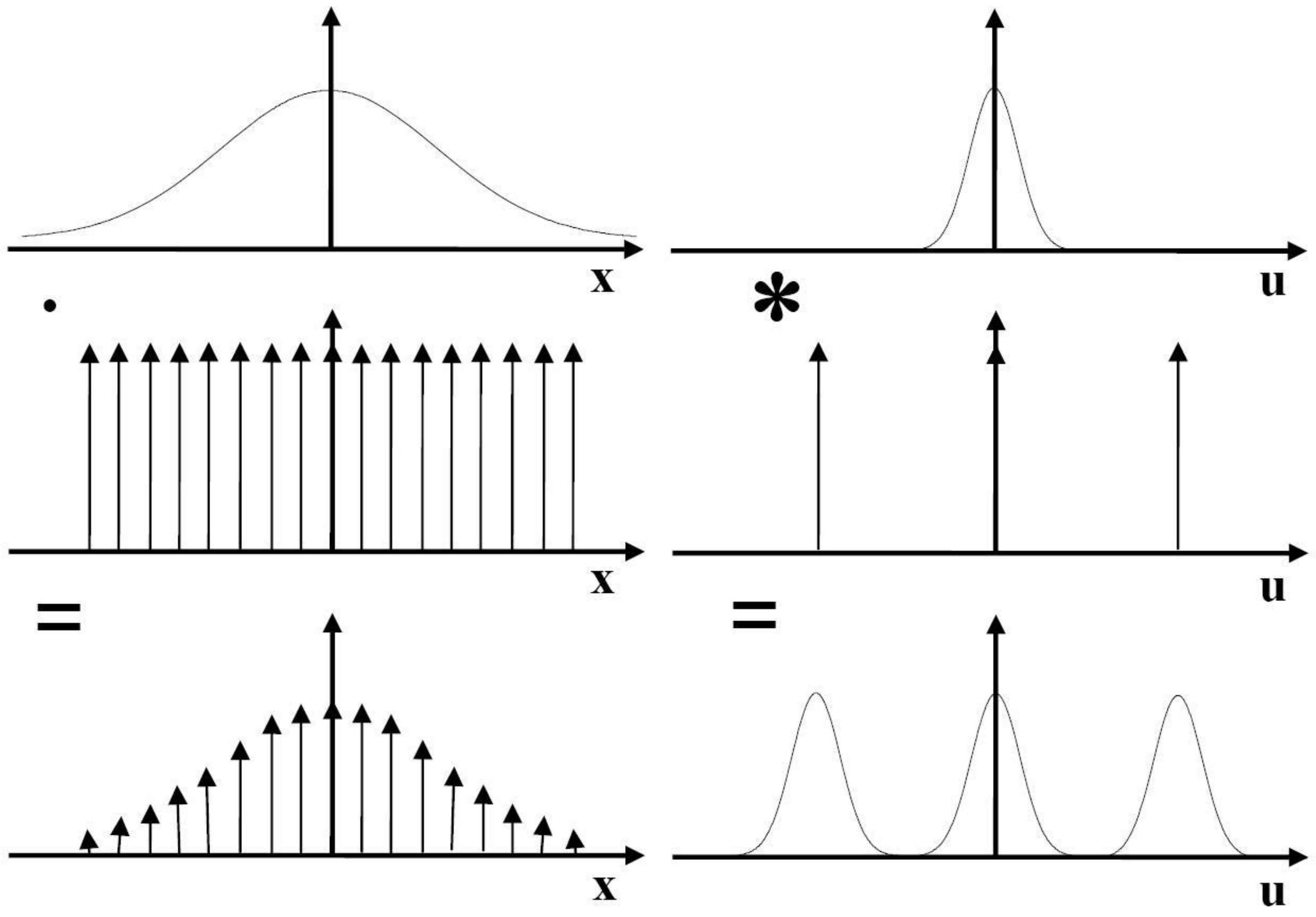
Fourier transform is also a Dirac train / Dirac comb

$$\frac{1}{wh} \sum_{k=-\infty}^{\infty} \sum_{l=\infty}^{\infty} \delta(u - k\frac{1}{w}, v - l\frac{1}{h})$$

Convolution with a Dirac train: periodic repetition

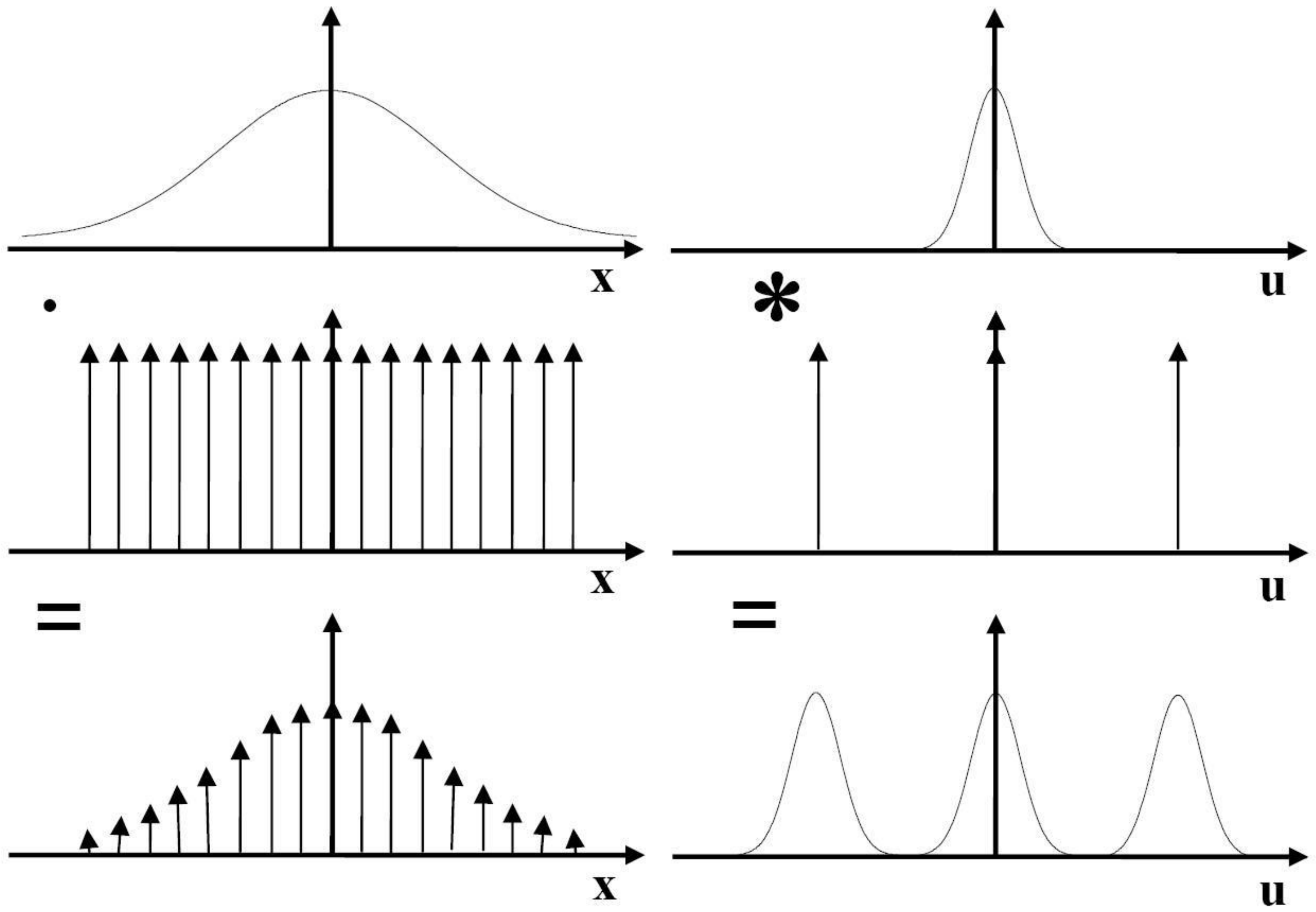
Yet another duality: discrete vs. periodic

Effect on the frequency domain



Effect on the frequency domain

1. After sampling you may not get back the original signal
2. It depends on the frequency domain representation, only band limited signals can be sampled and retrieved back
3. Even then you need to sample at a certain rate



The sampling theorem

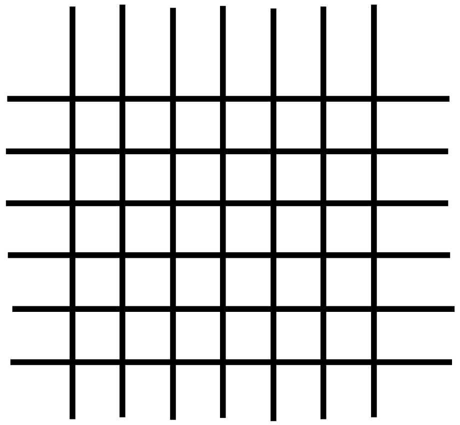
If the Fourier transform of a function f(x,y) is zero for all frequencies beyond u_b and v_b, i.e. if the Fourier transform is *band-limited*, then the continuous periodic function f(x,y) can be completely reconstructed from its samples as long as the sampling distances w and h along the x and y directions are such that $w \leq \frac{1}{2u_b}$ and $h \leq \frac{1}{2v_b}$

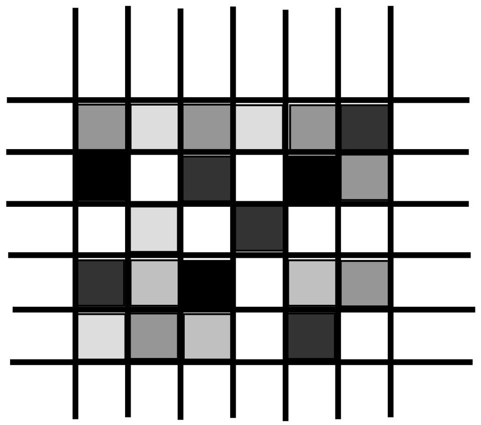
Summary for STEP II

- When we read off one value per pixel area, we are losing information on the image indefinitely, if the image is not band-limited, which is almost always the case.
- The information we lose is on the higher frequencies, meaning very fine details on edges, corners and texture patterns.

Discretization / Digitization

- Necessary computer to process an image
- Includes two parts
  1. Sampling – spatial discretization, creates “pixels”
  2. Quantization – intensity discretization, creates “grey levels”





Quantization

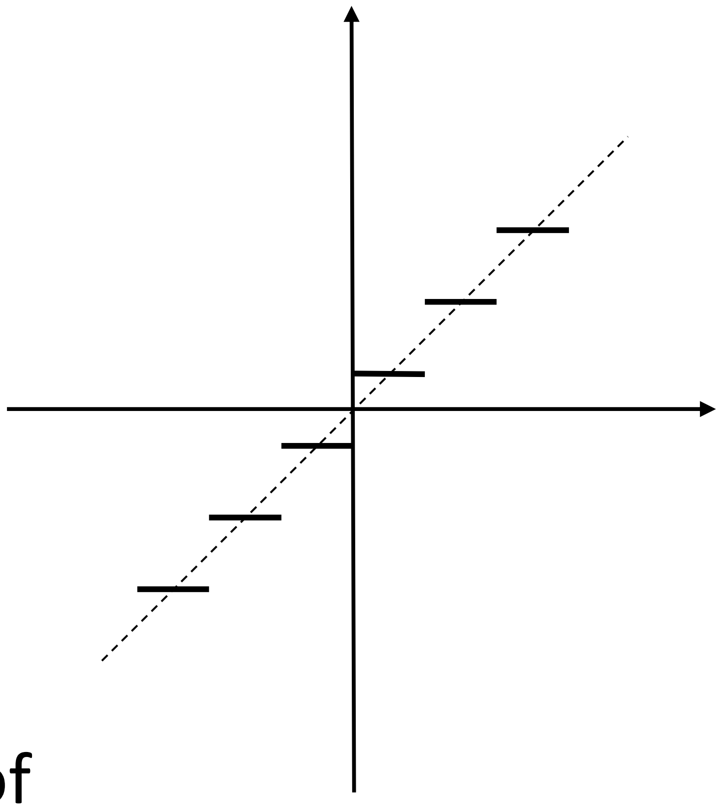
- Create K intervals in the range of possible intensities and each interval with only one value
- Measured in bits: log2(K)
- Design choices:
  - Decision levels / boundaries of intervals $z_1, z_2, \ldots, z_{K-1}$
  - Representative values for each interval $[z_i, z_{i+1}] \rightarrow q_i$
- Simplest selection
  - Equal intervals between min and max
  - Use mean in the interval as the representative value
  - Uniform quantizer
  - K=256 is used very often in practice

The uniform quantizer

- Simple interpretation
- Fine quantization is needed for perceptual quality (7-8 bits)
- It can be better designed if we know what intensities we expect

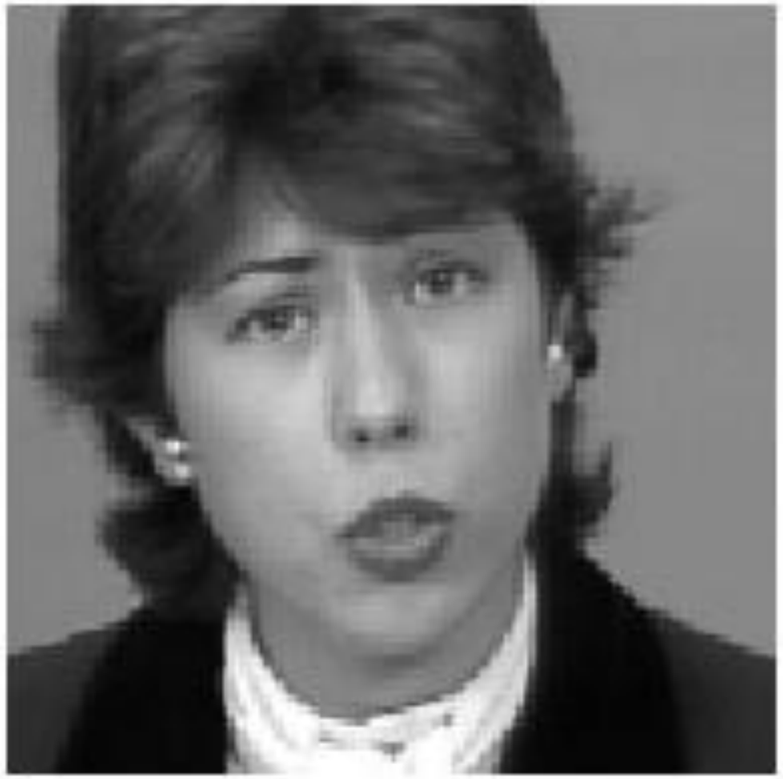
$$\min \sum_{k=1}^{K} \int_{z_k}^{z_{k+1}} (z - q_k)^2 p(z) dz$$

- p(z) is the probability density function of intensities – constant for uniform quantizer

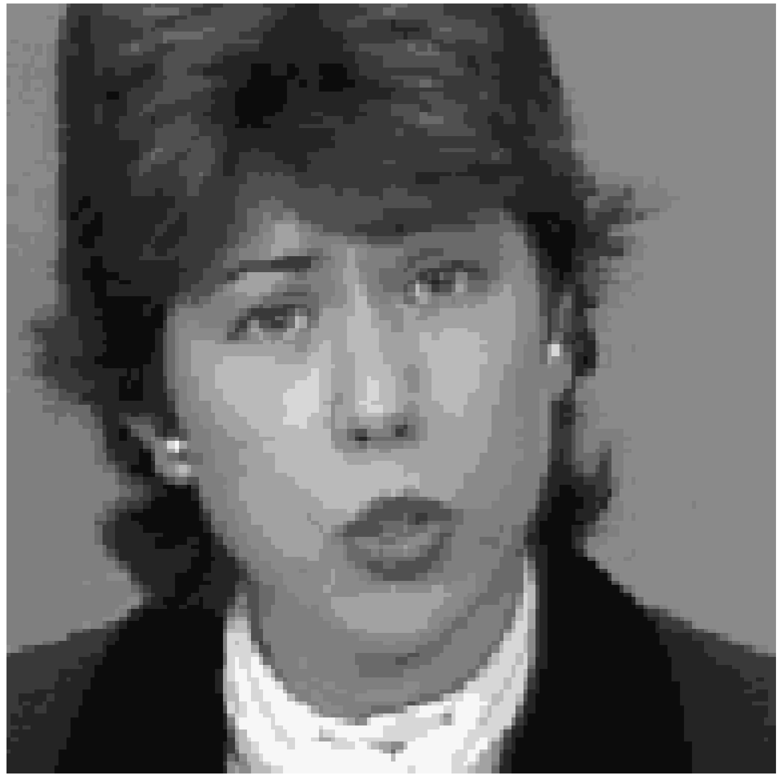


Underquantization examples

256 gray level (8 bit)



11 gray level



Small remarks on quantization

- 8 bits is often used in monochrome images
- 24 bits (8 x 3) used for RGB images per pixel
- Medical imaging may require finer quantization. 12 bits (4096 levels) and 16 bits (65536) are often used.
- Satellite imaging also use 12 or 16 bits regularly.

Computer Vision

Part I : Sampling & quantization

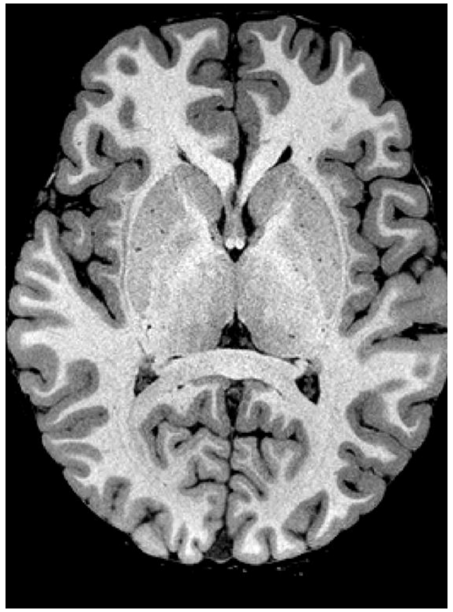
1. Discretization of continuous signals
2. Signal representation in the frequency domain
3. Effects of sampling and quantization

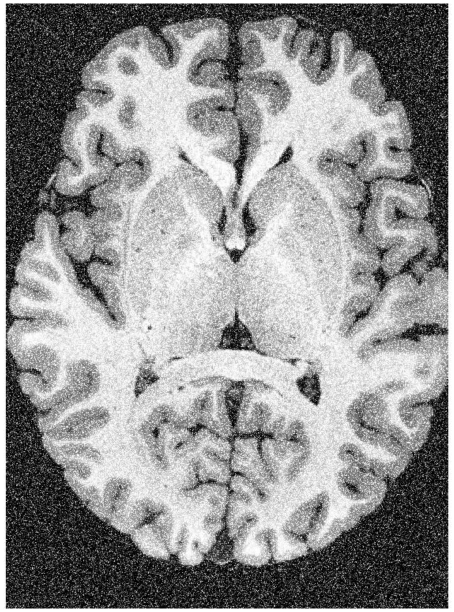
Part II : Image enhancement

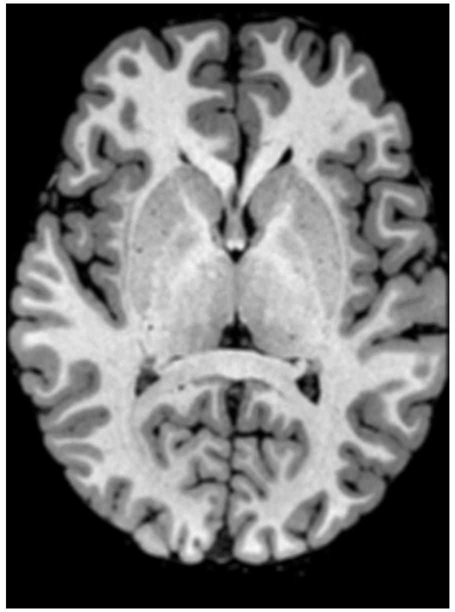
1. Noise suppression
2. De-blurring
3. Contrast enhancement

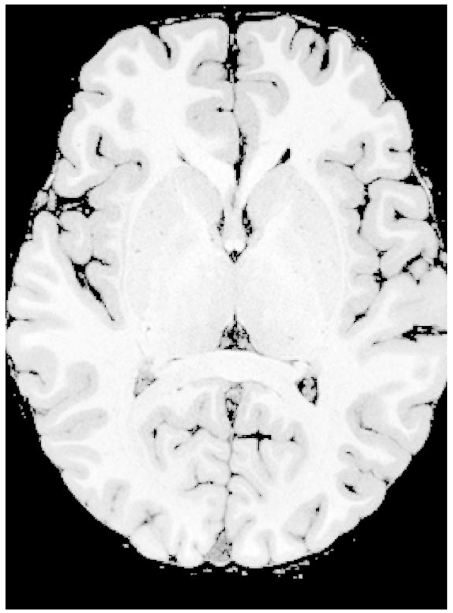
Three types of image enhancement

1. Noise suppression
2. Image de-blurring
3. Contrast enhancement


Original Image


Noise


Blur


Bad Contrast

More on Fourier transform

Signal and noise

The Fourier Transform: definition

Linear decomposition of functions in the new basis
Scaling factor for basis function (*u,v*)

The Fourier Transform

$$\mathcal{F}[f(x,y)] = F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-i2\pi(ux+vy)}dxdy$$

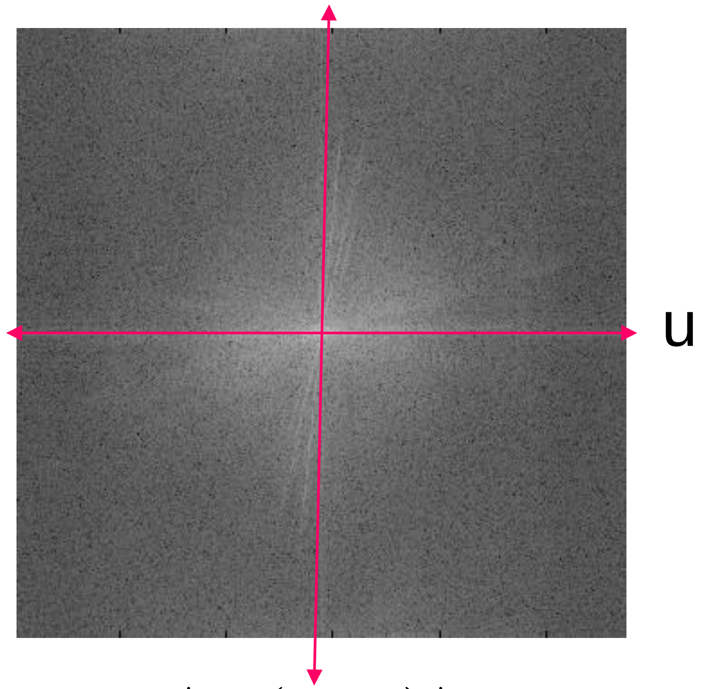
Reconstruction of the original function in the spatial domain: weighted sum of the basis functions

