# Recap from the last two weeks

- Two weeks ago:
  - How to capture light with the camera
  - Projection matrices from real world to camera
  - Discretization of images
  - Spatial and frequency domain
  - Sampling and quantization
  - LSI systems and Convolution

# Recap from the last two weeks

- Last week:
  - Feature extraction
  - Local features
  - Invariance to geometric and photometric changes
  - Points of interest
  - Local regions
  - Descriptive features
  - Combining points of interest, regions and descriptive features: SIFT, SURF,

# This week

- Image enhancement
  - Removing noise, improving sharpness, highlighting aspects
  - Simplifying interpretation
  - More pleasing look
  - Normalization for further processing
- Basic feature detection
  - Identifying the points of interest in an image
  - Edges
  - Corners

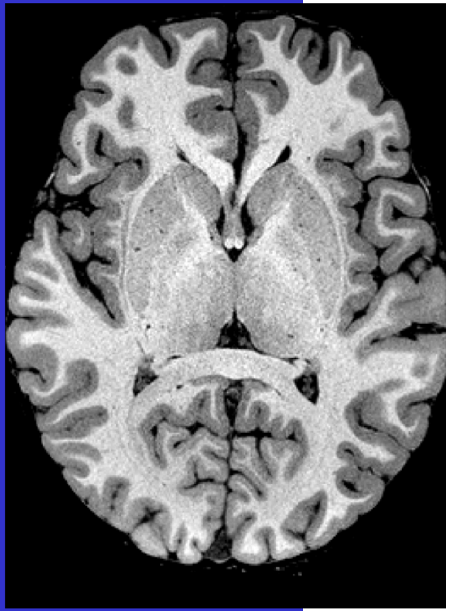
# Image Enhancement

# Learning objectives: what can you do after today?

- Reduce noise in images with linear and non-linear filters
- Choose appropriate filters for different noise patterns
- Describe anisotropic diffusion
- Sharpen / Deblur images
- Describe Wiener filter
- Improve image contrast

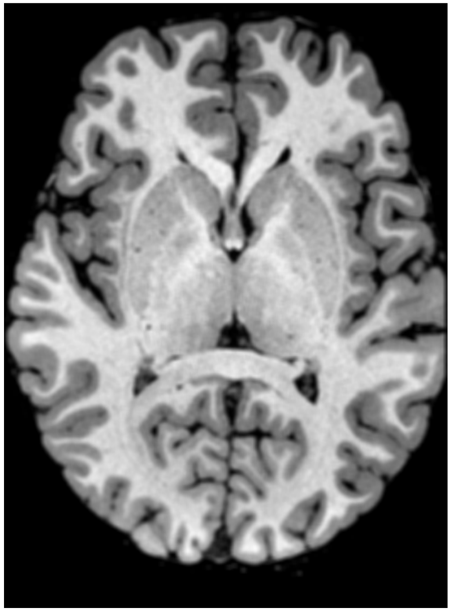
# Three types of image enhancement

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

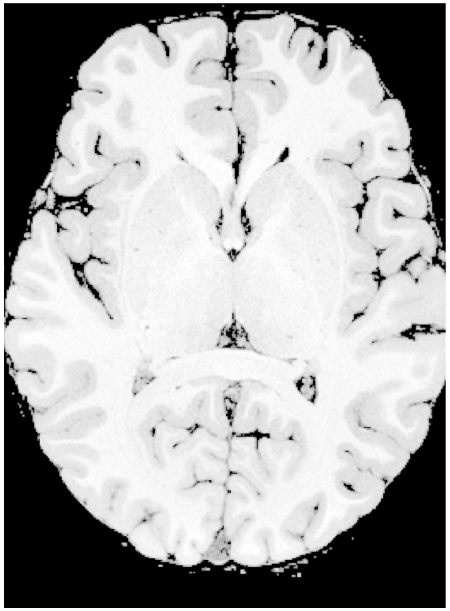


**Original Image**

**Noise**



**Blur**



**Bad Contrast**

# Overview

1. Preliminaries
    a. Reminders from previous lecture
    b. Fourier power spectra of images
2. Noise suppression
    a. Convolutional (Linear) filters
    b. Non-linear filters
3. Image de-blurring
    a. Unsharp masking
    b. Inverse Filtering
    c. Wiener Filters
4. Contrast enhancement
    a. Histogram Equalization