The Inverse Fourier Transform

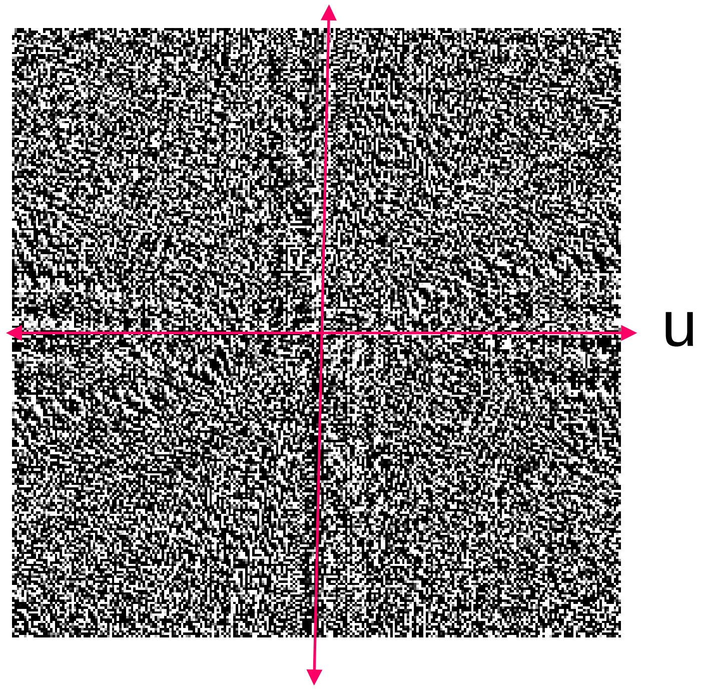
$$\mathcal{F}^{-1}[F(u,v)] = f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{i2\pi(ux+vy)}dxdy$$

Computer Vision

$f(x, y)$



$|F(u, v)|$



Phase

Convolution theorem

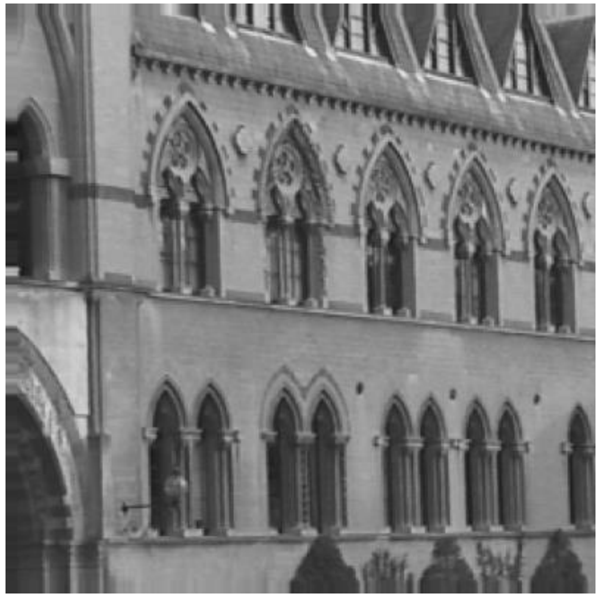
$$\begin{aligned} C(u,v) &= A(u,v)B(u,v) \\ c(x,y) &= a(x,y) * b(x,y) \end{aligned}$$

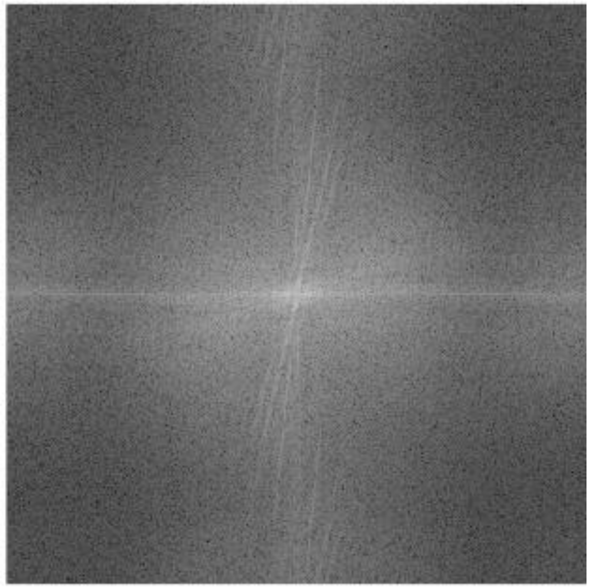
$$\boxed{\text{Space convolution = frequency multiplication}}$$

$$\begin{aligned} C(u,v) &= A(u,v) * B(u,v) \\ c(x,y) &= a(x,y)b(x,y) \end{aligned}$$

$$\boxed{\text{Space multiplication = frequency convolution}}$$

Fourier power spectra of images


$i(x,y)$


$\phi_{ii} = |I(u,v)|^2$

Amount of signal at each frequency pair

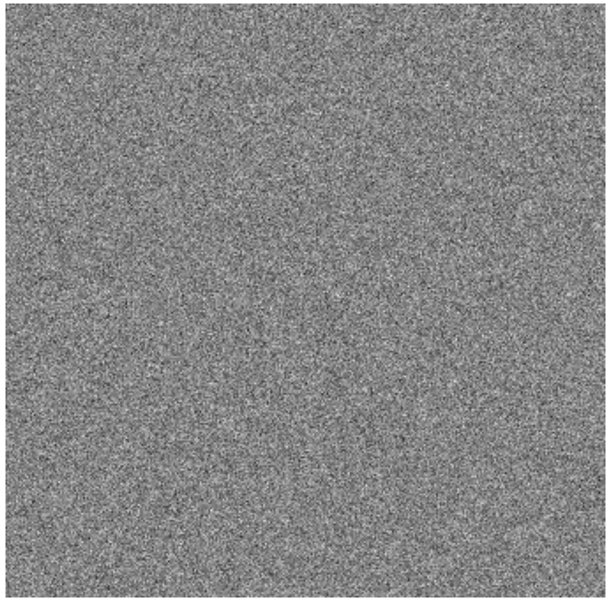
Images are mostly composed of homogeneous areas

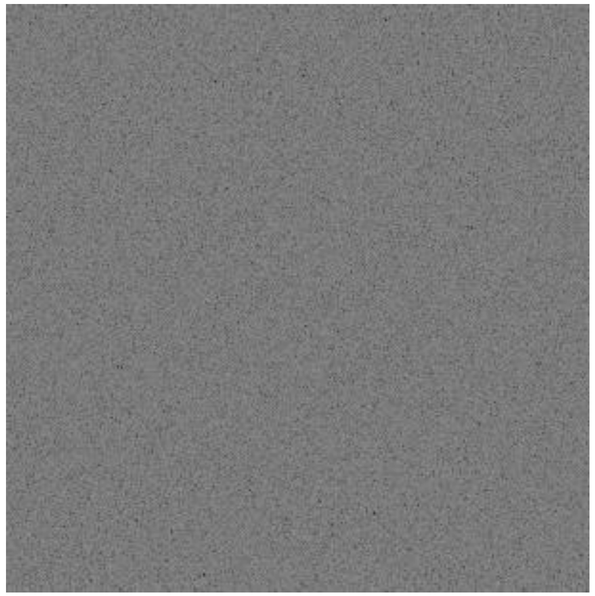
Most nearby object pixels have similar intensity

Most of the signal lies in low frequencies!

High frequency contains the edge information!

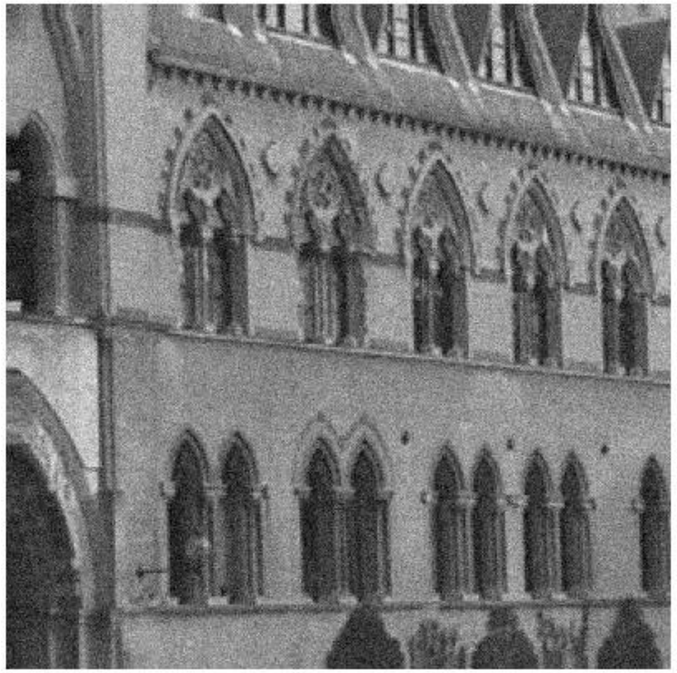
Fourier power spectra of noise


$n(x,y)$

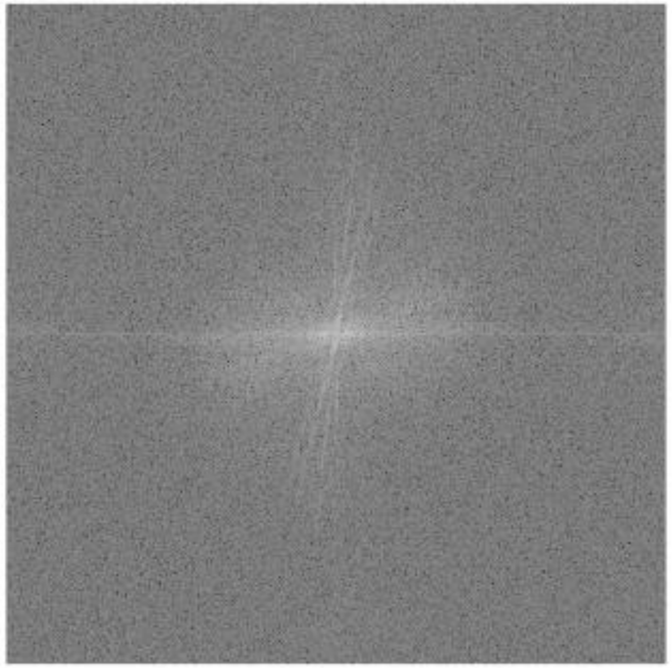

$\phi_{nn} = |N(u,v)|^2$

- Pure noise has a uniform power spectra
- Similar components in high and low frequencies.

Fourier power spectra of noisy image



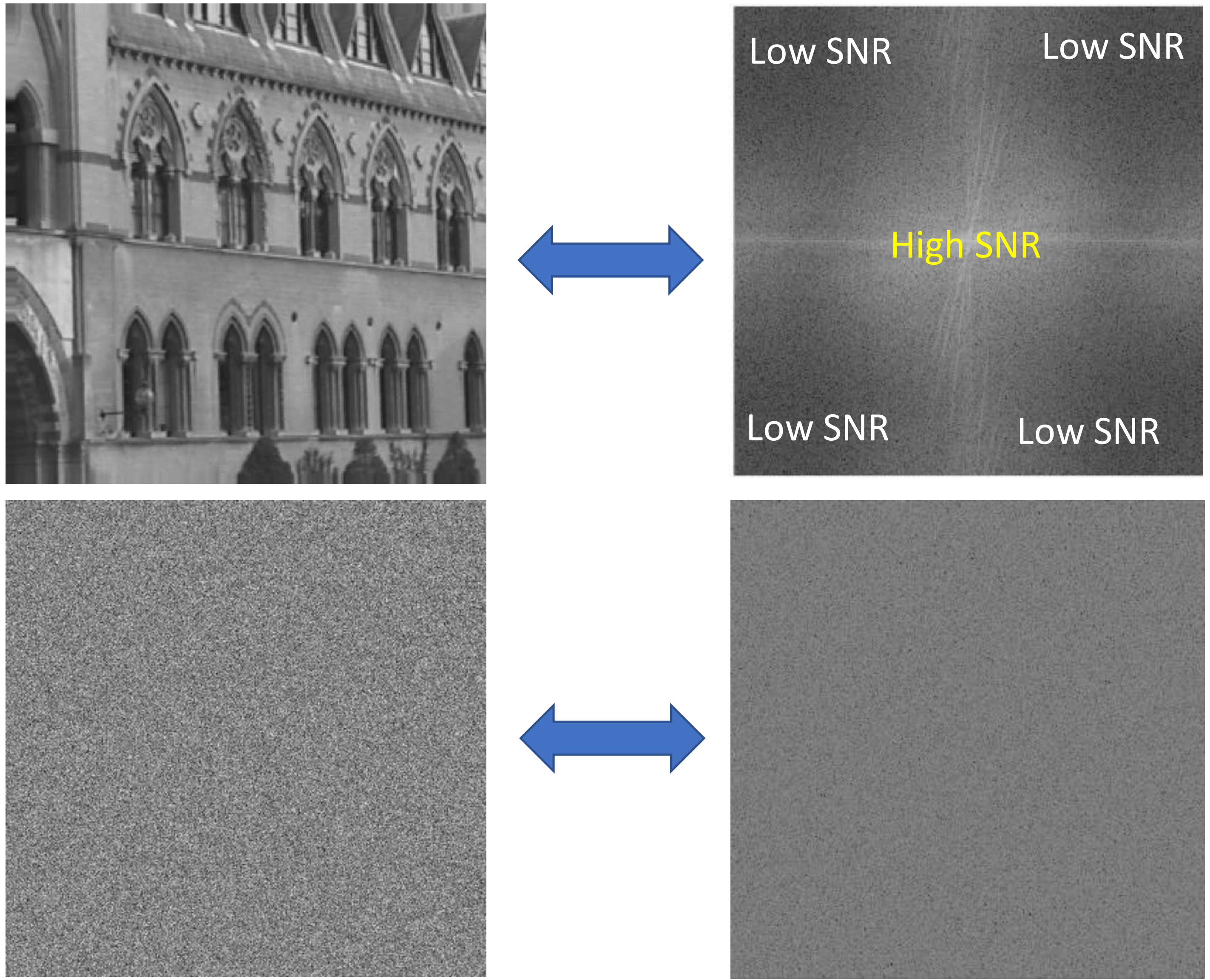
$f(x,y)$



$\phi_{ff} = |F(u,v)|^2$

Power spectra is a combination of image and noise

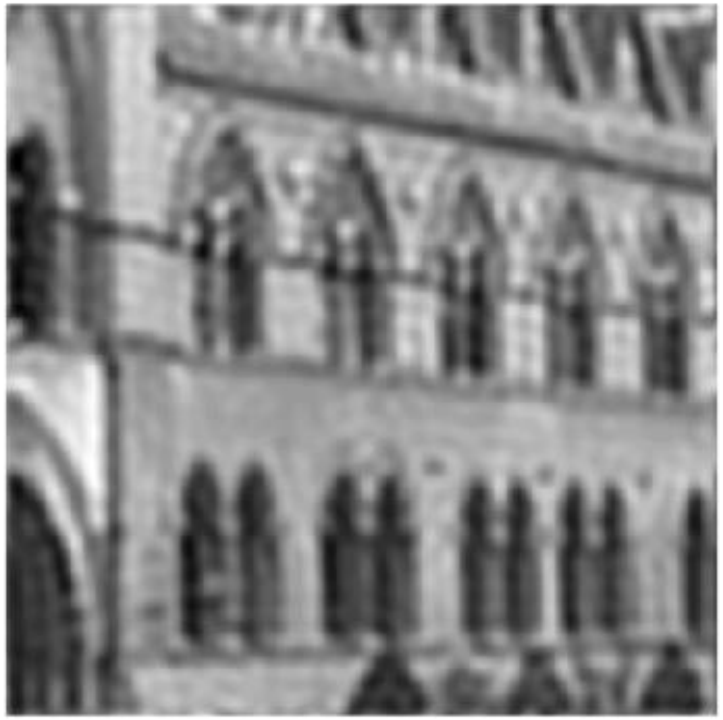
Signal to noise ratio (SNR)

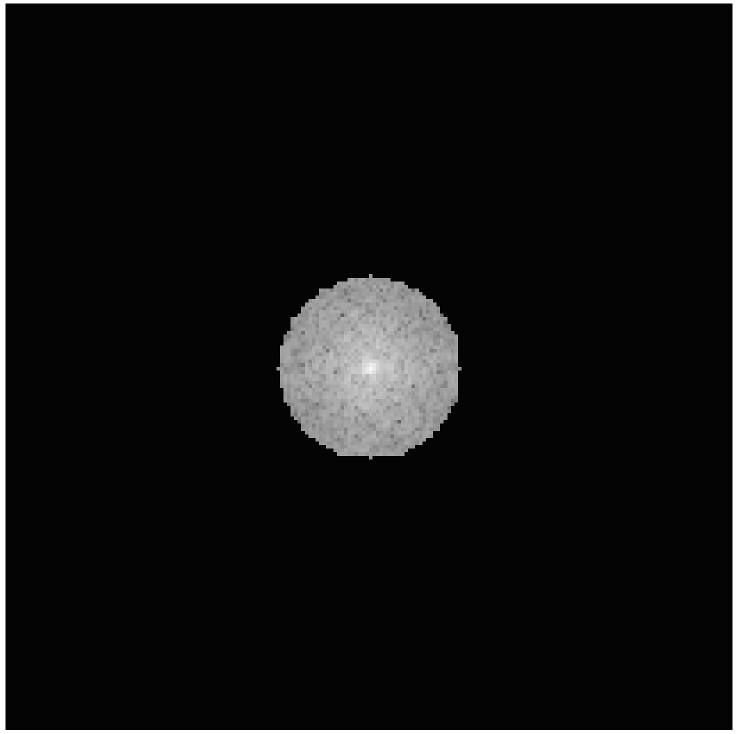


$\phi_{ii}(u,v) \, / \, \phi_{nn}(u,v)$

Only retaining the low frequencies

Low signal/noise ratio at high frequencies $\Rightarrow$ eliminate these



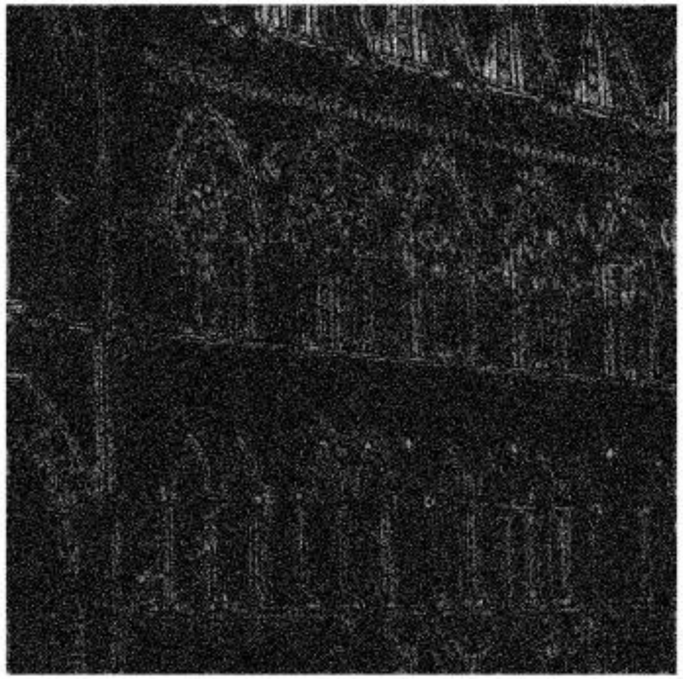


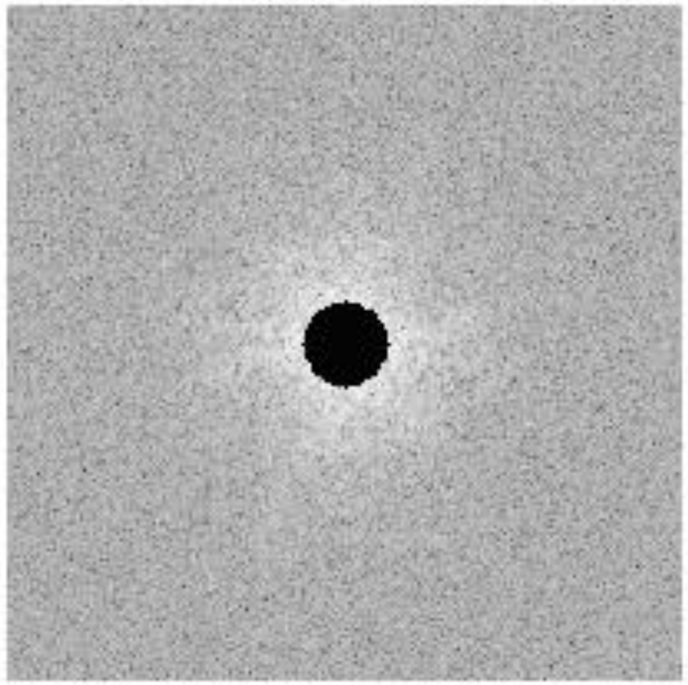
Smoother image but we lost details!