# Overview

1. **Preliminaries**
   a. **Reminders from previous lecture**
   b. **Fourier power spectra of images**
2. Noise suppression
   a. Convolutional (Linear) filters
   b. Non-linear filters
3. Image de-blurring
   a. Unsharp masking
   b. Inverse Filtering
   c. Wiener Filters
4. Contrast enhancement
   a. Histogram Equalization

Computer Vision

# Reminders from previous lecture: Fourier Transform

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

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

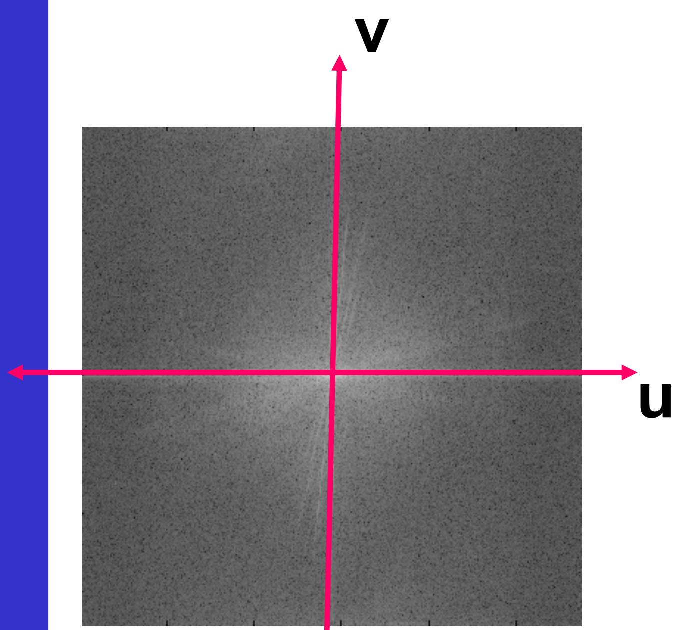
**→ The Fourier transform**

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

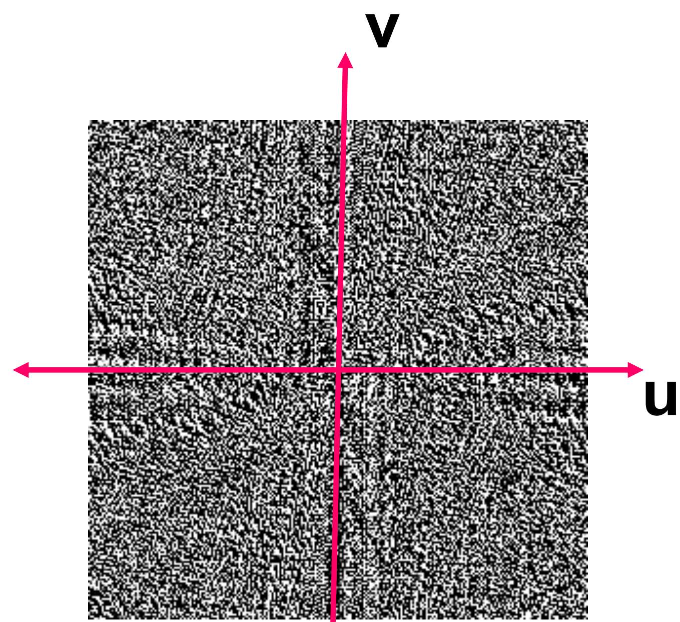
$$\mathcal{F}^{-1}[F(u,v)] = f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{i2\pi(ux+vy)}dxdy$$

**→ The inverse Fourier transform**

*f(x,y)*




*|F(u,v)|*




*phase F(u,v)*

# Reminders from previous lecture: Convolution Theorem

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

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

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

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

# Reminders from previous lecture: Modulation Transfer Function

**For any Linear Shift Invariant Operator**
**For any Convolutional Operator**

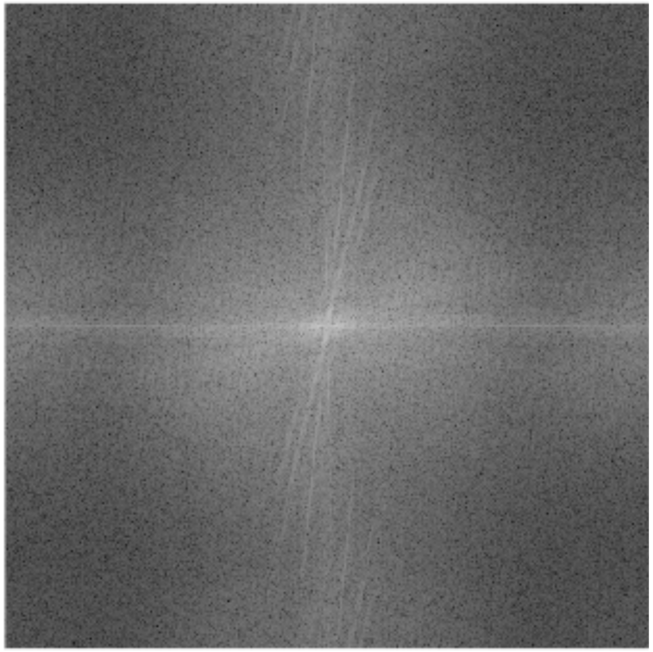
$$o(x, y) = i(x, y) * r(x, y)$$

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

$=$ modulation transfer function (MTF)

# Fourier power spectra of images

$i(x,y)$



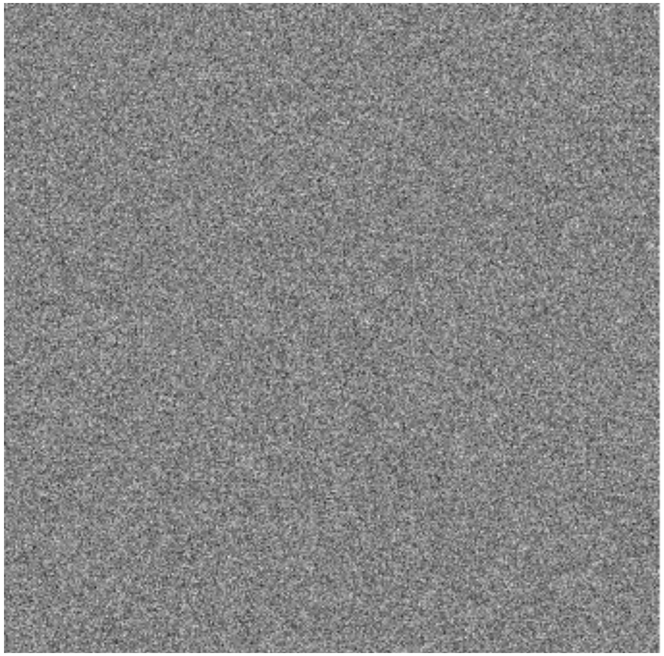
$\phi_{ii} = |I(u,v)|^2$

Amount of signal at each frequency pair

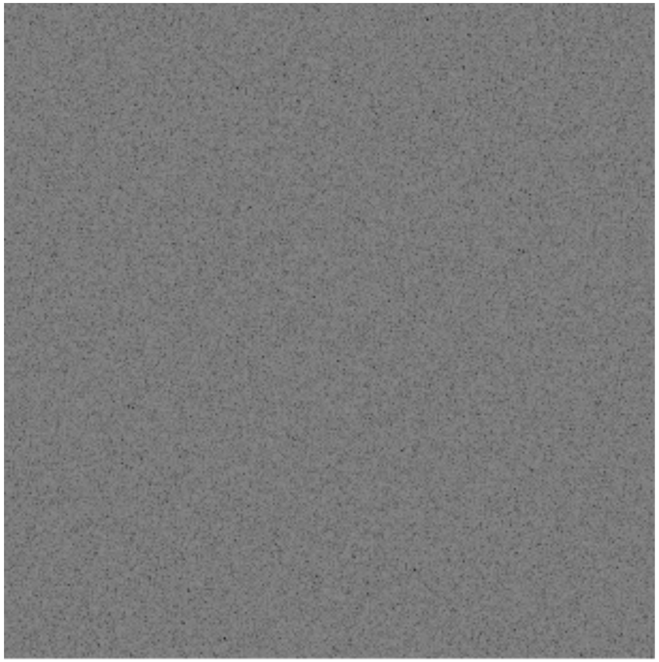
Images are mostly composed of homogeneous areas