High frequencies contains noise and edge information

We cannot simply discard the higher frequencies

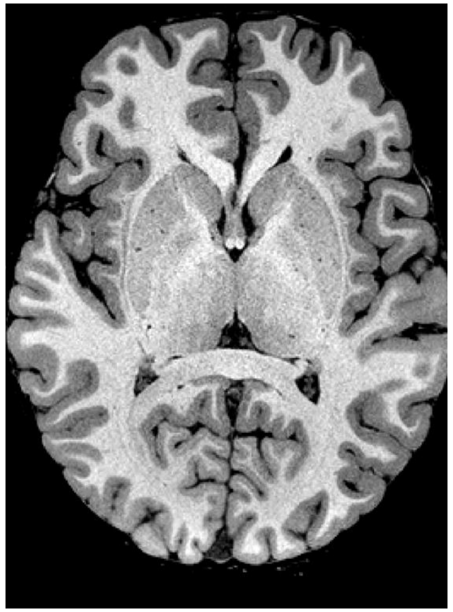
They are also introduced by edges

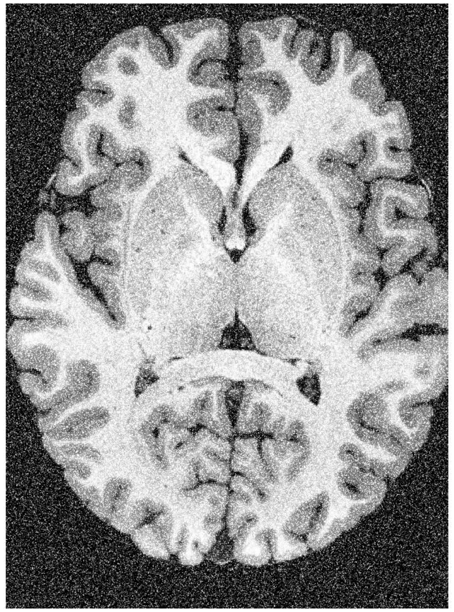


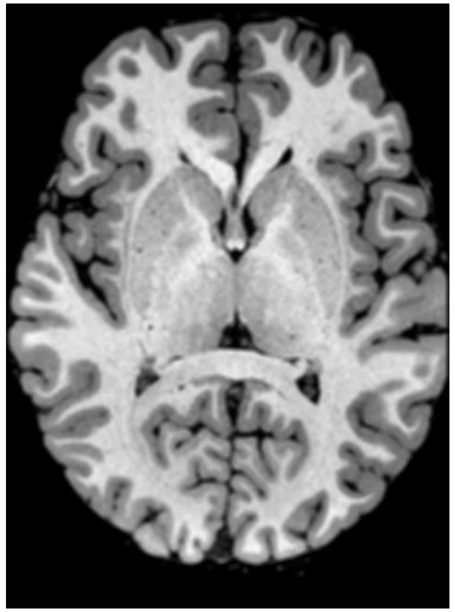


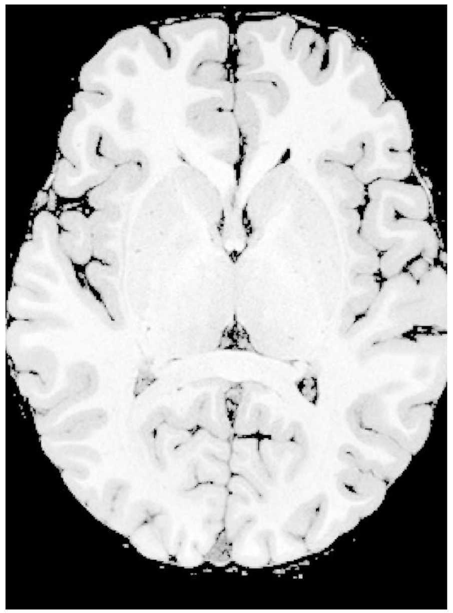
Three types of image enhancement

1. Noise suppression
2. Image de-blurring
3. Contrast enhancement

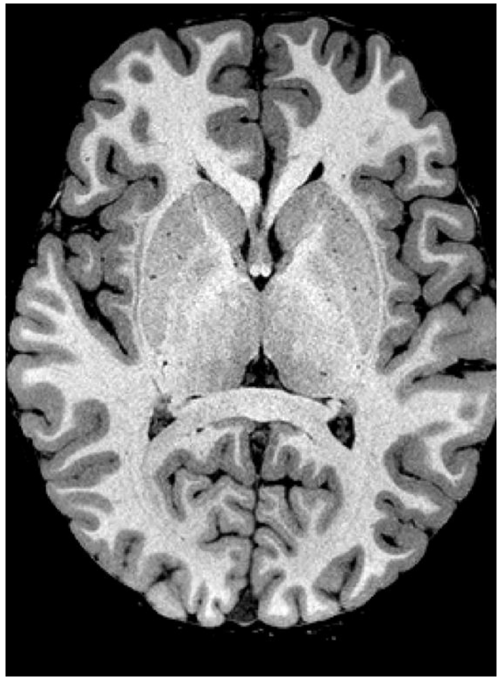

Original Image


Noise

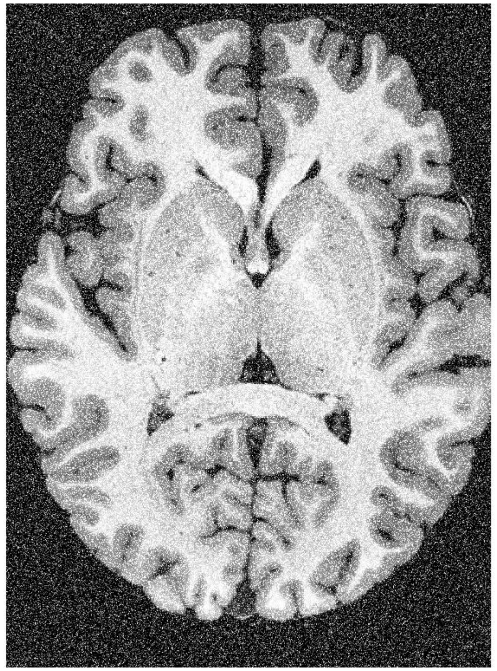

Blur


Bad Contrast

Computer Vision


Original Image

Noise Suppression


Noisy Observation

Noise suppresion

- In general specific methods for specific types of noise
- We only consider 2 general options here:

1. Convolutional linear filters – low-pass convolutional filters

2. Non-linear filters - edge-preserving filters
   a. Median
   b. Anisotropic diffusion

Low-pass filtering - principle

Goal: remove low-signal/noise part of the spectrum

Approach 1: Multiply the Fourier domain by a mask

Such spectrum filters yield “rippling”
due to ripples of the spatial filter and convolution

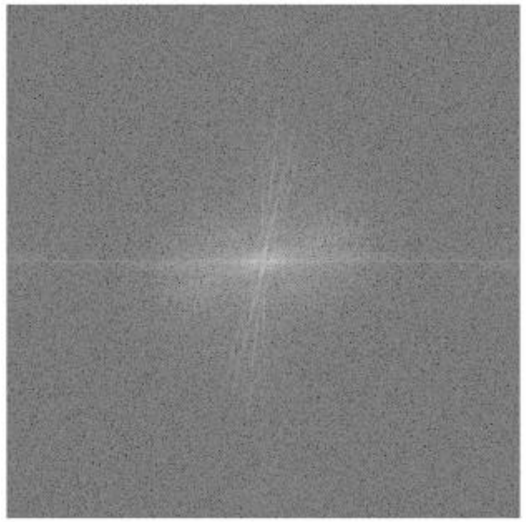
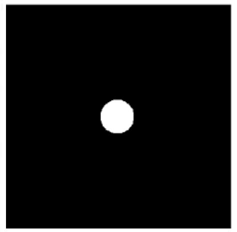
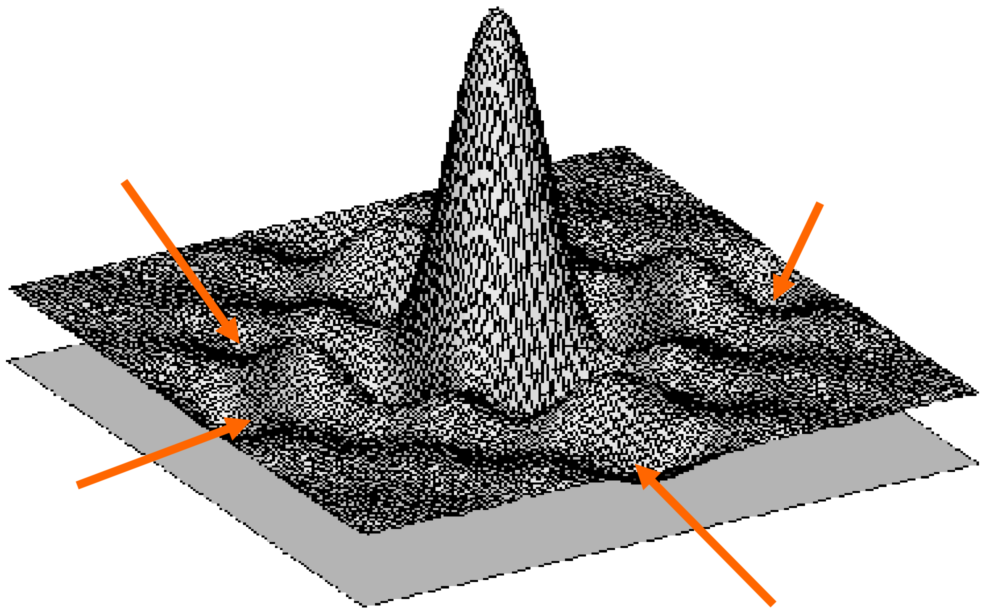
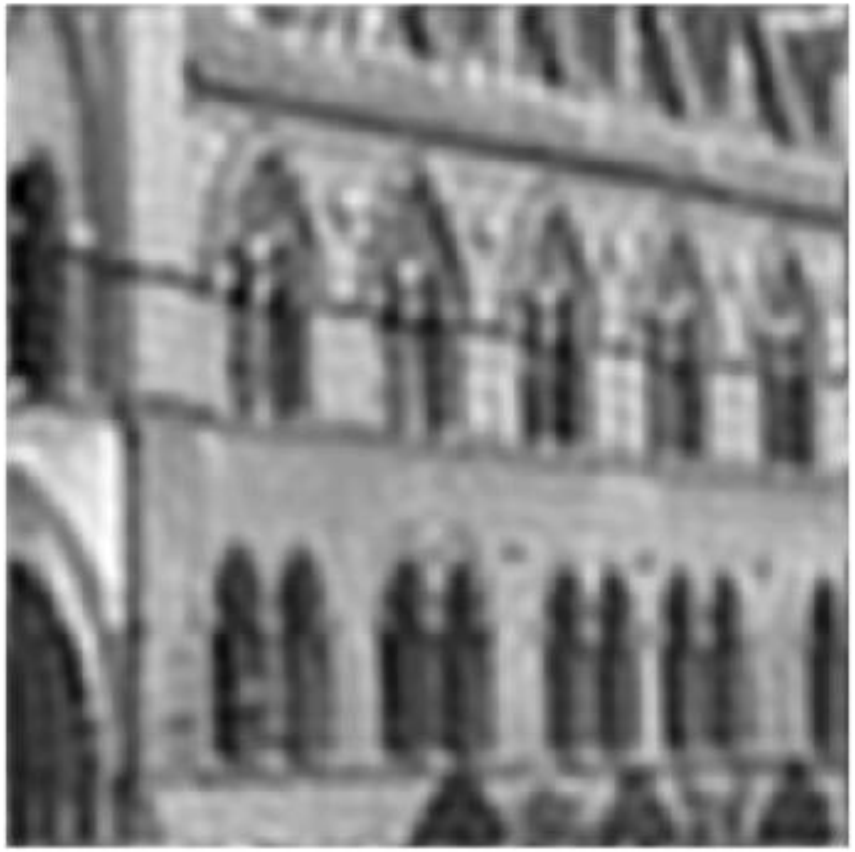
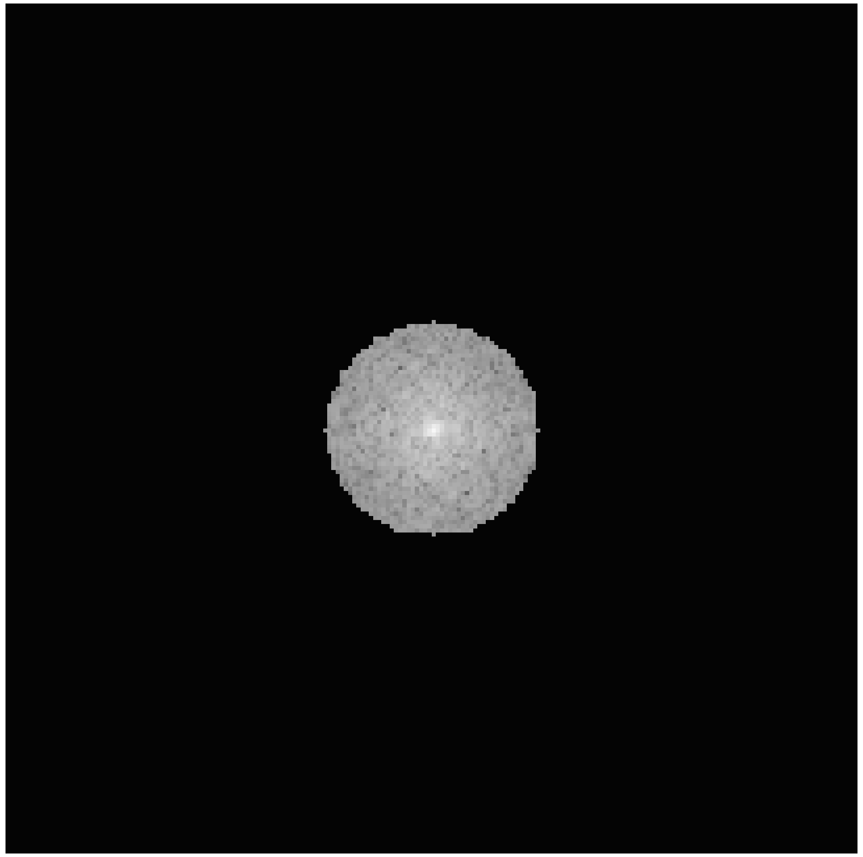




Illustration of rippling

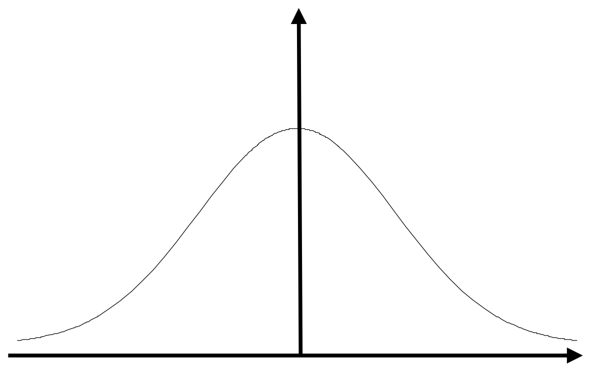


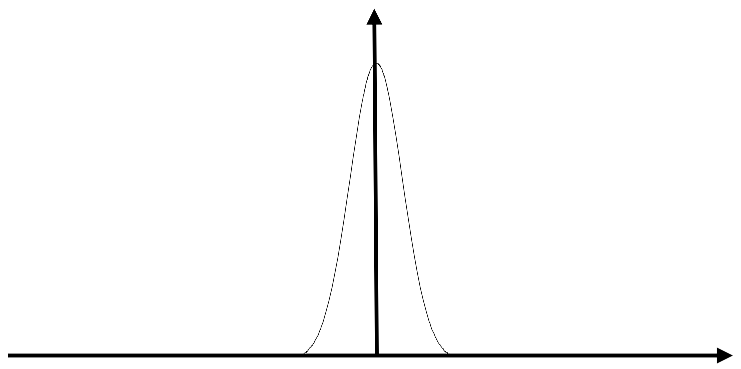


Low-pass filtering - principle

Approach 2: Low-pass convolution filters
generate low-pass filters that do not cause rippling

Idea: Model convolutional filters in the spatial domain to approximate low-pass filtering in the frequency domain


Convolutional filter


Frequency mask

Average filtering – Box filtering

One of the most straight forward convolution filters: averaging filters

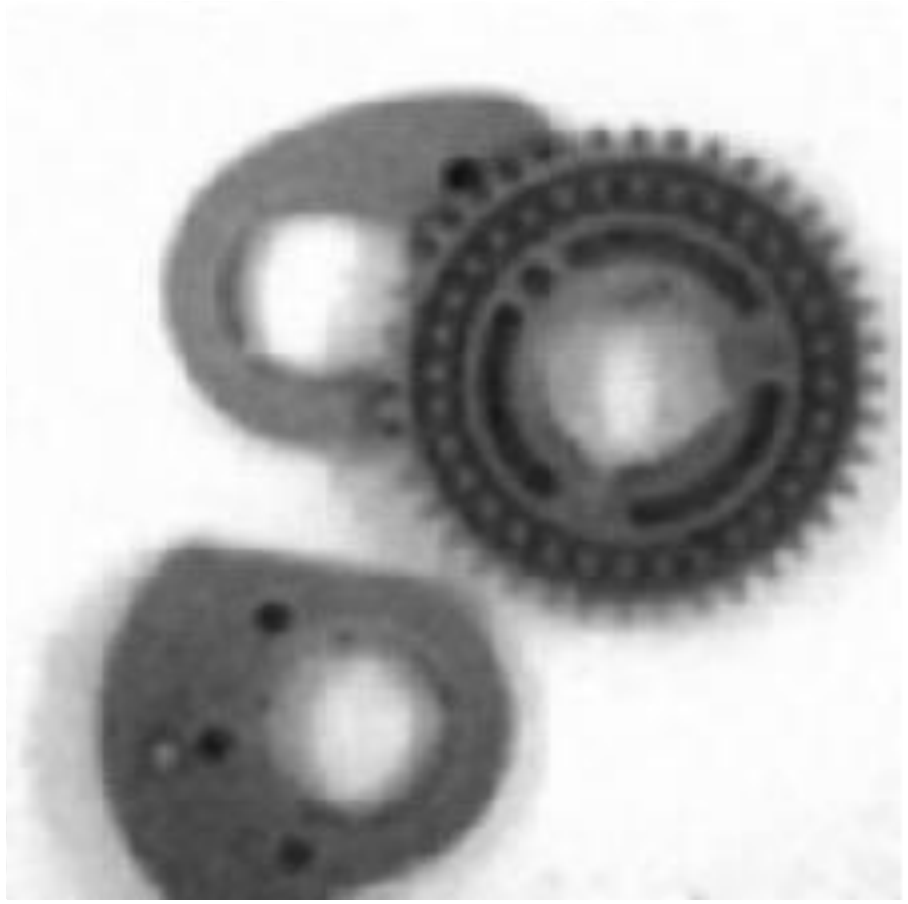
$$1/9\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array} \qquad 1/25\begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline\end{array}$$

Separable:

$$1/9\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array} = 1/3\begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 1 \\ \hline\end{array} * 1/3\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline\end{array}$$

$$o(x,y) = f(x,y) * i(x,y) = f_1(x,y) * (f_2(x,y) * i(x,y))$$

Example for box/average filtering

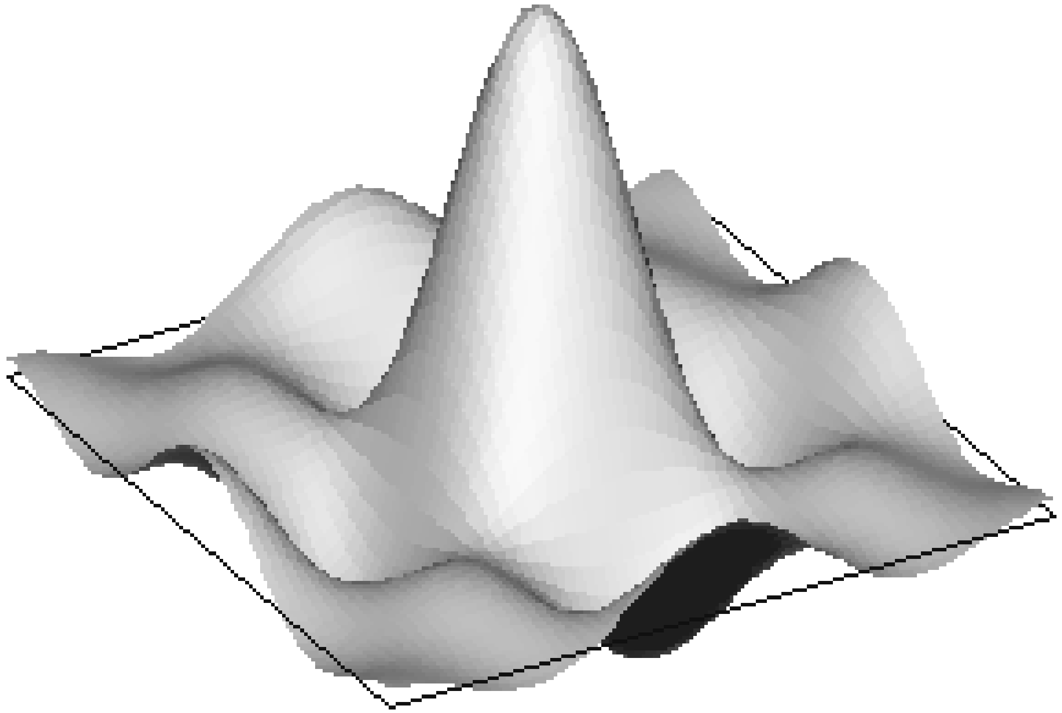

Noise is gone.
Result is blurred!

Computer Vision

MTF for box / average filtering

5 x 5 (separable)

$$(1+2\cos(2\pi u)+2\cos(4\pi u))(1+2\cos(2\pi v)+2\cos(4\pi v))$$



not even low-pass!

So far

1. Masking frequency domain with window type low-pass filter yields sinc-type of spatial filter and ripples -> disturbing effect

2. box filters are not exactly low-pass, ripples in the frequency domain at higher freq. remember phase reversals?

no ripples in either domain required!