- 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



$n(x,y)$

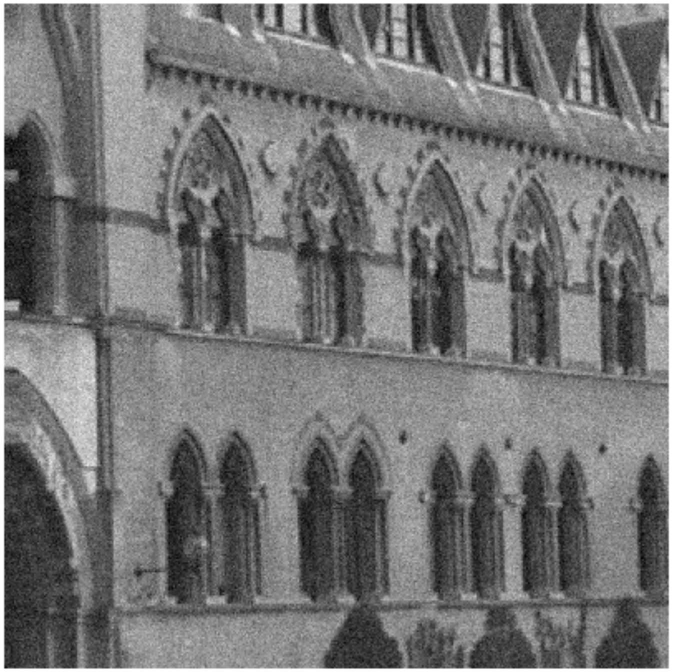


$\phi_{nn} = |N(u,v)|^2$

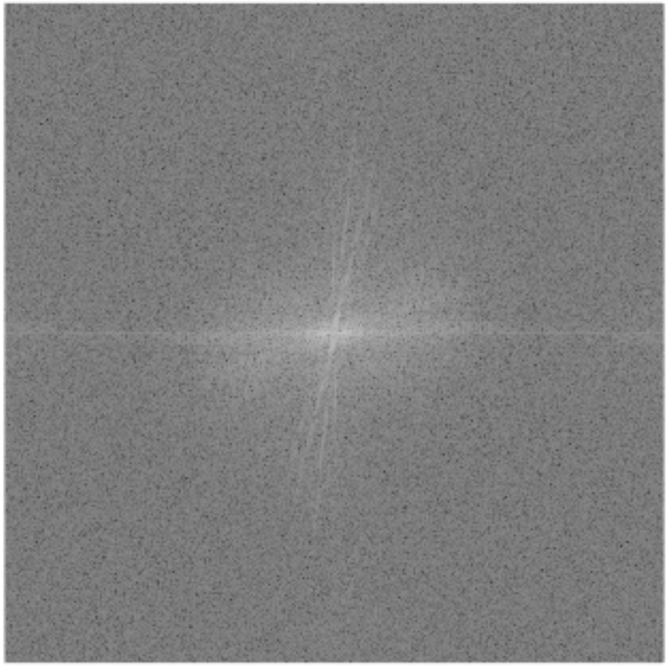
-Pure noise has a uniform power spectra
-Similar components in high and low frequencies.

Computer Vision

# Fourier power spectra of noisy image



$f(x,y)$

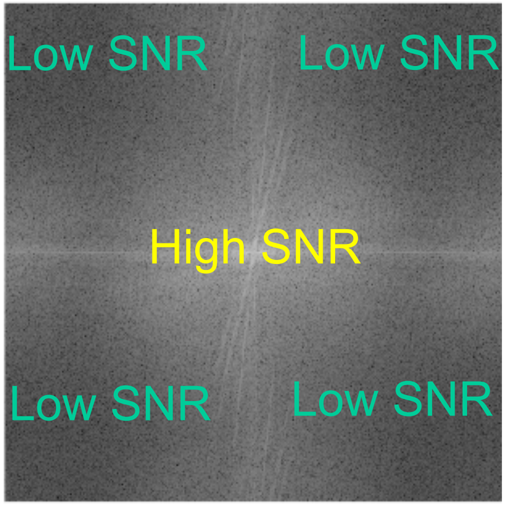


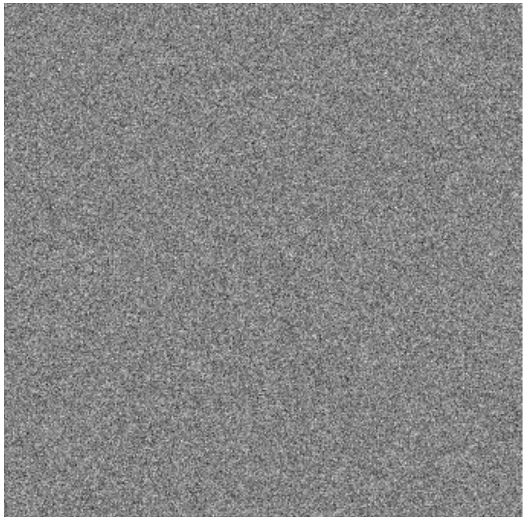
$\phi_{ff} = |F(u,v)|^2$

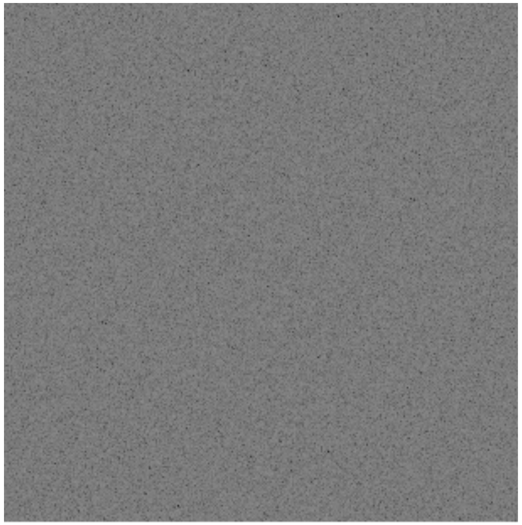
Power spectra is a combination of image and noise