Solution: Binomial filtering

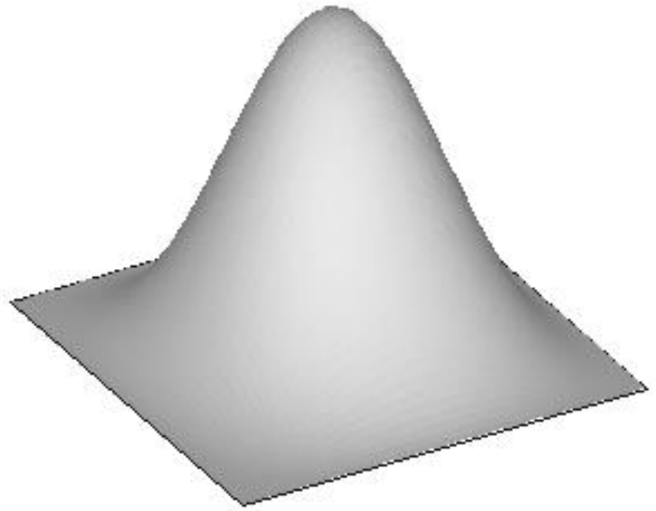
iterative convolutions of (1,1)

only odd filters : (1,2,1), (1,4,6,4,1)

2D :

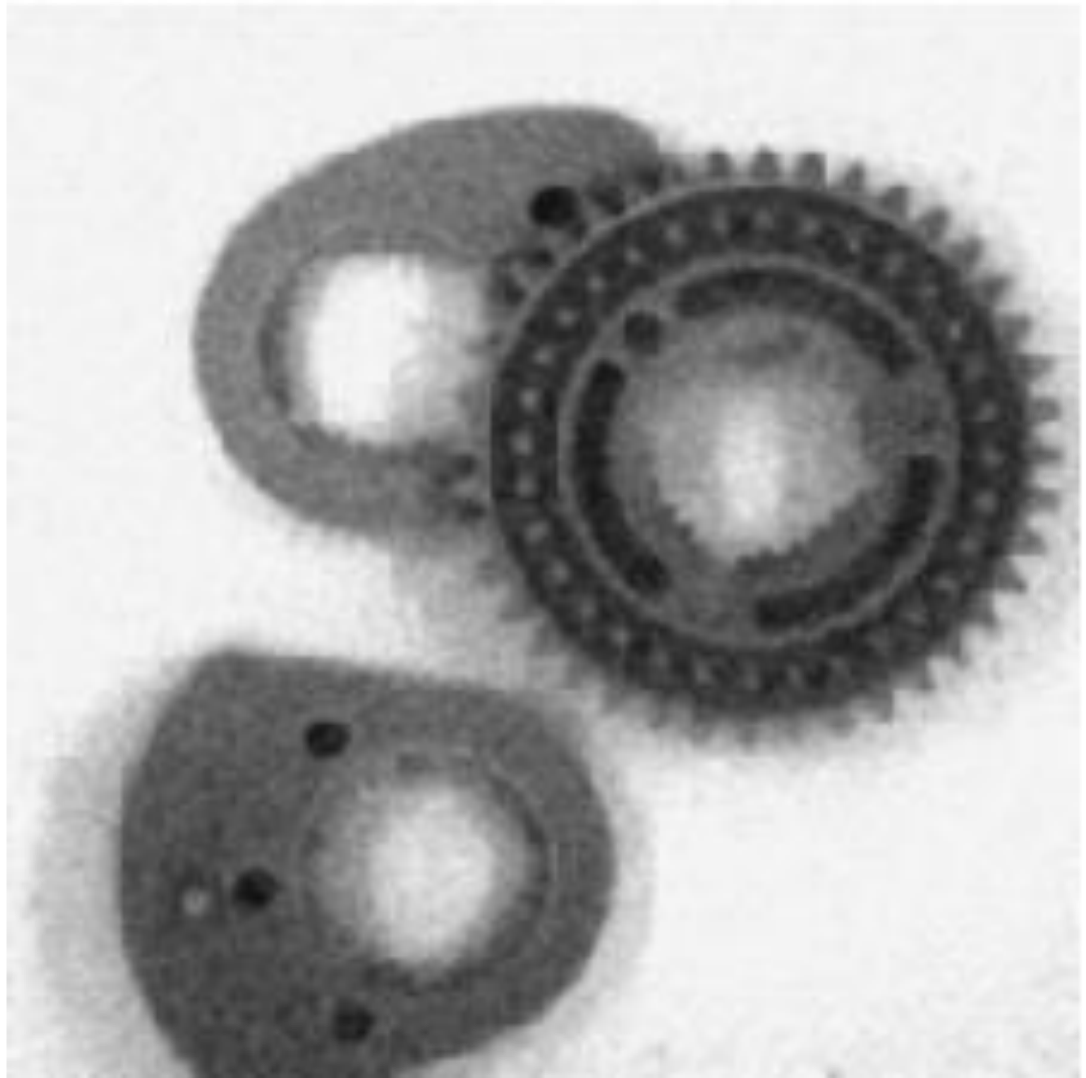
1	2	1
2	4	2
1	2	1

Also separable



MTF : $(2+2\cos(2\pi u))(2+2\cos(2\pi v))$

Results of binomial filtering



Computer Vision

Limit of binomial filtering

f :

1	2	1
2	4	2
1	2	1

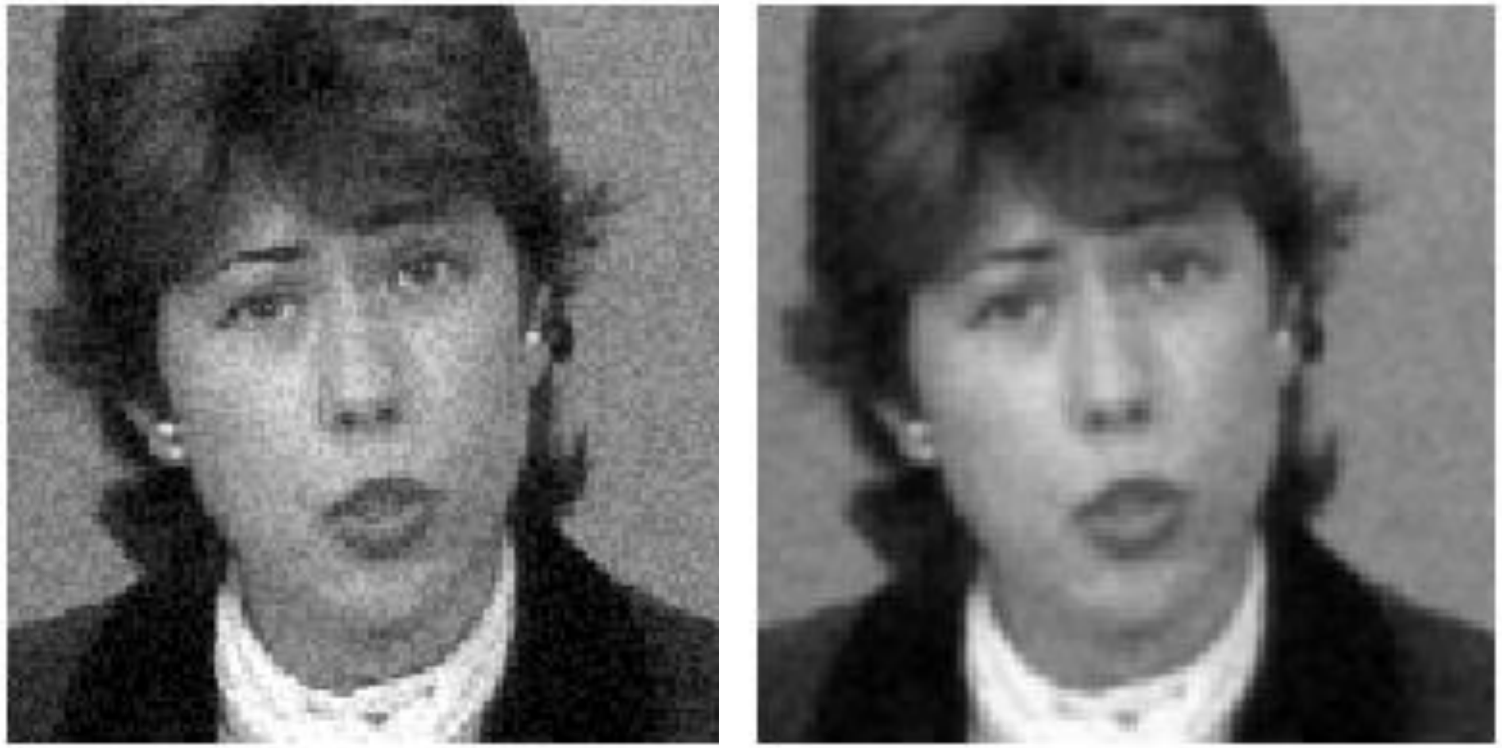
$$f(x,y) * f(x,y) * \cdots * f(x,y) = f^n(x,y)$$

$$f^n(x,y) \to a \exp\left(\frac{\|(x,y)\|^2}{b}\right), \quad \text{as } n \to \infty$$

Gaussian with b controlling the amount of smoothing

Gaussian smoothing

Gaussian is limit case of binomial filters



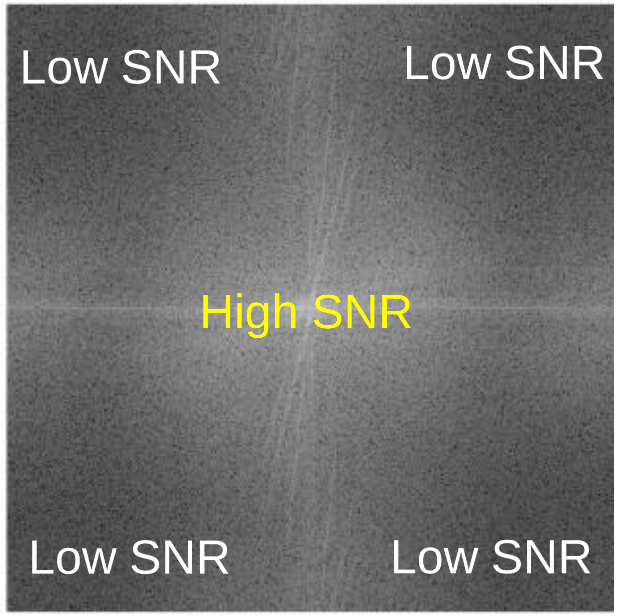
noise gone, no ripples, but still blurred…

Actually linear filters cannot solve this problem

Some notes on implementation

- separable filters can be implemented efficiently
- large filters through multiplication in the frequency domain
- integer mask coefficients increase efficiency powers of 2 can be generated using shift operations
- In Gaussian filter increasing b (the standard deviation) leads to more smoothing and blurring

Questions



Can convolutional filters do a perfect job?

Can they separate edge information from noise in the higher frequency components?

Why?

Noise suppresion

- In general specific methods for specific types of noise
- We only consider 2 general options here:

1. Convolutional linear filters – low-pass convolutional filters
2. Non-linear filters - edge-preserving filters
    a. Median
    b. Anisotropic diffusion

Median filtering: principle

- Non-linear filter

- Simple method:
    1. Rank-order neighborhood intensities in a patch of the image
    2. Take middle value and assign it to the patch center
    3. Go over all the image in a sliding window

- No new grey levels will emerge.

Median filtering – main advantage “odd-man-out”

advantage of this type of filter is its “odd-man-out” effect

e.g.

1,1,1,7,1,1,1,1

↓

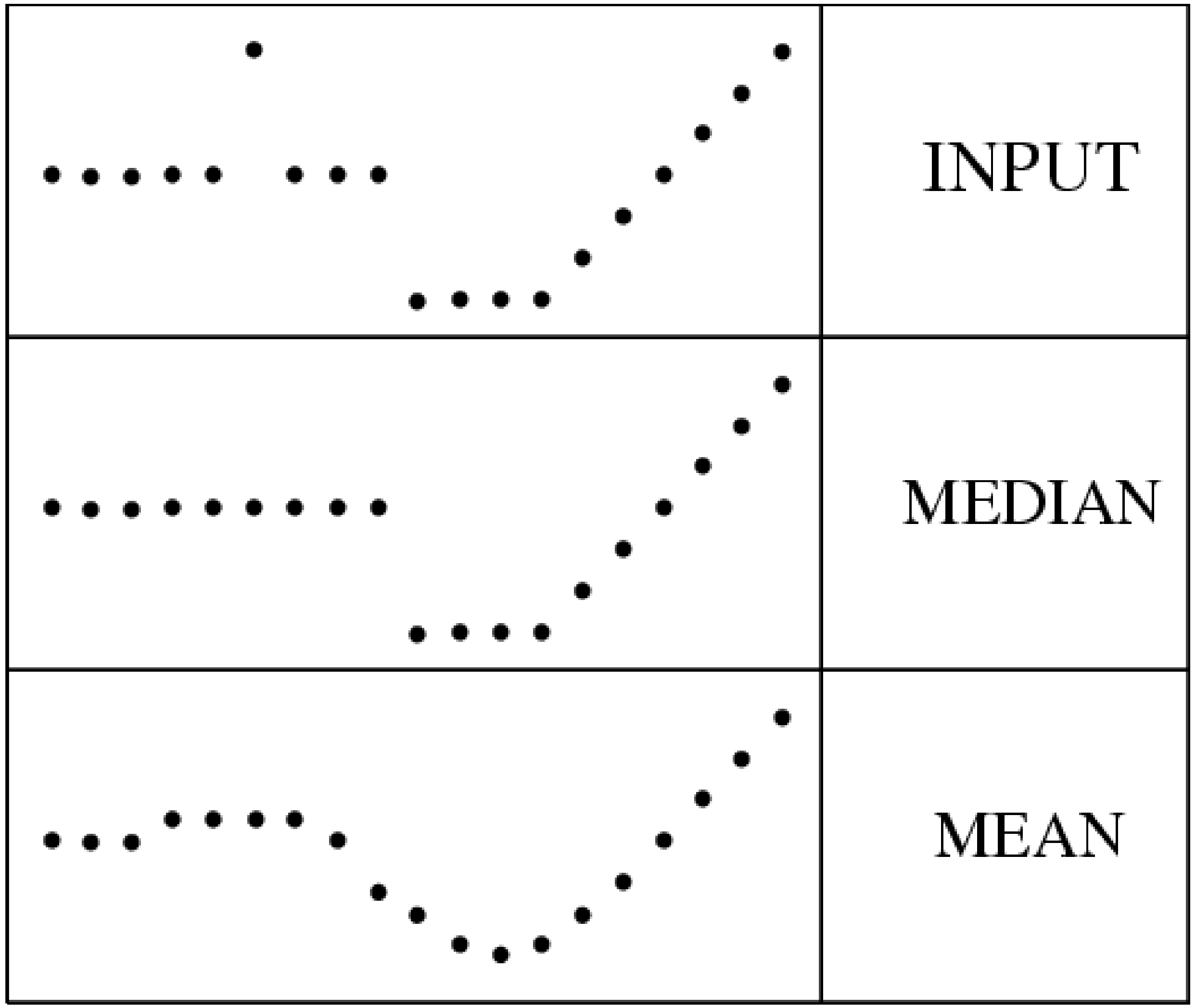
?,1,1,1.1,1,1,?

Example showing the advantage

Notice that the outlier is gone and sharp transitions (edge) are preserved

Median filtering with a patch width of 5

Mean filtering with a box width of 5

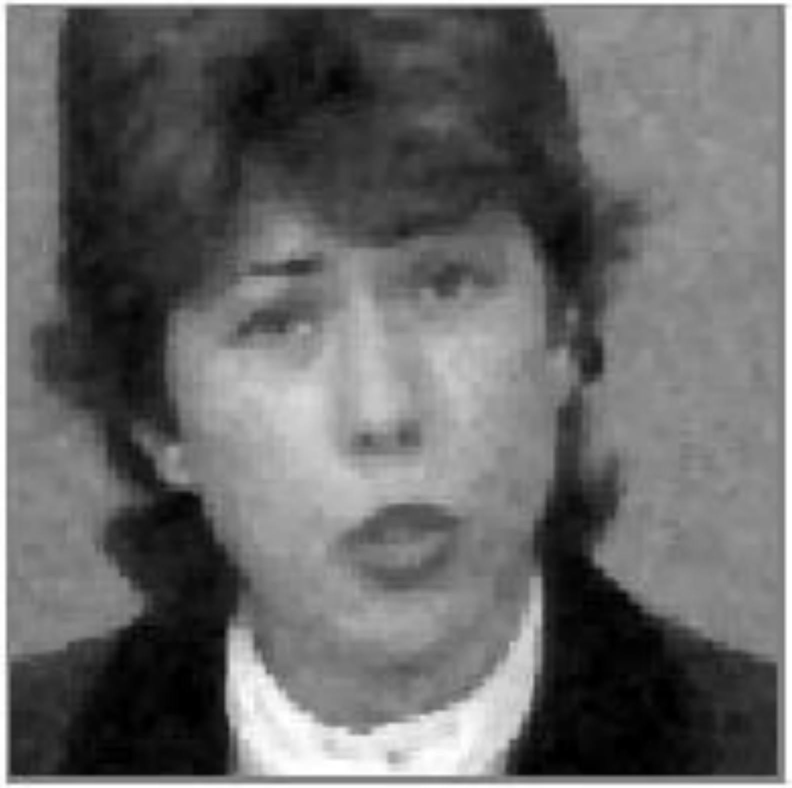


Median filtering – is it the solution to our blurring problem?

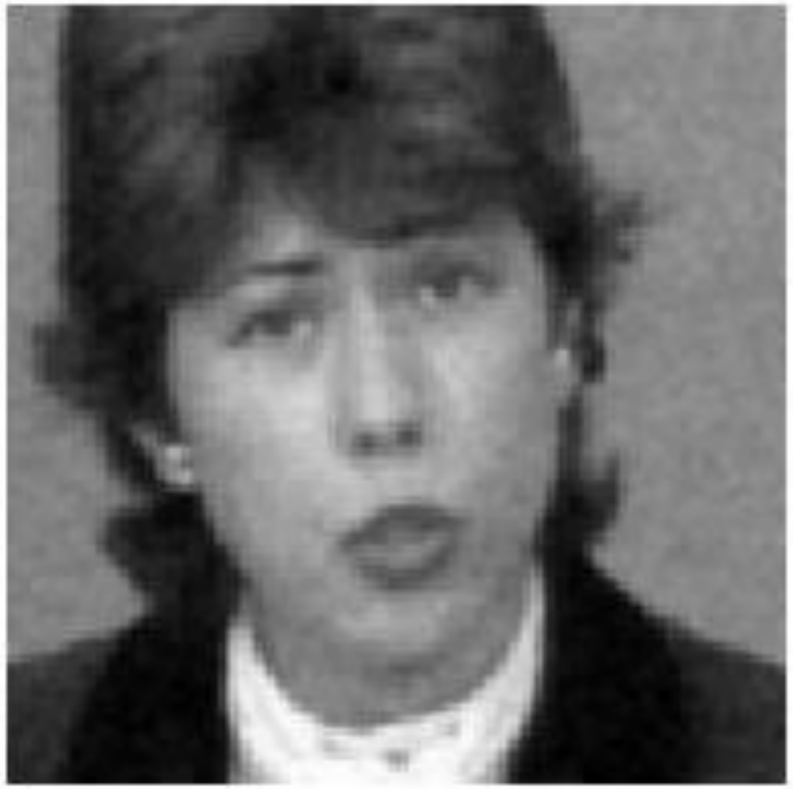
- median completely discards the spike, linear filter always responds to all aspects. Great for robustness to outliers and salt-and-pepper type noise
- median filter preserves discontinuities, linear filter produces rounding-off effects. Great for preserving sharp transitions, high frequency components and, essentially, edges and corners.
- DON'T become all too optimistic

Median filtering results

3 x 3 median filter :



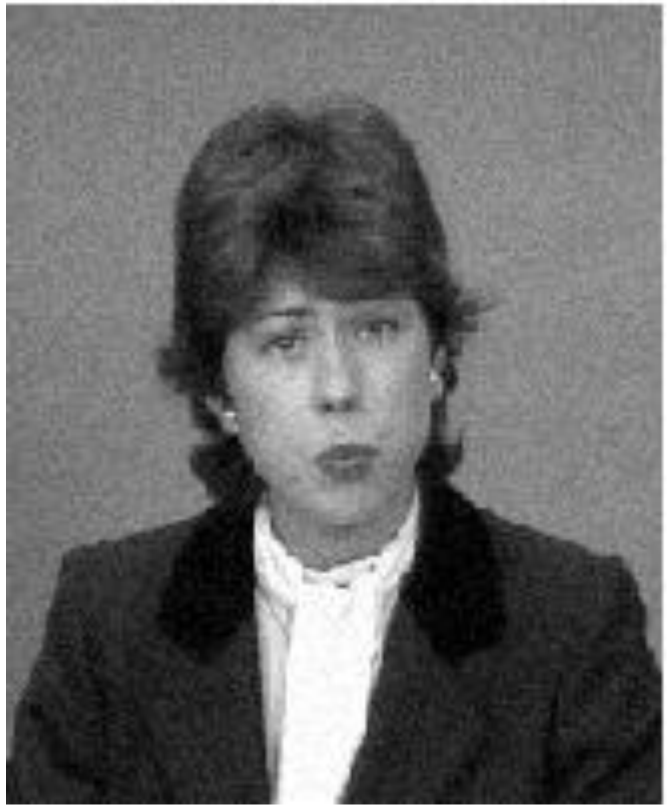
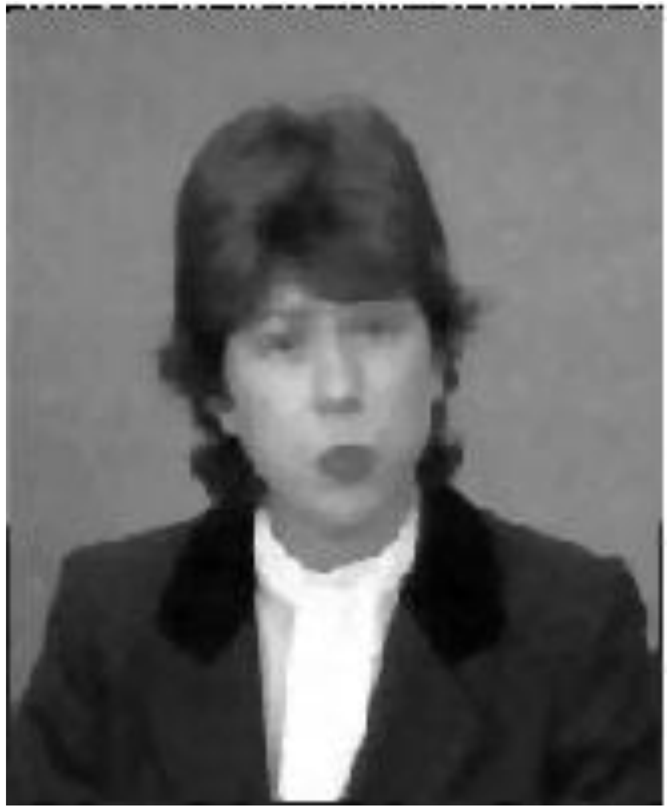
Comparison with Gaussian :



sharpens edges, destroys edge cusps and protrusions

Further results

10 times 3 X 3 median

patchy effect
important details lost (e.g. ear-ring)

Question

For what types of noise would you clearly prefer median filtering over Gaussian filtering?

a) Gaussian noise, i.e. noise distributed by independent normal distribution
b) Salt and pepper noise
c) Uniform noise, i.e. distributed by uniform distribution
d) Exponential noise model
e) Rayleigh noise

Noise suppresion

- In general specific methods for specific types of noise
- We only consider 2 general options here:

1. Convolutional linear filters – low-pass convolutional filters

2. Non-linear filters - edge-preserving filters
   a. Median
   b. Anisotropic diffusion

Anistropic diffusion: principle

- Non-linear filter

- More complicated method:

  1. Gaussian smoothing across homogeneous intensity areas

  2. No smoothing across edges

Gaussian filter revisited

The diffusion equation

$$\frac{\partial f(\vec{x},t)}{\partial t} = \nabla \cdot (c(\vec{x},t)\nabla f(\vec{x},t))$$

Initial/Boundary conditions

$$f(\vec{x},0) = i(x,y), \text{ for } \vec{x} \in \Omega$$
$$f(\vec{x},t) = 0, \text{ for } \vec{x} \in \delta(\Omega)$$

If $c(\vec{x},t) = c$

$$\frac{\partial f(\vec{x},t)}{\partial t} = c\Delta f(\vec{x},t)$$ in1D: $$\frac{\partial f(x,t)}{\partial t} = c\frac{\partial^2 f(x,t)}{\partial x^2}$$

Solution is a convolution!

$$f(\vec{x},t) = f(\vec{x},0) * g(\vec{x},t) = i(\vec{x}) * g(\vec{x},t)$$

Diffusion as Gaussian low-pass filter

$$f(\vec{x},t) = i(\vec{x}) * \frac{1}{(2\pi)^{d/2}\sqrt{ct}} \exp\left\{-\frac{\vec{x}\cdot\vec{x}}{4ct}\right\}$$

Gaussian filter with time dependent standard deviation:

$$\sigma = \sqrt{2ct}$$

Nonlinear version can change the width of the filter locally

$$c(\vec{x},t) = c(f(\vec{x},t))$$

Specifically dependening on the edge information through gradients

$$c(\vec{x},t) = c(|\nabla f(\vec{x},t)|)$$

Selection of diffusion coefficient

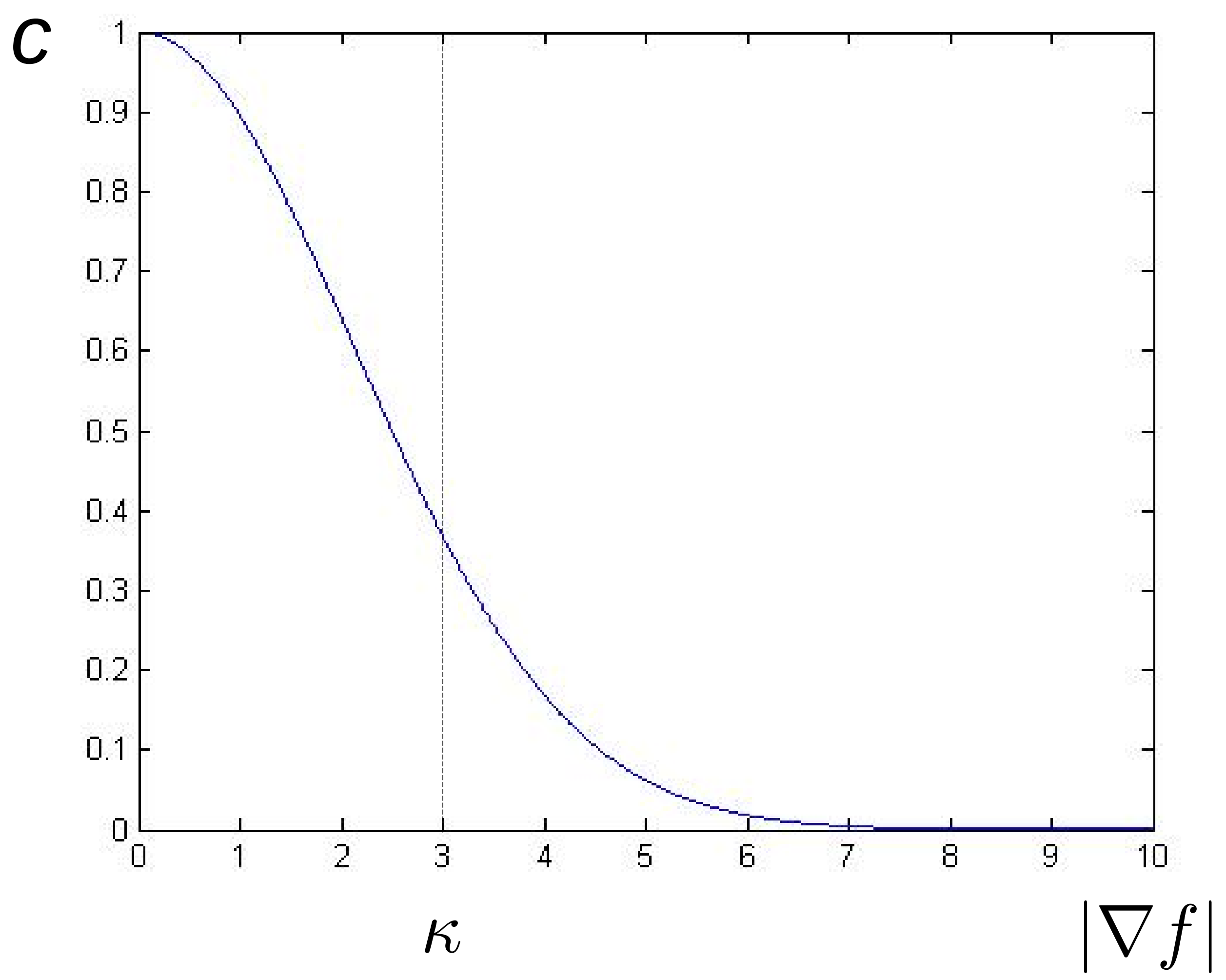
$$c(|\nabla f(\vec{x},t)|) = \exp\left\{-\frac{|\nabla f|^2}{2\kappa^2}\right\}$$

or

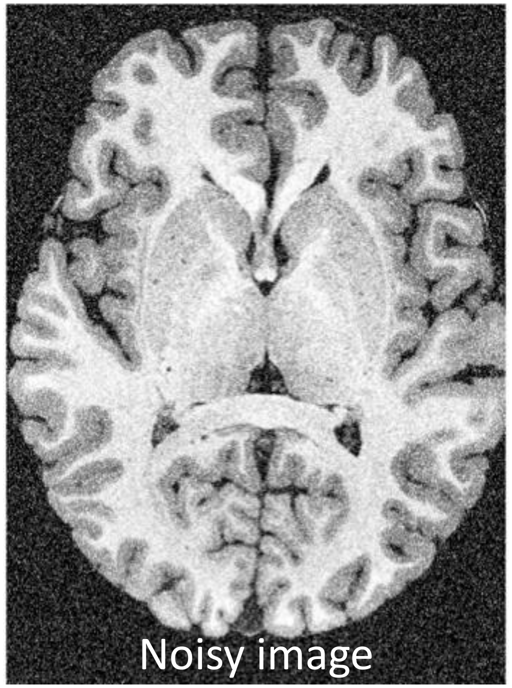
$$c(|\nabla f(\vec{x},t)|) = \frac{1}{1+\left(\frac{|\nabla f|}{\kappa}\right)^2}$$

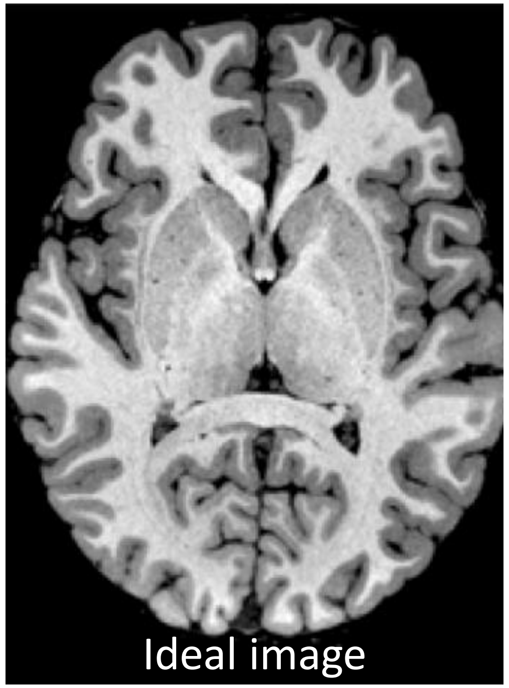
κ controls the contrast to be preserved by smooting
actually edge sharpening happens

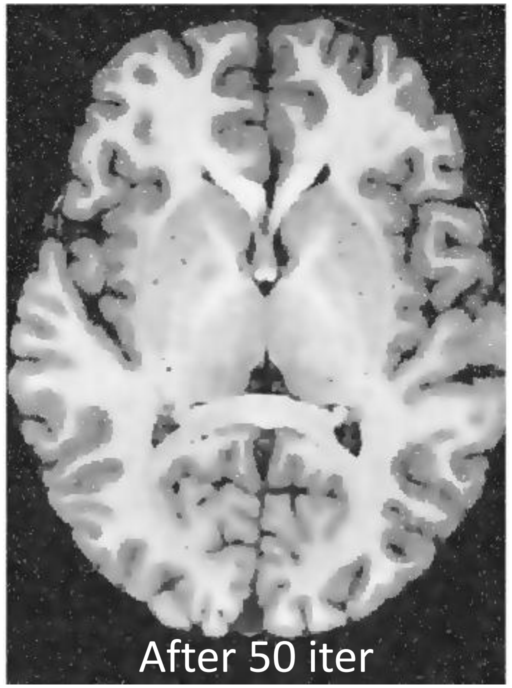
Dependence on contrast

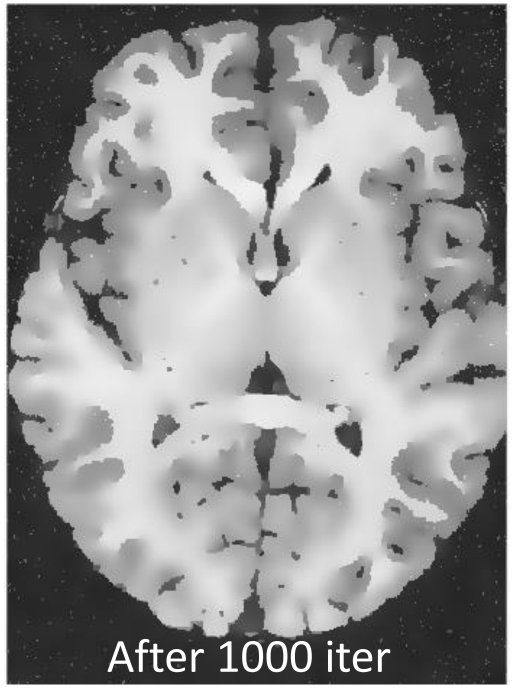


Computer Vision

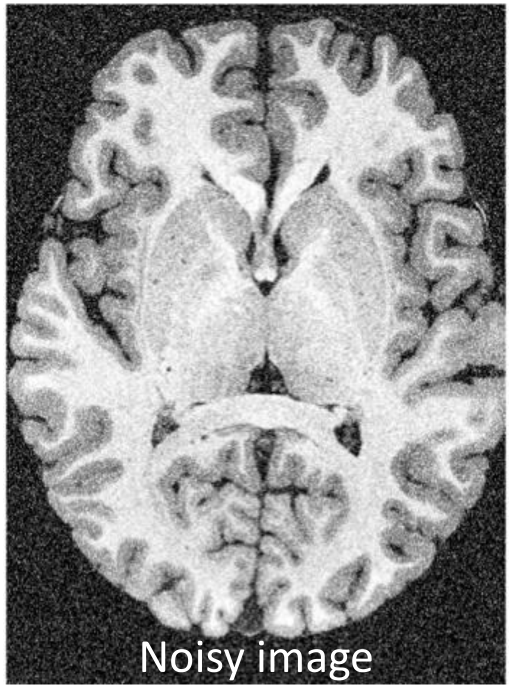


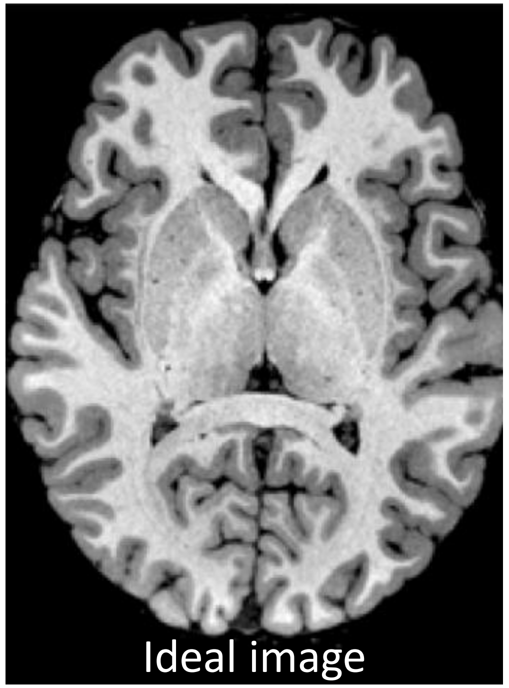


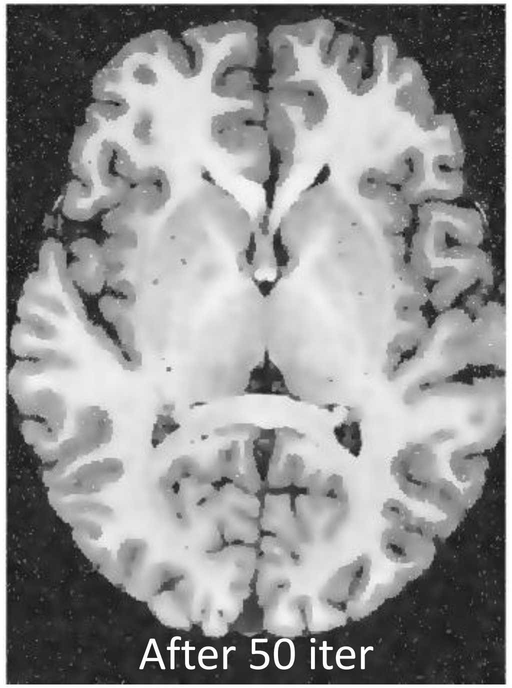


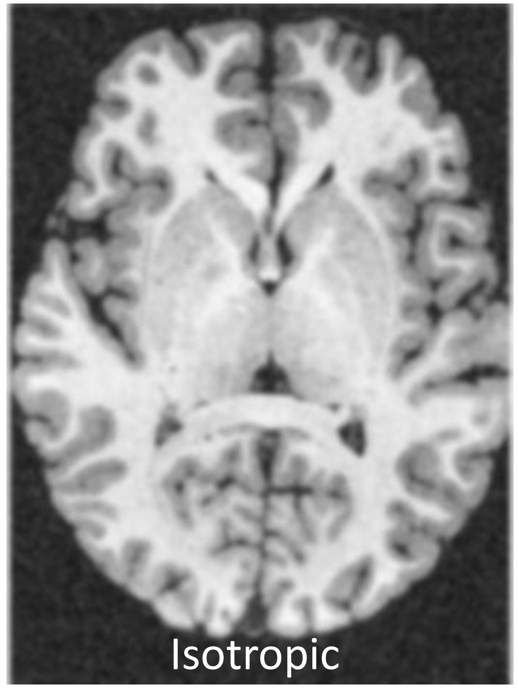


Computer Vision

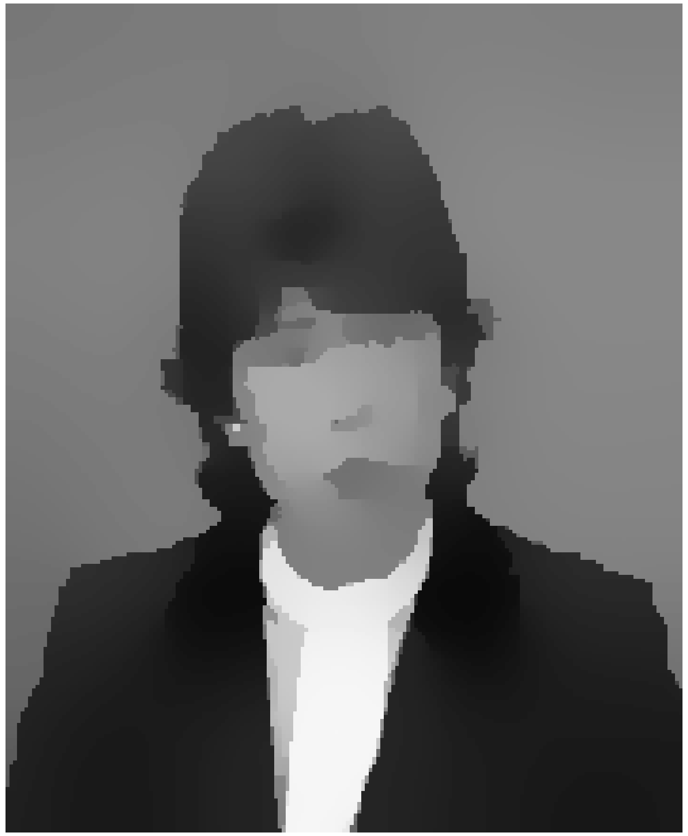






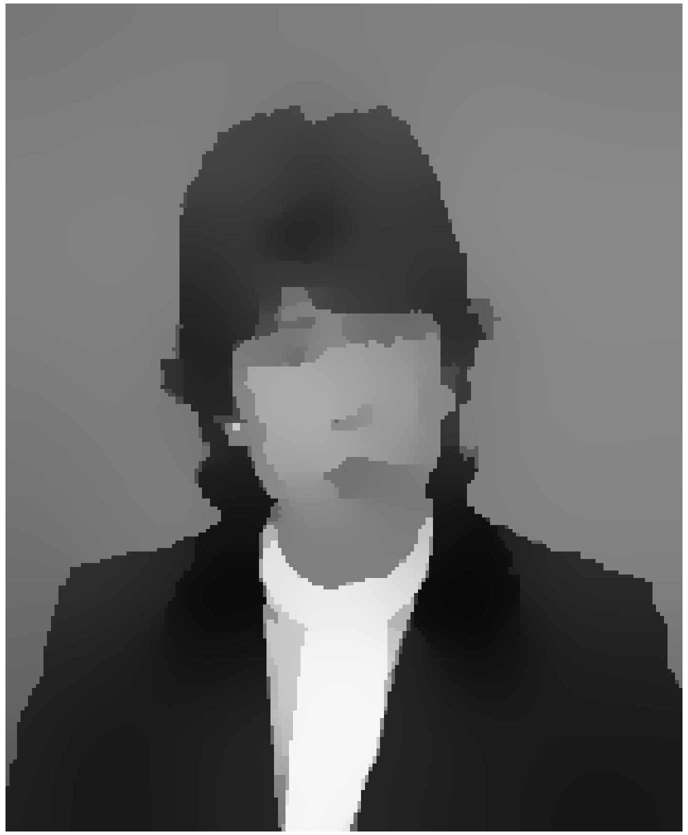


Unrestrained anisotropic diffusion



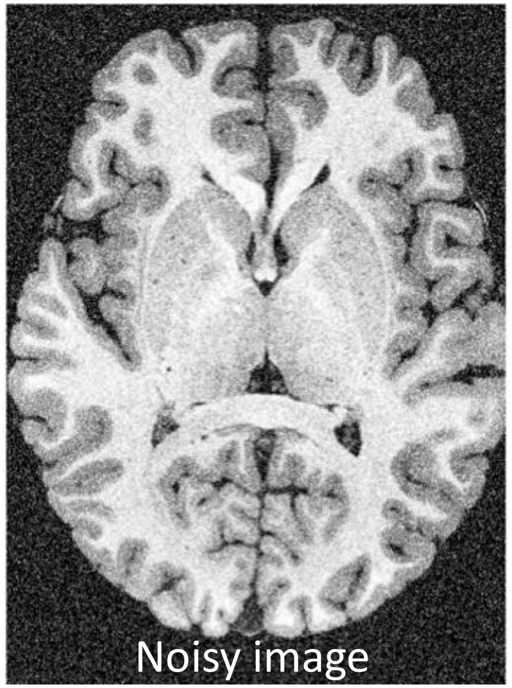
End state is homogeneous

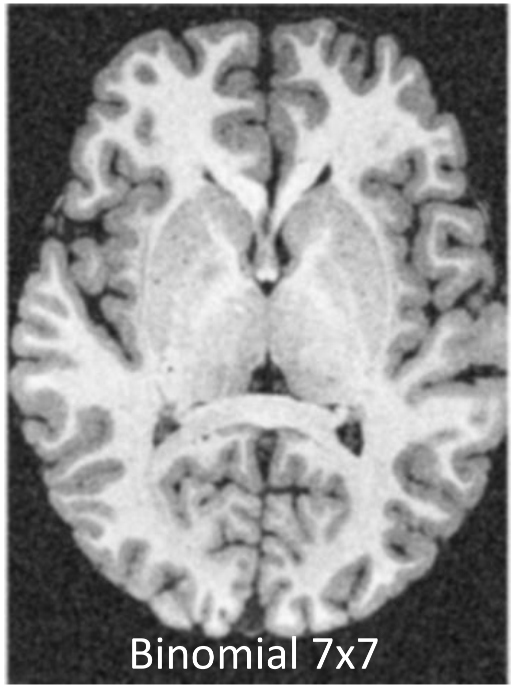
Restraining anisotropic diffusion

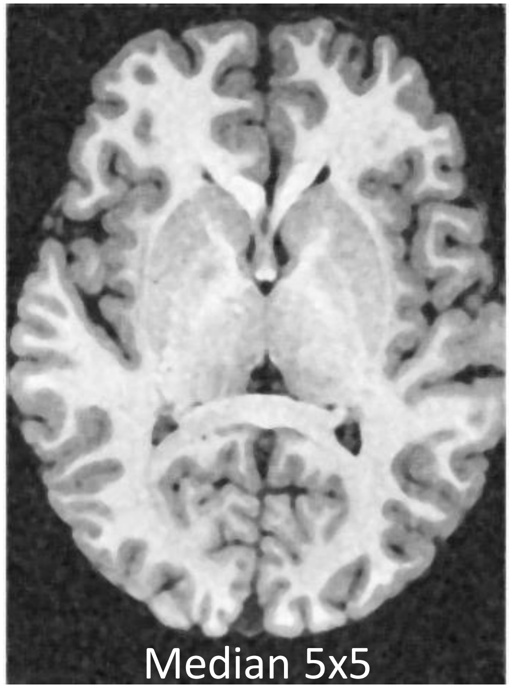


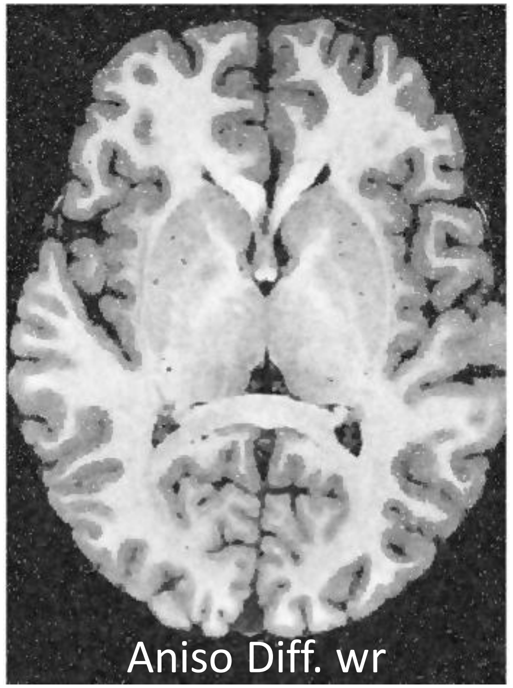
adding restraining force :

$$\frac{\partial f}{\partial t} = \Delta \cdot (c(|\nabla f|)\nabla f) - \frac{1}{\sigma^2}(f - i)$$


Noisy image


Binomial 7x7


Median 5x5


Aniso Diff. wr

Anisotropic diffusion – numerical solutions

When c is not a constant solution is found through solving the equation

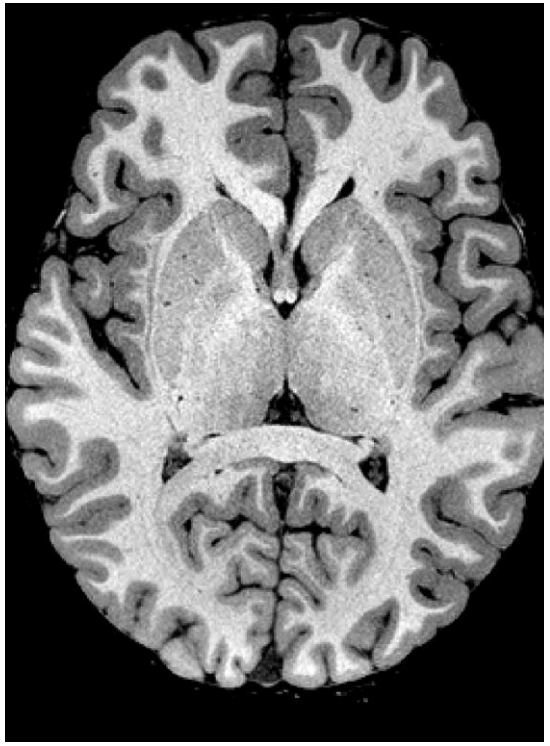
$$\frac{\partial f(\vec{x},t)}{\partial t} = \nabla \cdot (c(\vec{x},t)\nabla f(\vec{x},t))$$

Partial differential equation

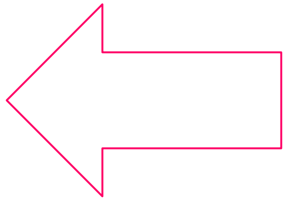
- Numerical solutions through discretizing the differential operators and integrating

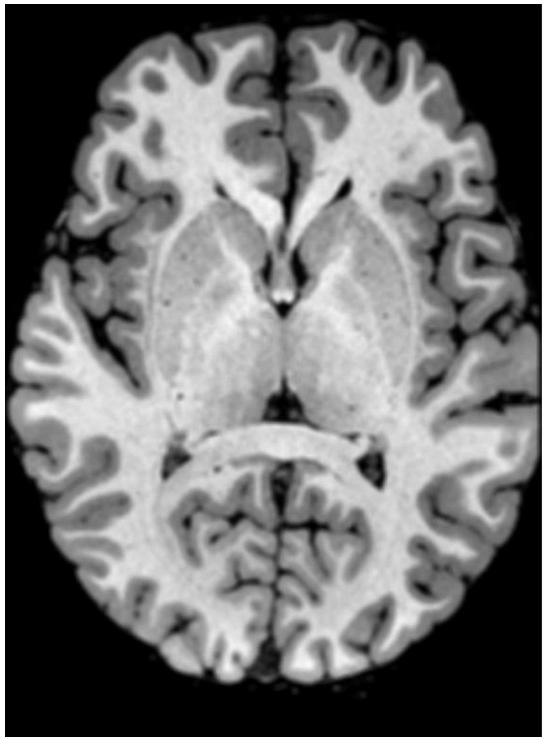
- Finite differences in space and integration in time

Computer Vision



Deblurring





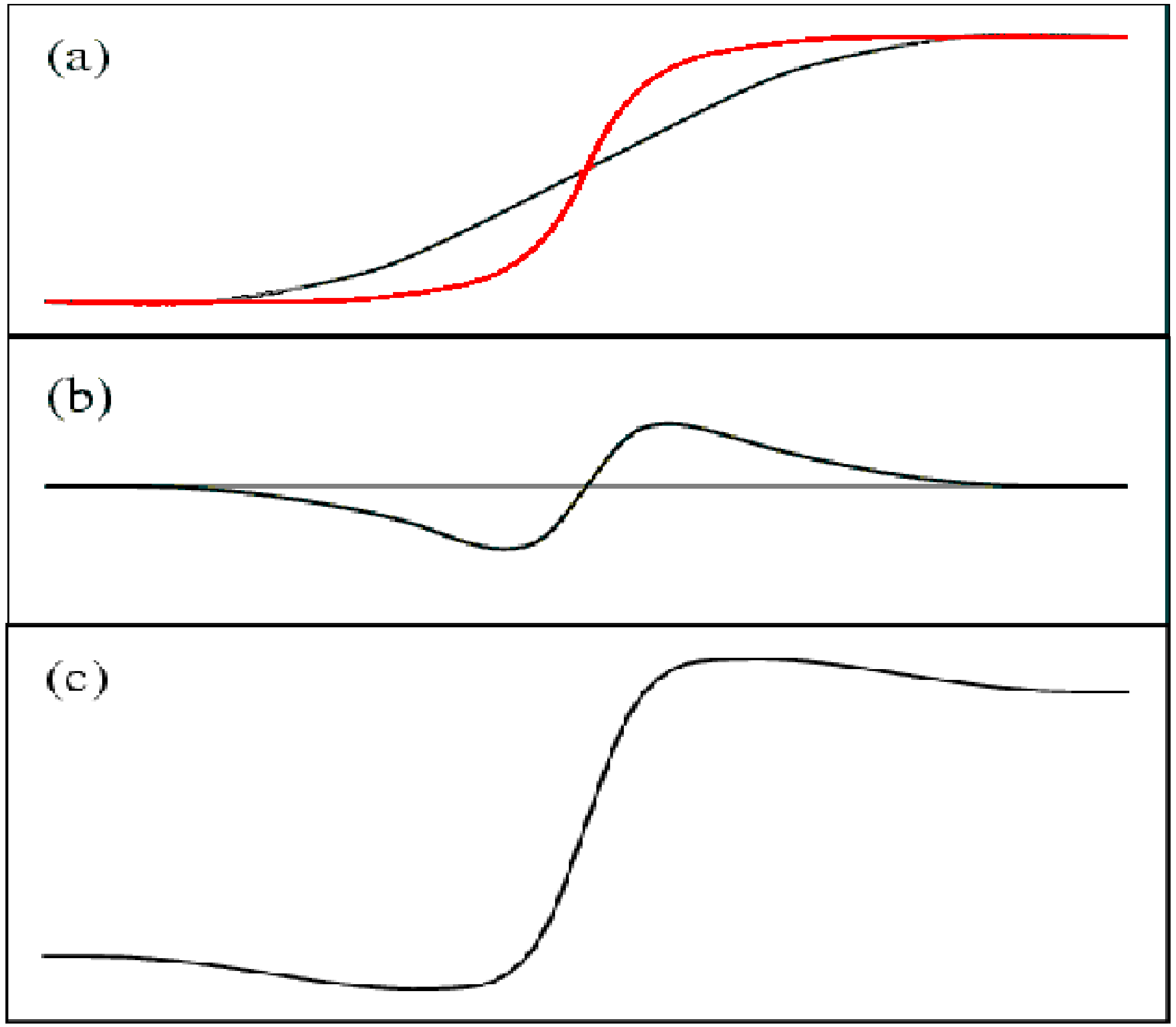
Original Image
What we want

Blurred image
What we observe

Approach I: Unsharp masking

- simple but effective method
- image independent
- linear
- used e.g. in photocopiers and scanners

Unsharp masking - sketch



red = original
black = smoothed

orginal – smooth

original + difference

Unsharp masking - principle

Interpret blurred image as snapshot of diffusion process

$$\frac{\partial f}{\partial t} = c(\nabla^2 f)$$

In a first order approximation, we can write

$$f(x, y, t) \approx f(x, y, 0) + \frac{\partial f}{\partial t} t$$

Hence,

$$f(x, y, 0) \approx f(x, y, t) - \frac{\partial f}{\partial t} t = f(x, y, t) - ct\nabla^2 f$$

Unsharp masking produces *o* from *i*

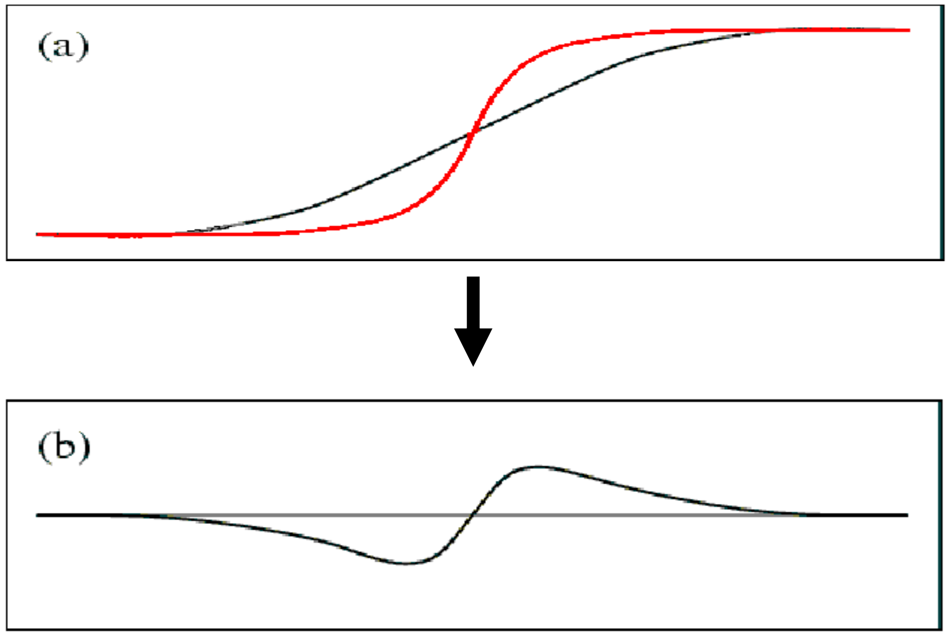
$$o = i - k\nabla^2 i$$

with *k* a well-chosen constant

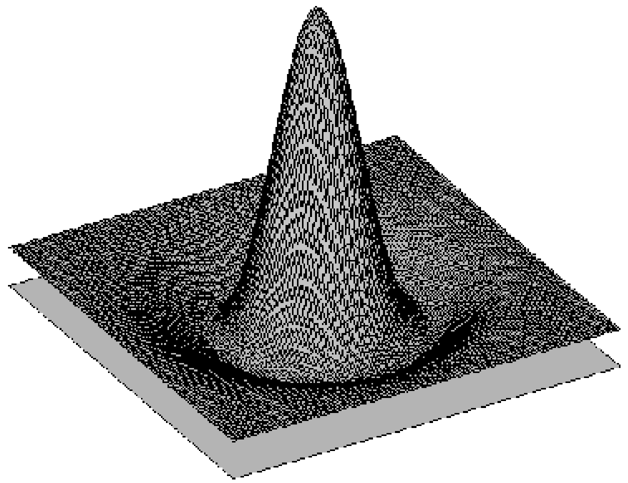
Need to estimate $\nabla^2 i(x, y)$

DOG (Difference-of-Gaussians) approximation for Laplacian :

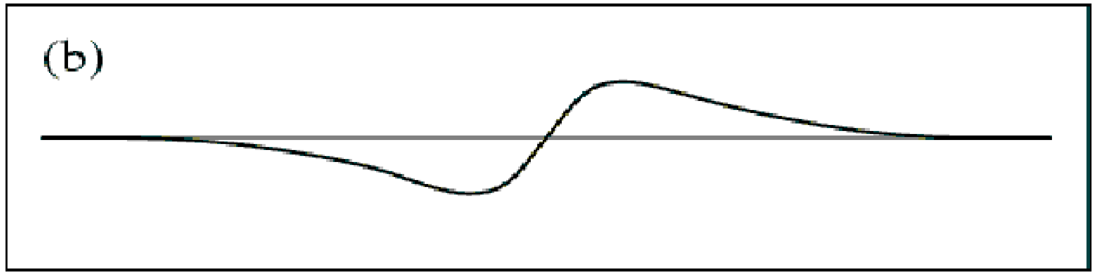
Our 1D example:



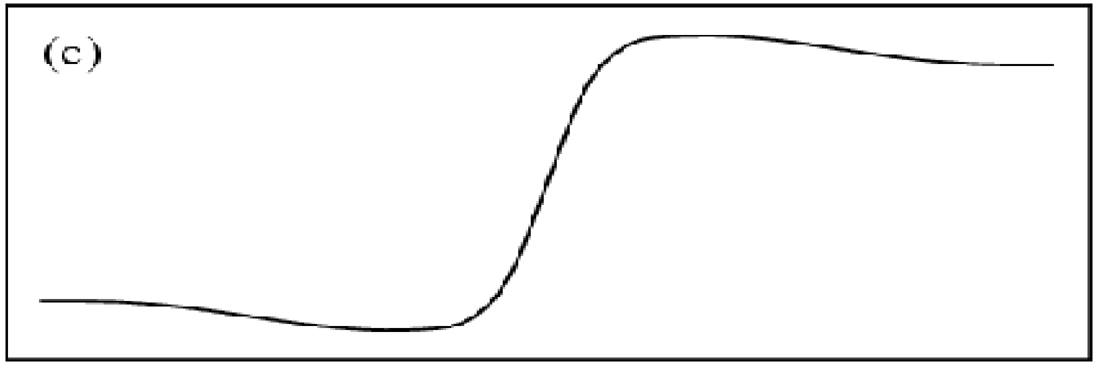
Convolution mask in 2D:



Unsharp masking analysis



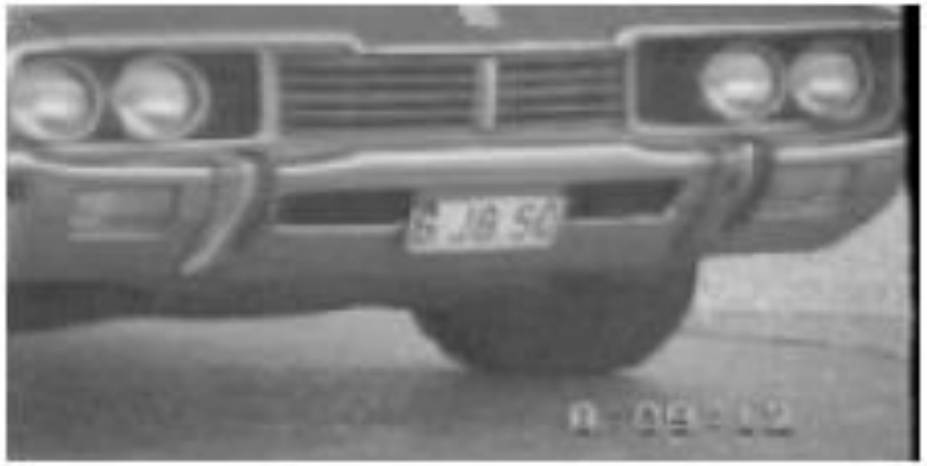
$$\downarrow \quad o = i - k\nabla^2 i$$

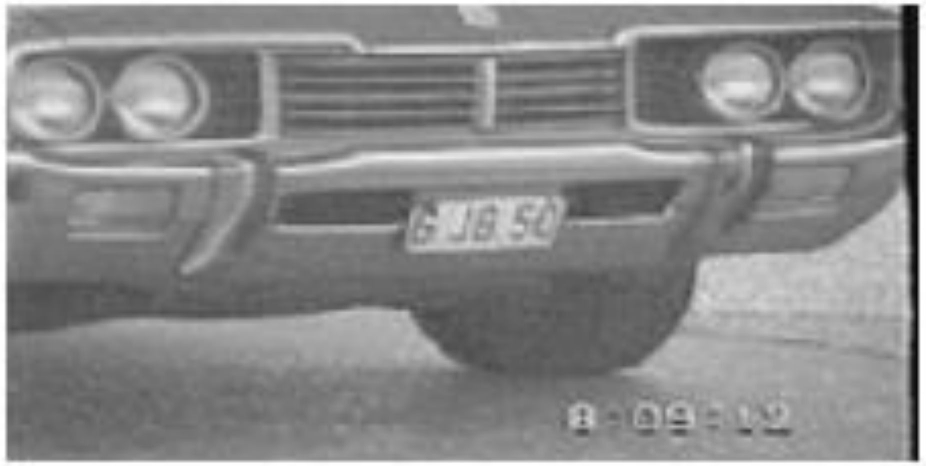


The edge profile becomes steeper, giving a sharper impression

Under-and overshoots flanking the edge further increase the impression of image sharpness

Unsharp masking results





Approach II: Inverse filtering

- Relies on system view of image processing
- Frequency domain technique
- Defined through Modulation Transfer Function
- Links to theoretically optimal approaches

Inverse filtering principle

Frequency domain technique

suppose you know the MTF ***B(u,v)*** of the blurring filter

$$f(x,y) = b(x,y) * i(x,y)$$

$$F(u,v) = B(u,v)I(u,v)$$

to undo its effect new filter with MTF ***B' (u,v)*** such that

$$B'(u,v)B(u,v) = 1$$

$$I(u,v) = B'(u,v)F(u,v)$$

Inverse filtering principle

$$B'(u,v) = 1/B(u,v)$$

For additive noise after filtering

$$F(u,v) = B(u,v)I(u,v) + N(u,v)$$

Result of inverse filter

$$F(u,v)B'(u,v) = I(u,v) + N(u,v)/B(u,v)$$

Inverse problem’s main issue

$$F(u,v) = B(u,v)I(u,v) + N(u,v)$$

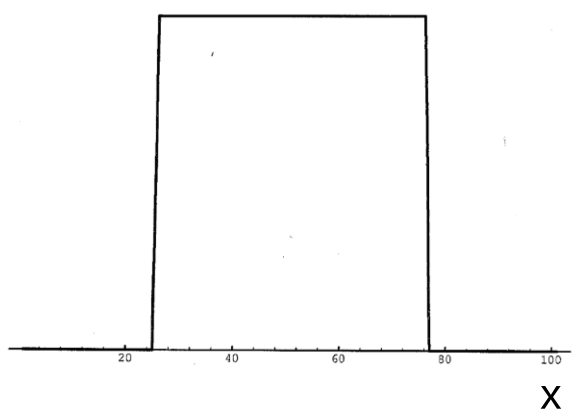
- Frequencies with *B (u,v)* = 0
  Information fully lost during filtering
  Cannot be recovered
  Inverse filter is ill-defined

$$F(u,v)B'(u,v) = I(u,v) + N(u,v)/B(u,v)$$

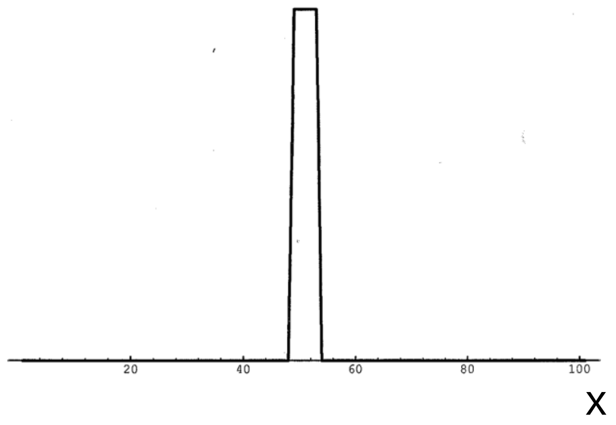
- Also problem with noise added after filtering
  B(u,v) is low = 1/*B(u,v)* is high,
  VERY strong noise amplification

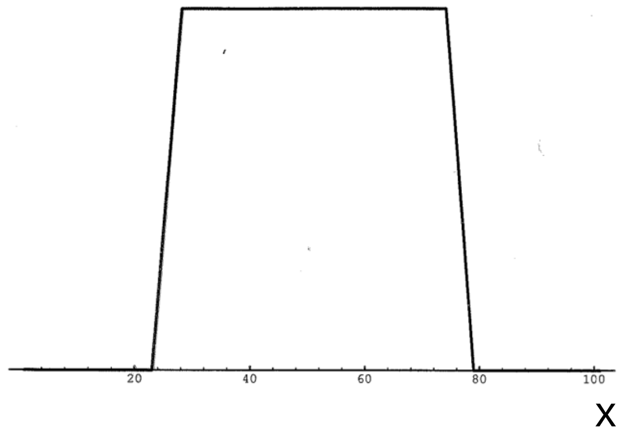
Computer Vision

1D example

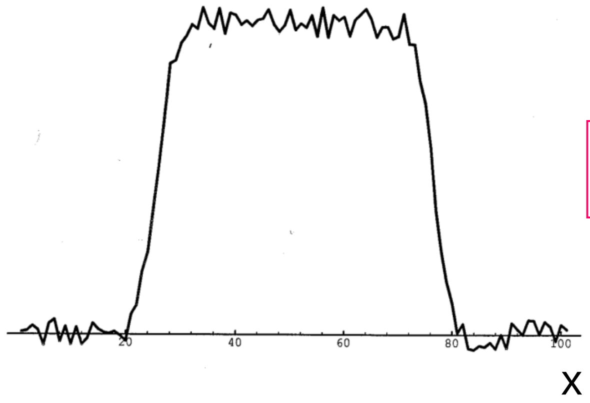


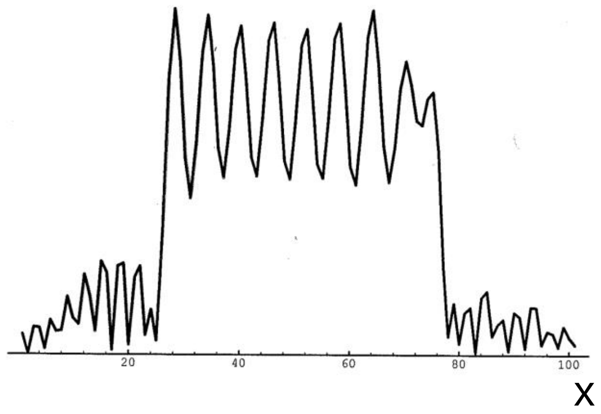
*

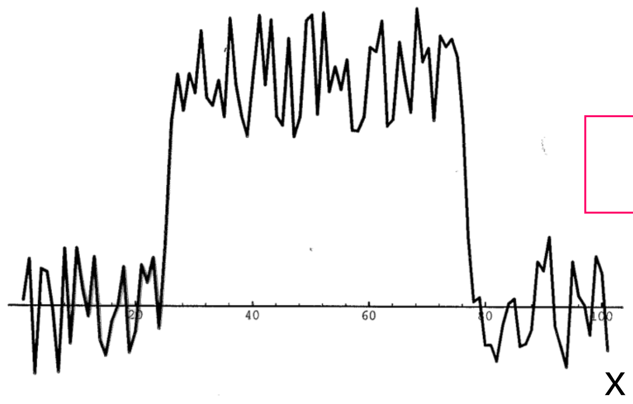


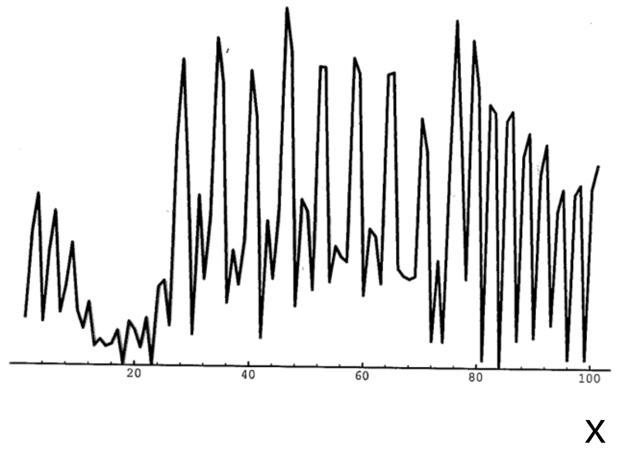


Deblurring the noisy version



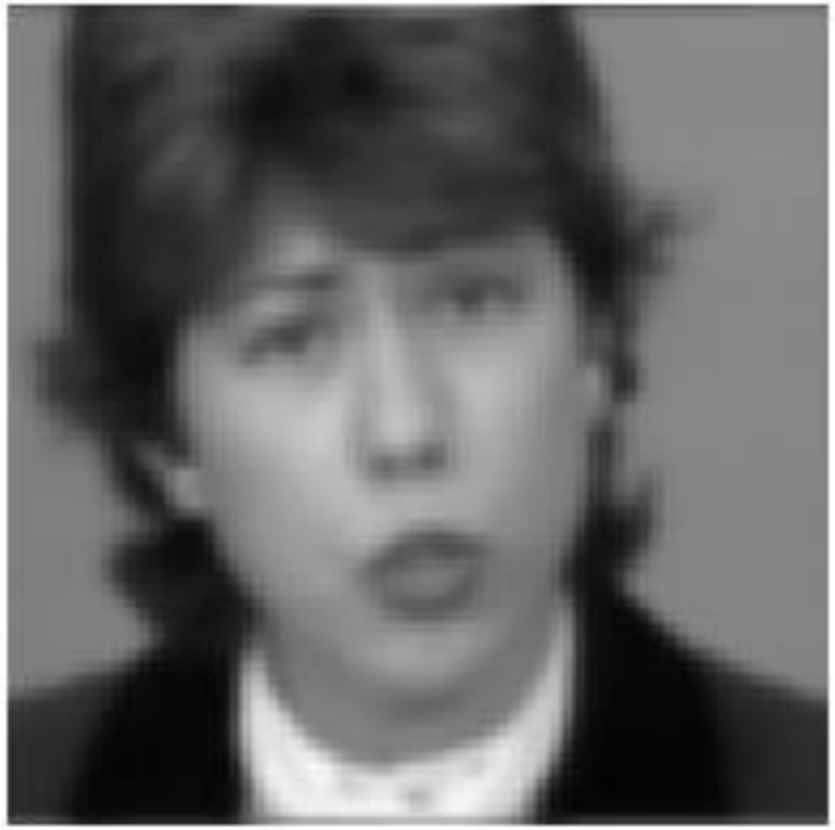




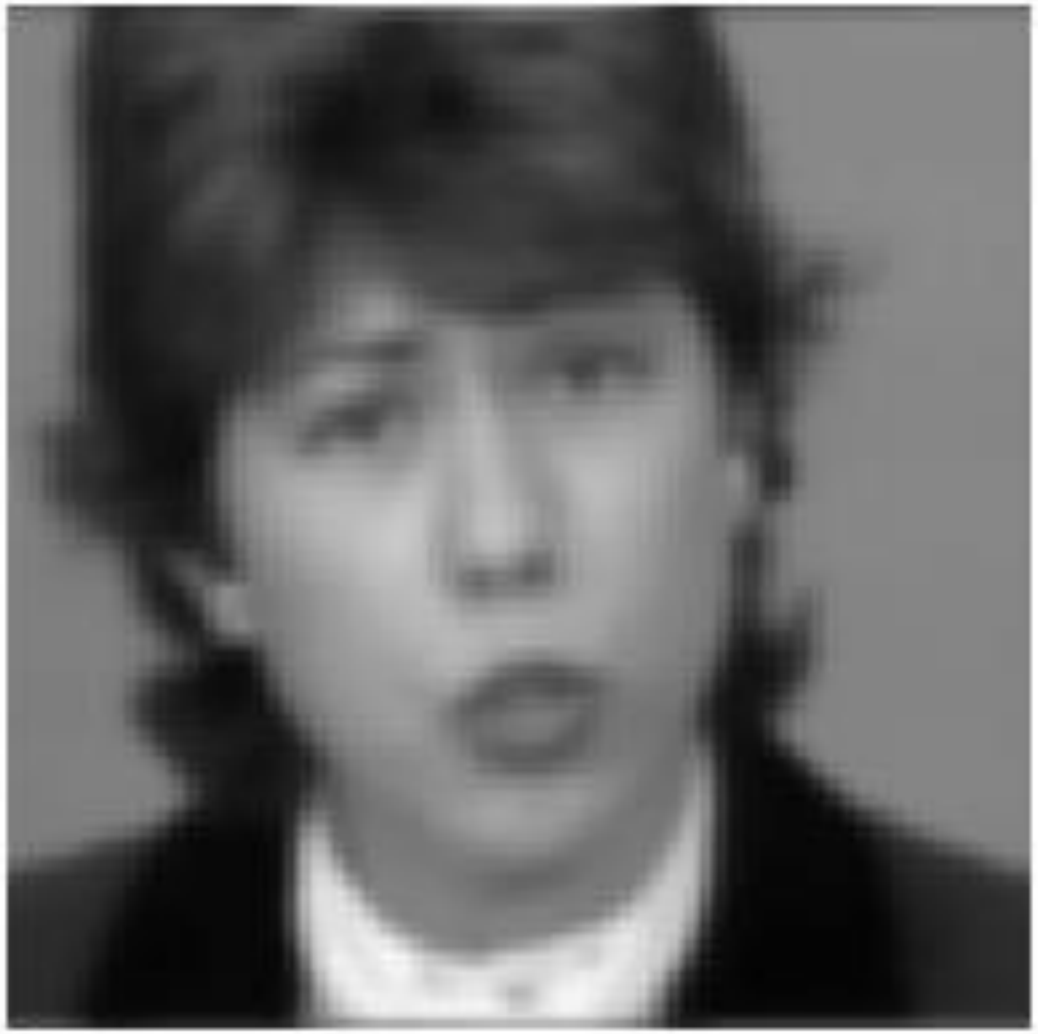


Inverse filtering example on an image

we will apply the method to a Gaussian smoothed example (σ = 16 pixels)



Result



noise leads to spurious high frequencies

Wiener filter

- Looking for the optimal filter to do the deblurring
- Consider the noise to avoid amplification
- A much better version of inverse filtering
- Optimization formulation
- Filter is given analytically in the Fourier Domain

Wiener filter and its behavior

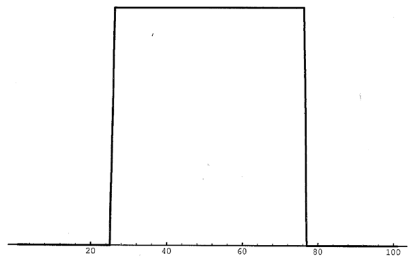
$$Wf(H) = H'(u,v) = \frac{H(u,v)}{H^*(u,v)H(u,v) + 1/\mathrm{SNR}}$$

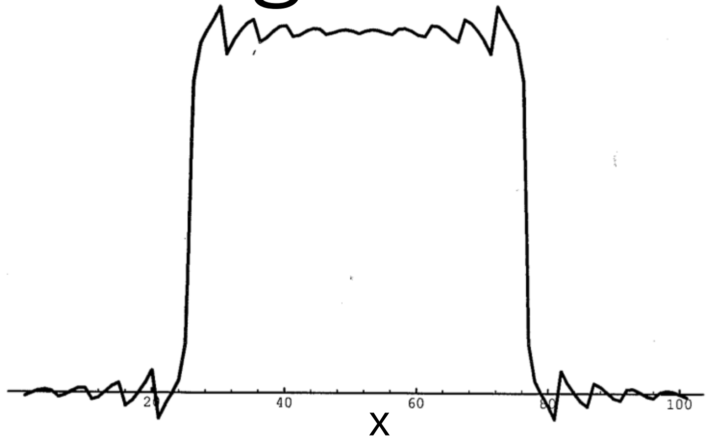
$$\mathrm{SNR} = \frac{\Phi_{ii}}{\Phi_{nn}}$$

- $H(u,v) = 0 \implies Wf(H) = 0$ ✓
- $\mathrm{SNR} \to \infty \implies 1/\mathrm{SNR} \to 0$, $Wf(H) \to \frac{1}{H}$ ✓
- $\mathrm{SNR} \to 0 \implies 1/\mathrm{SNR} \to \infty$, $Wf(H) \to 0$ ✓

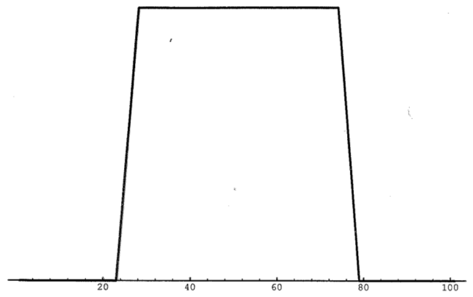
Computer Vision

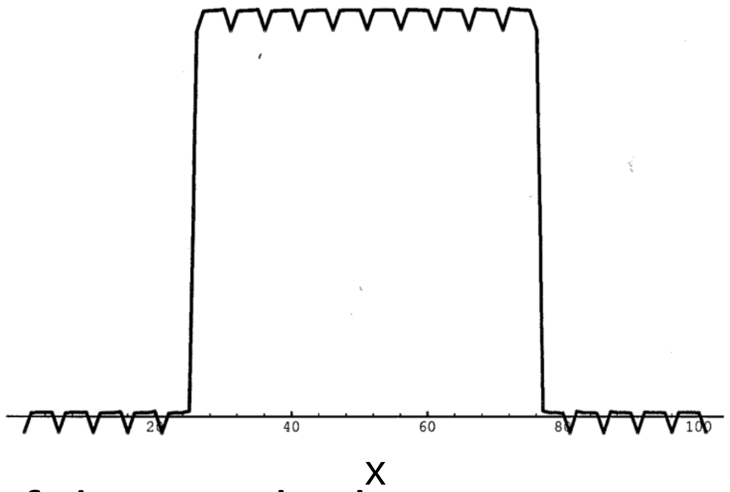
Deblurring noise-free signal





Medium confidence – medium SNR assumption

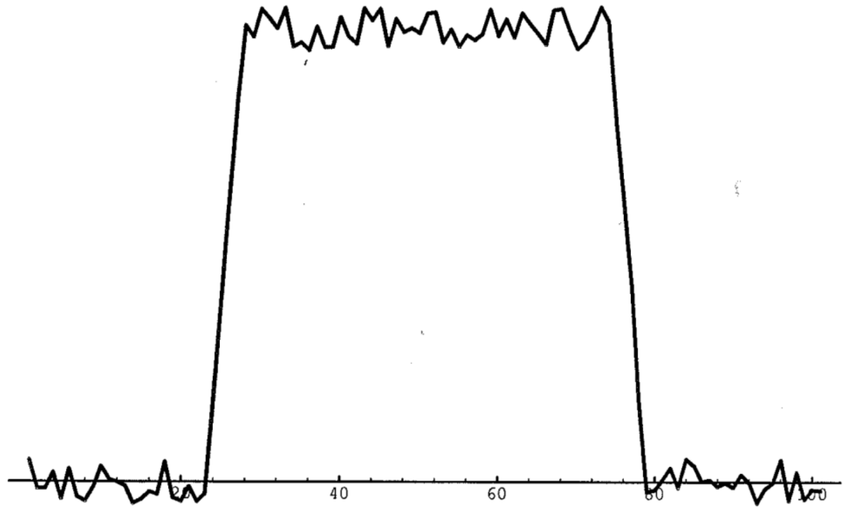


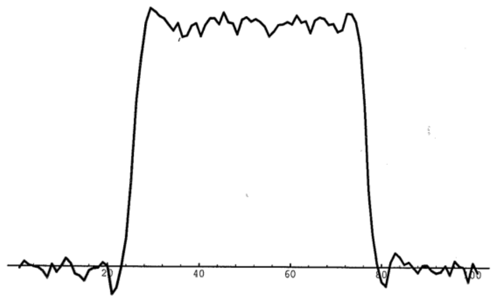


High confidence – high SNR assumption

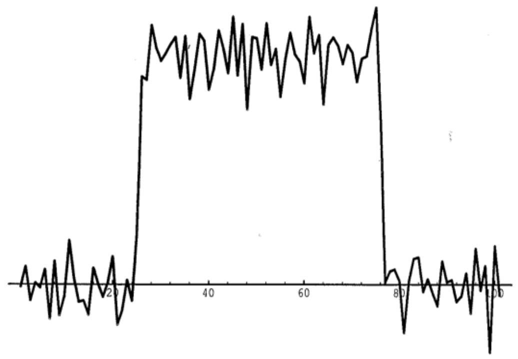
Computer Vision

Deblurring noisy signal

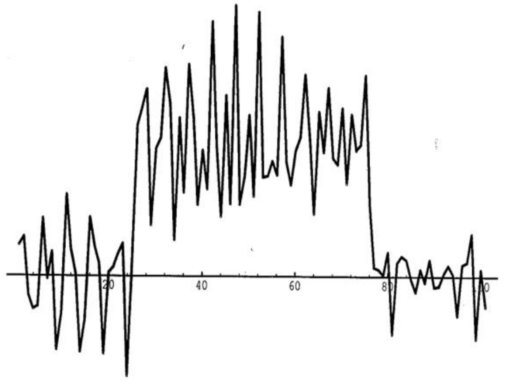




Low confidence

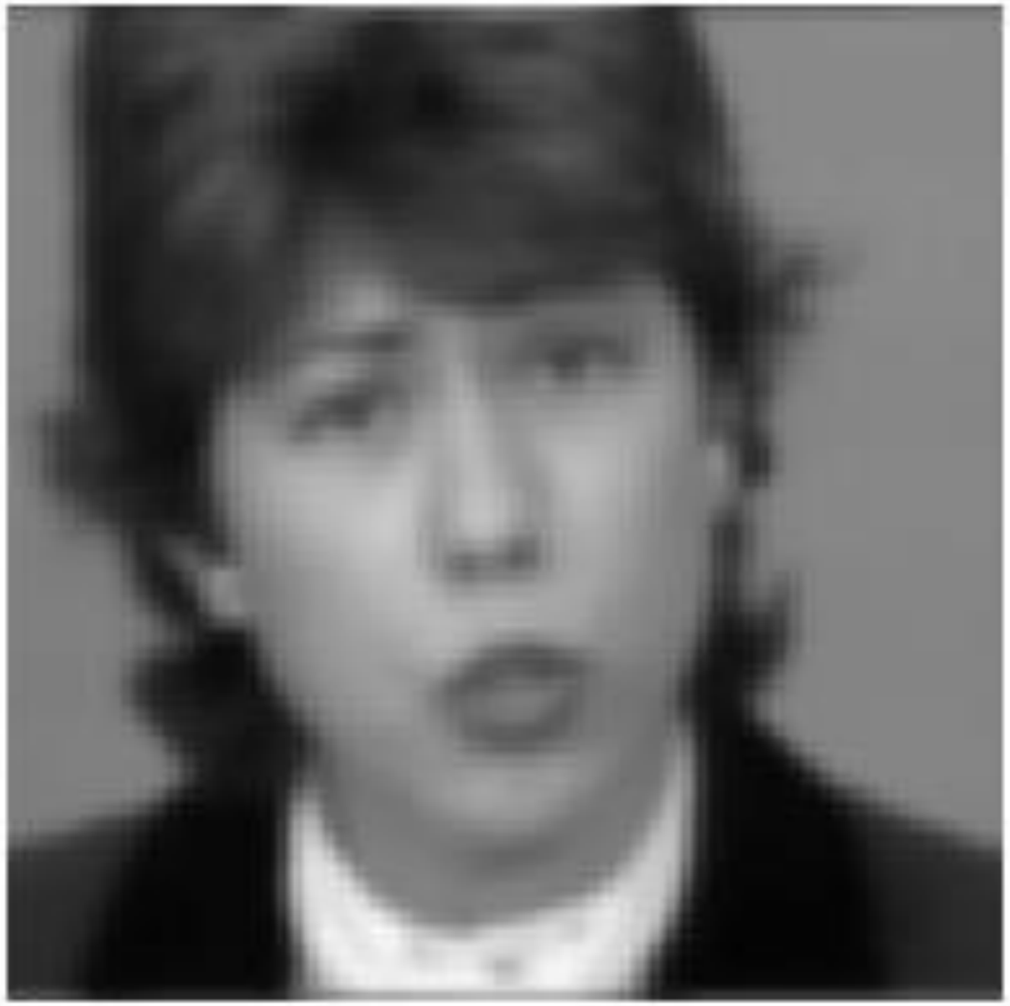


Medium confidence
Correct SNR



High confidence

Wiener filtering example



spurious high freq. eliminated, conservative

Problems in applying Wiener filtering

$$\begin{aligned} O(u,v) &= Wf(H)(H(u,v)I(u,v)) \\ &= (Wf(H)H(u,v))I(u,v) \end{aligned}$$

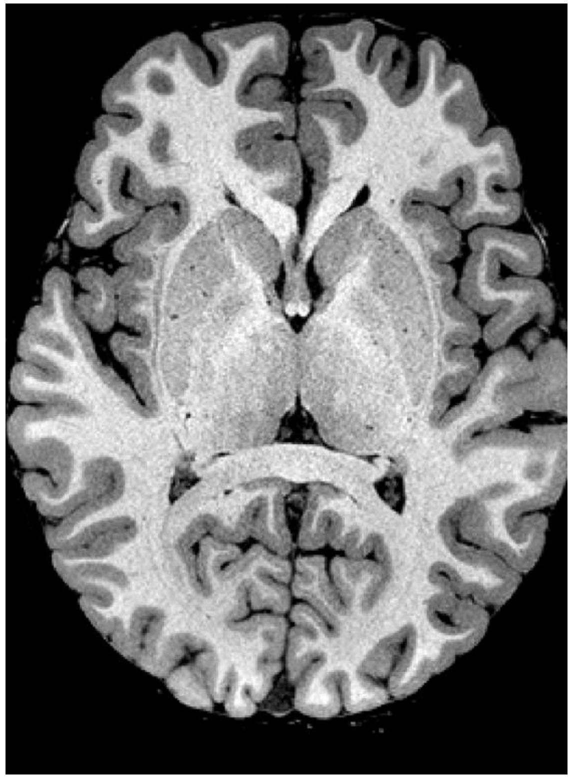
$Ef = Wf(H)H$ is the effective filter (should be 1)

- Conservative if SNR is low tends to become low-pass blurring instead of sharpening

- SNR = $\Phi_{ii}(u,v)/\Phi_{nn}(u,v)$ depends on I(u,v) strictly speaking is unknown

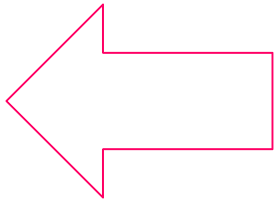
- H(u,v) must be known very precisely

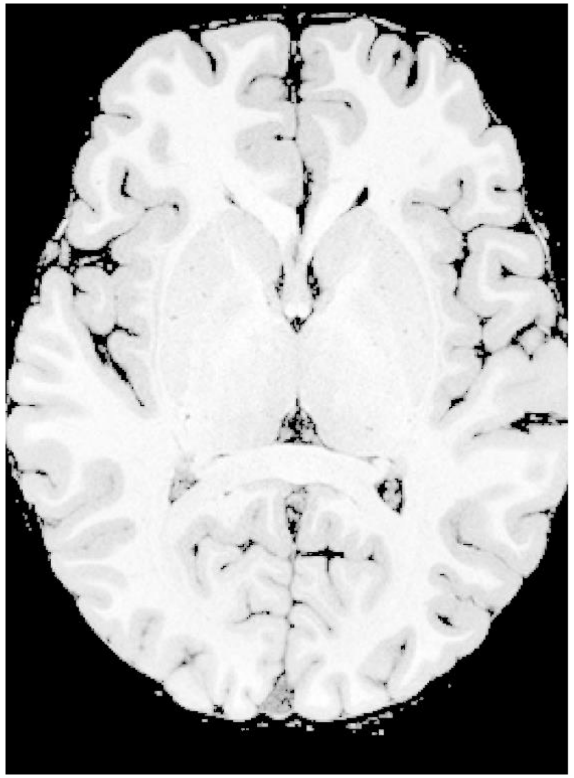
Computer Vision



Original Image

Contrast Enhancement





Observation with Bad Contrast

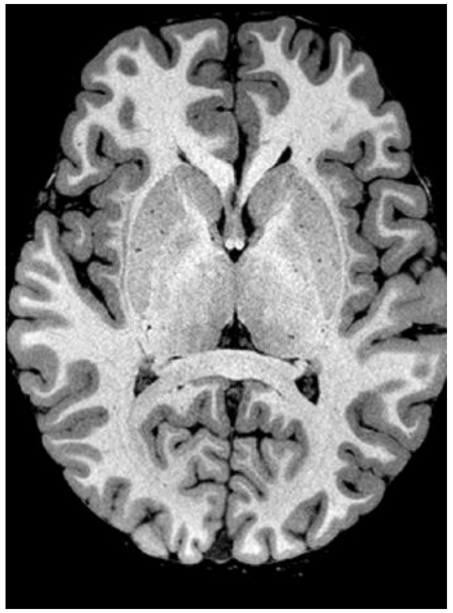
Contrast Enhancement

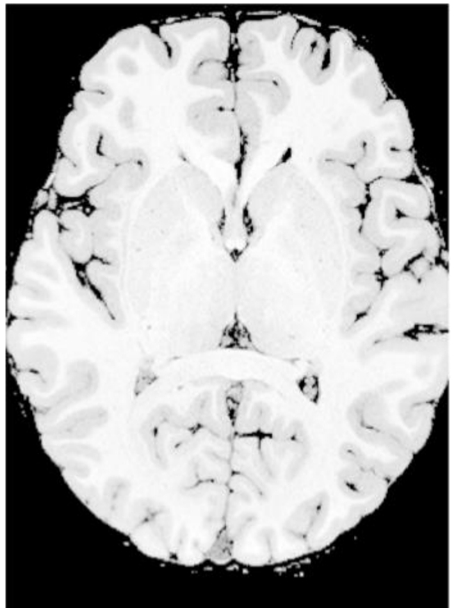
- Two use cases:

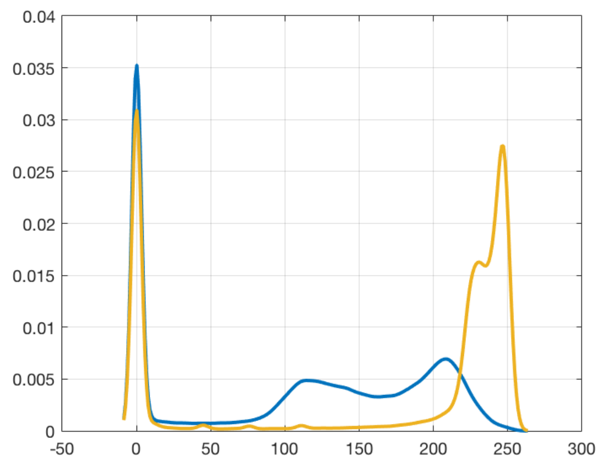
1. Compensating under-, over-exposure
2. Spending intensity range on interesting part of the image

- We will study histogram equalization

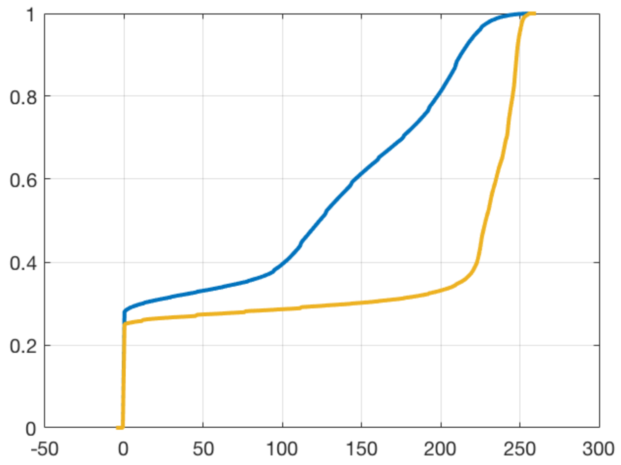
Intensity distributions - histogram







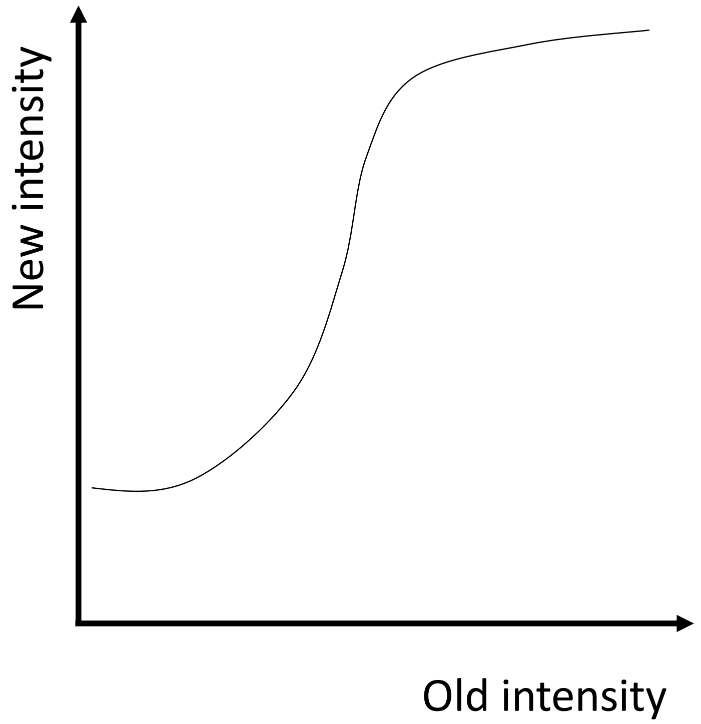
Histogram



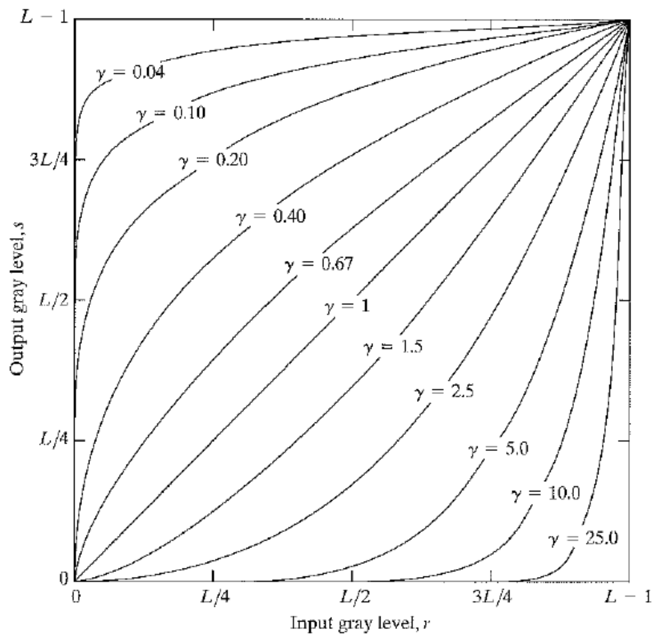
Cumulative histogram

Intensity mappings

Usually monotonic mappings required



Generic transformation function

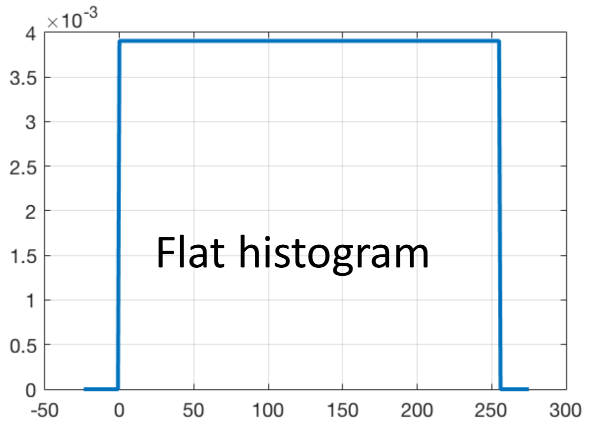


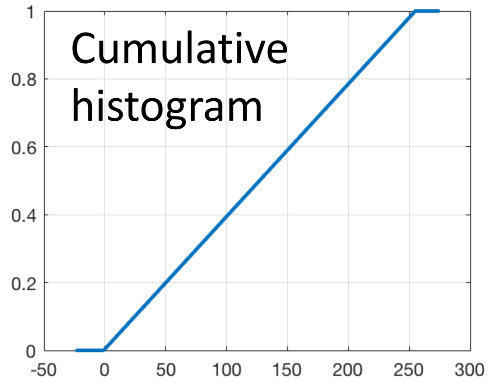
Power law transformation

$I_{new} = I_{old}^{\gamma}$

Histogram equalization

WHAT : create a flat histogram

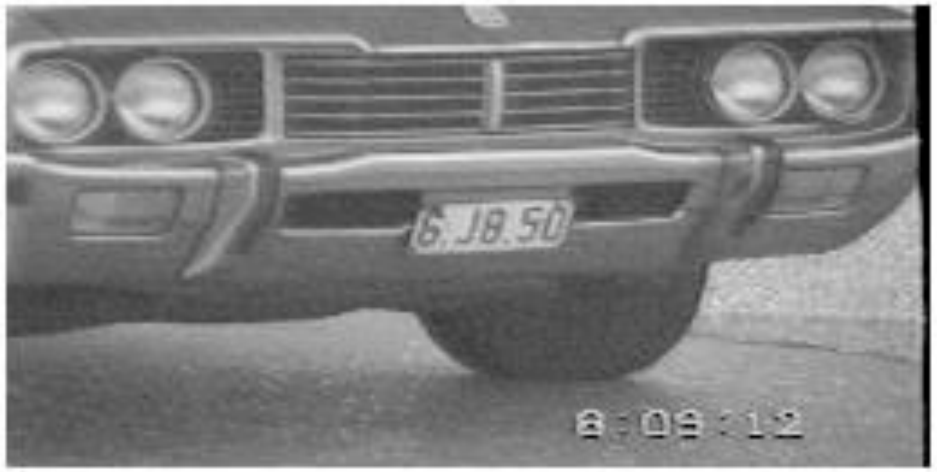


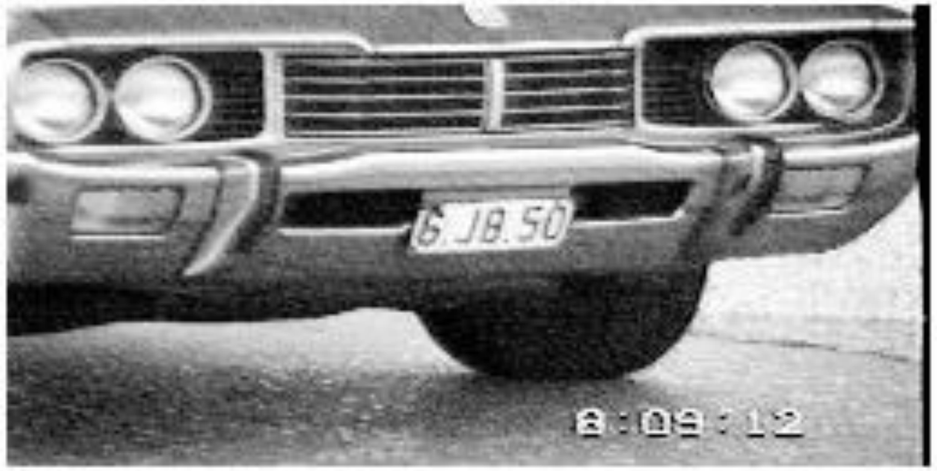


HOW : apply an appropriate intensity map depending on the image content

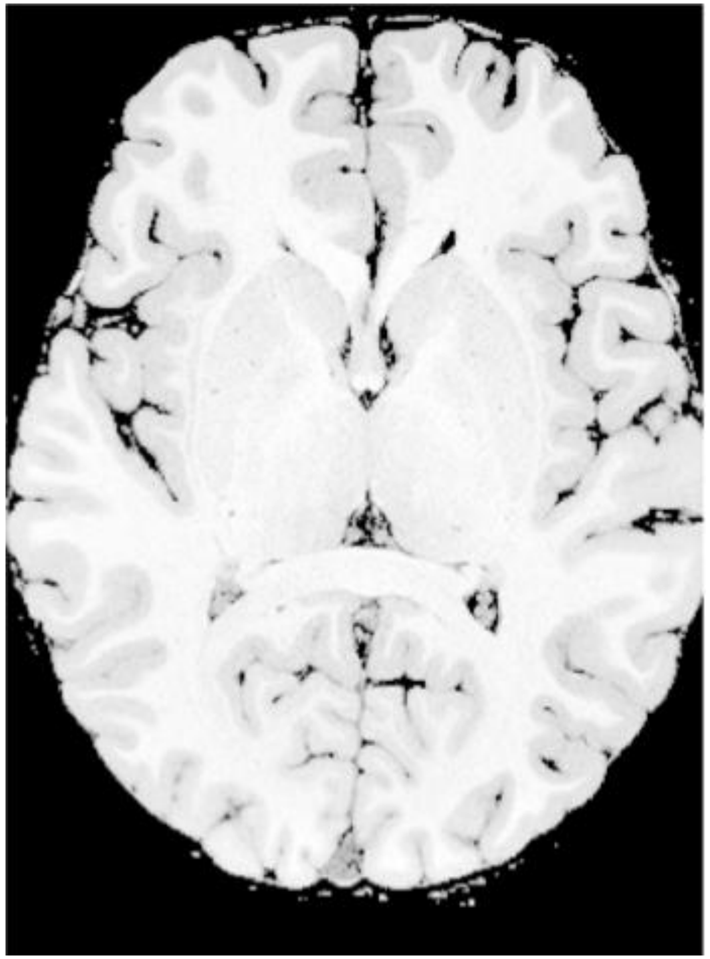
method will be generally applicable

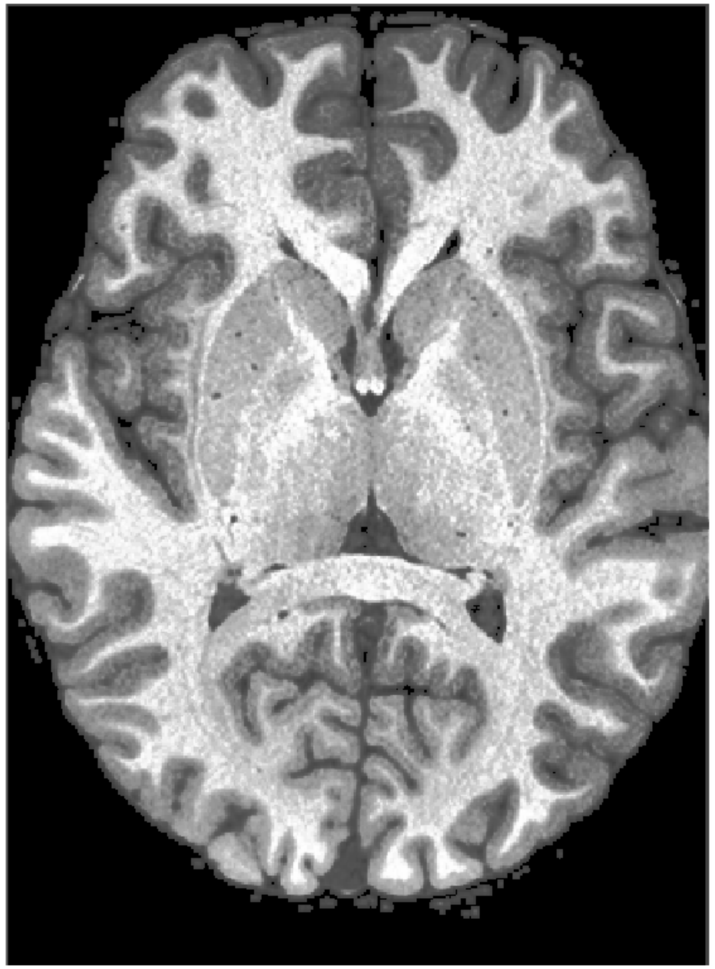
Histogram equalization example





Histogram equalization example

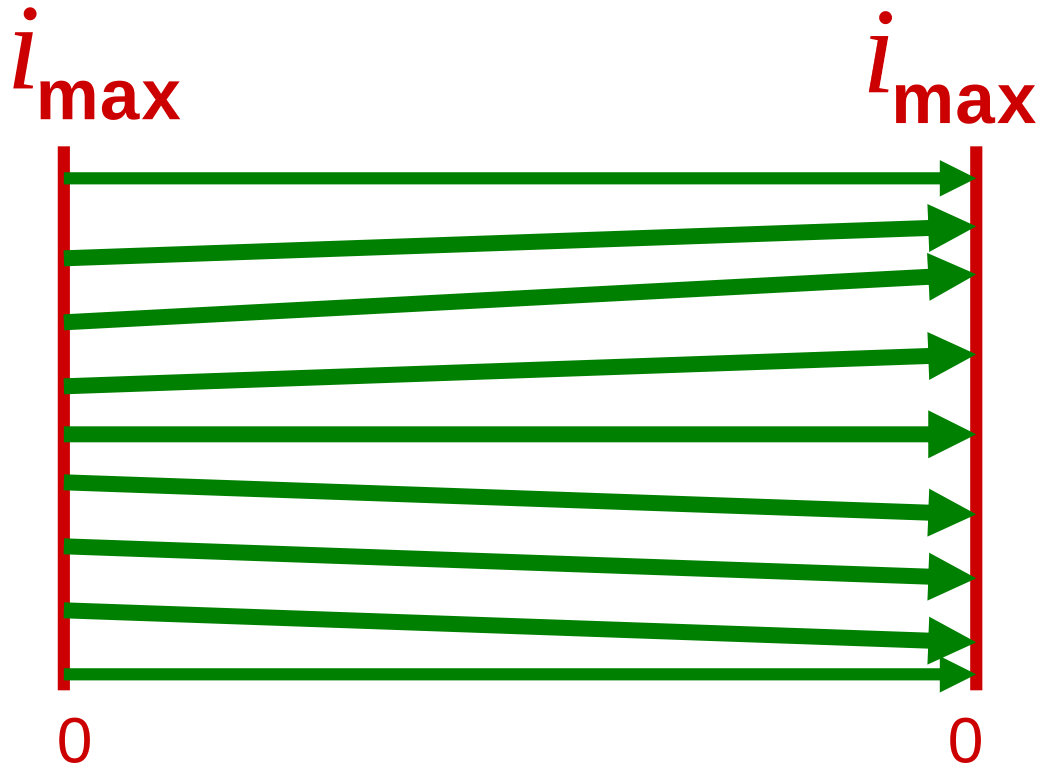




Histogram equalization - principle

Redistribute the intensities, 1-to-several (1-to-1 in the continuous case) and keeping their relative order, as to use them more evenly

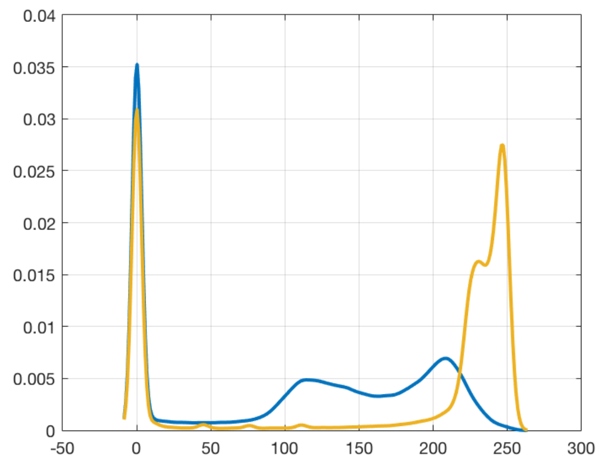
Ideally, obtain a constant, flat histogram



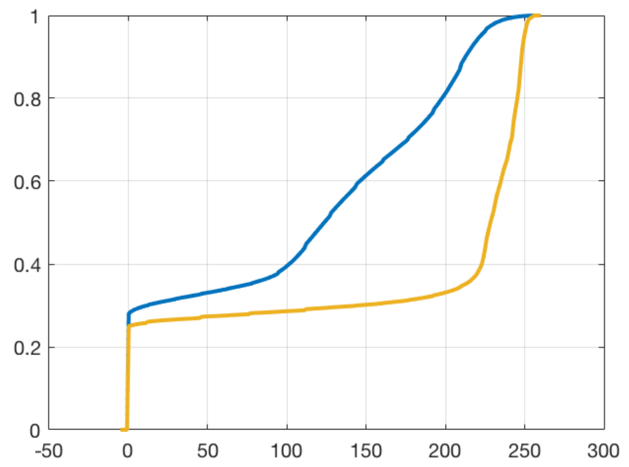
Histogram equalization - algorithm

This mapping is easy to find:

It corresponds to the cumulative intensity probability or cumulative histogram

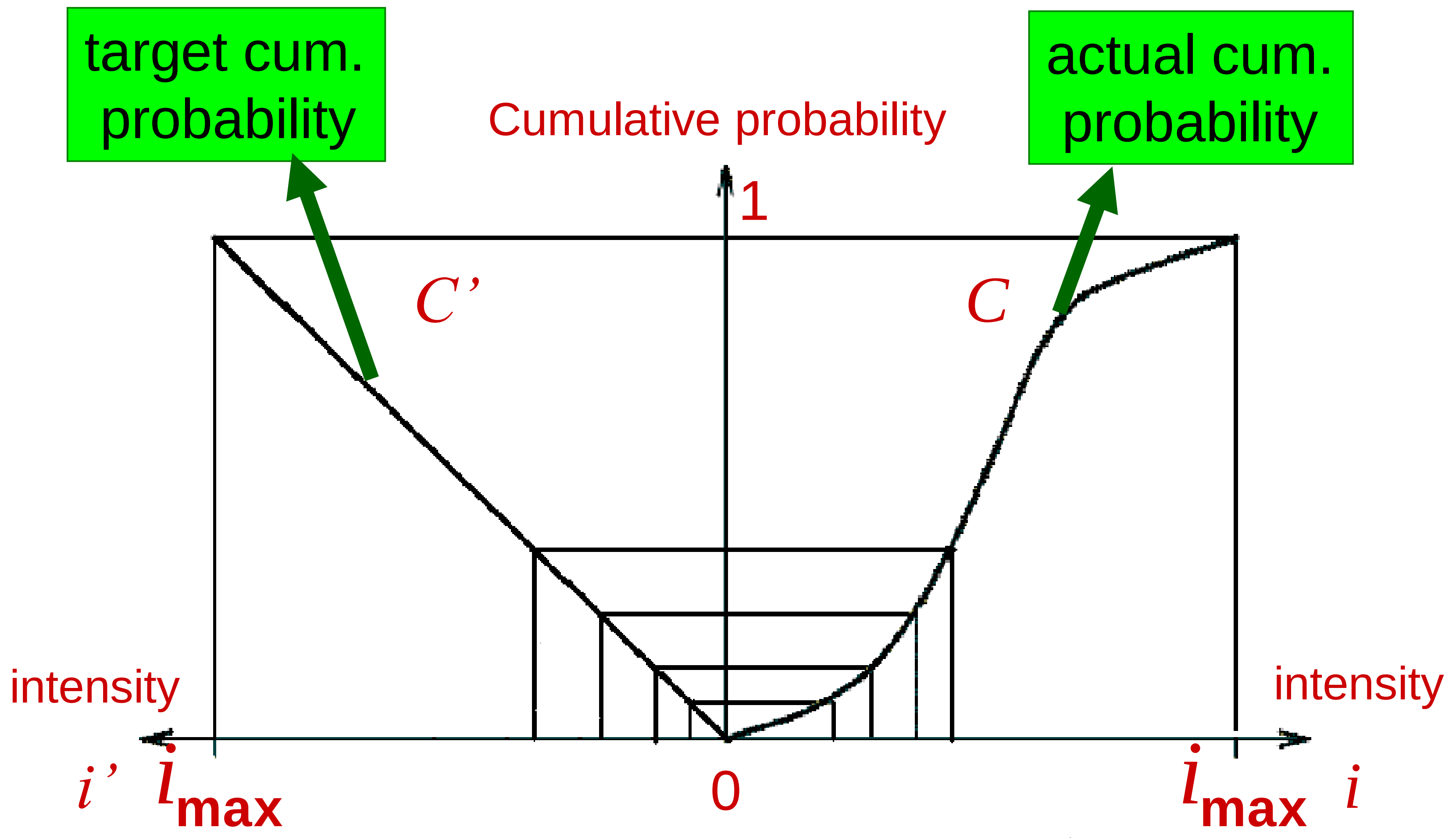


Histogram



Cumulative histogram

Algorithm sketch



$$i' = T(i) = i_{max} C(i) = i_{max} \int_0^i p(j)dj$$

Mathematical justification in continuous case

suppose continuous probability density of original intensities i: $p(i)$

Our mapping $$i' = T(i) = i_{max} \int_0^i p(j)dj$$

Probability density of the transformed intensities are given as

$$p(i') = p(i)\frac{di}{di'} = p(i)\frac{1}{p(i)}\frac{1}{i_{max}} = \frac{1}{i_{max}}$$

Indeed a flat distribution!