Computer Vision

# Signal to Noise Ratio



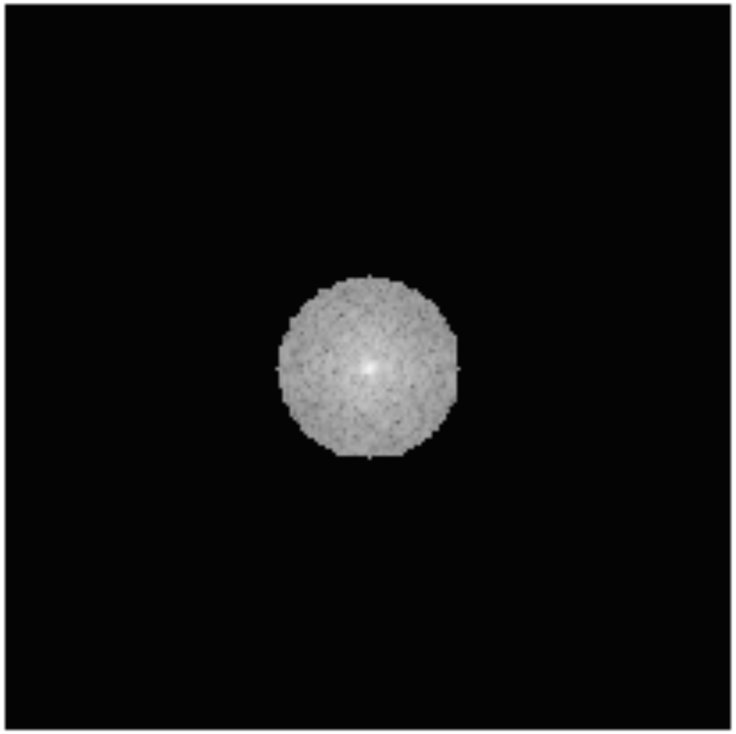





$$\phi_{ii}(u,v) \,/\, \phi_{nn}(u,v)$$

# Only retaining the low frequencies

Low signal/noise ratio at high frequencies ⇒ eliminate these



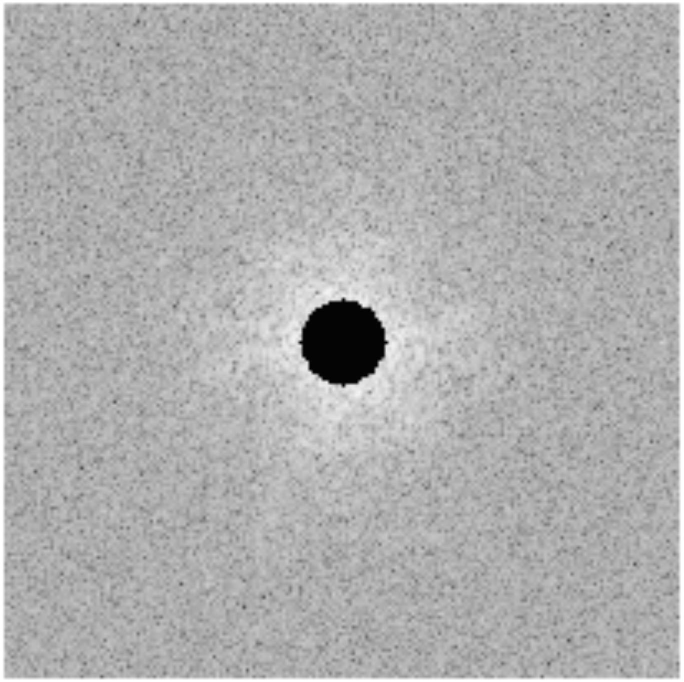
Smoother image but we lost details!

Computer Vision

# High frequencies contain noise but also Edges!

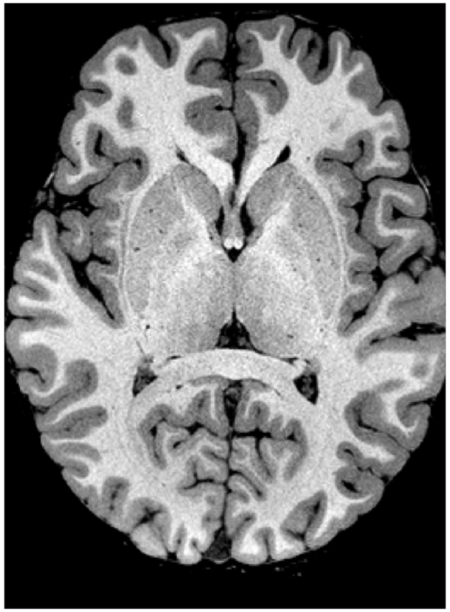
We cannot simply discard the higher frequencies

They are also introduces by edges ; example :



# Overview

1. Preliminaries
   a. Reminders from previous lecture
   b. Fourier power spectra of images
2. **Noise suppression**
   a. **Convolutional (Linear) filters**
   b. **Non-linear filters**
3. Image de-blurring
   a. Unsharp masking
   b. Inverse Filtering
   c. Wiener Filters
4. Contrast enhancement
   a. Histogram Equalization



Original Image

**Noise Suppression**

Noisy Observation

# Noise suppression

specific methods for specific types of noise

we only consider 2 general options :

- 1. Convolutional linear filters
  low-pass convolution filters

- 2. Non-linear filters
  edge-preserving filters
  - a. median
  - b. anisotropic diffusion

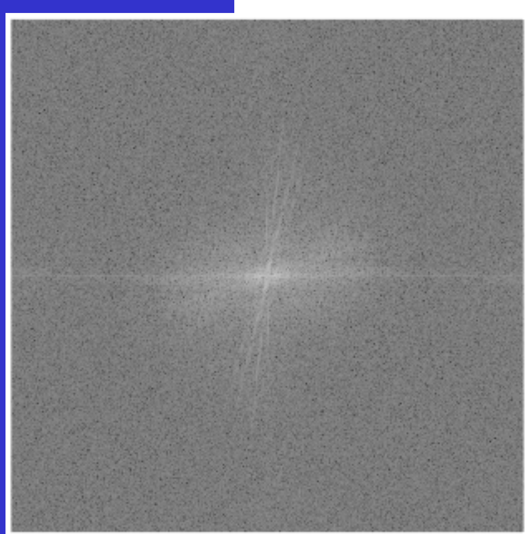
Computer Vision

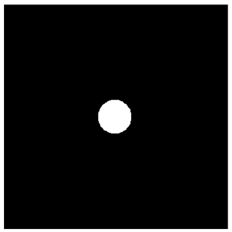
# Low-pass filters: principle

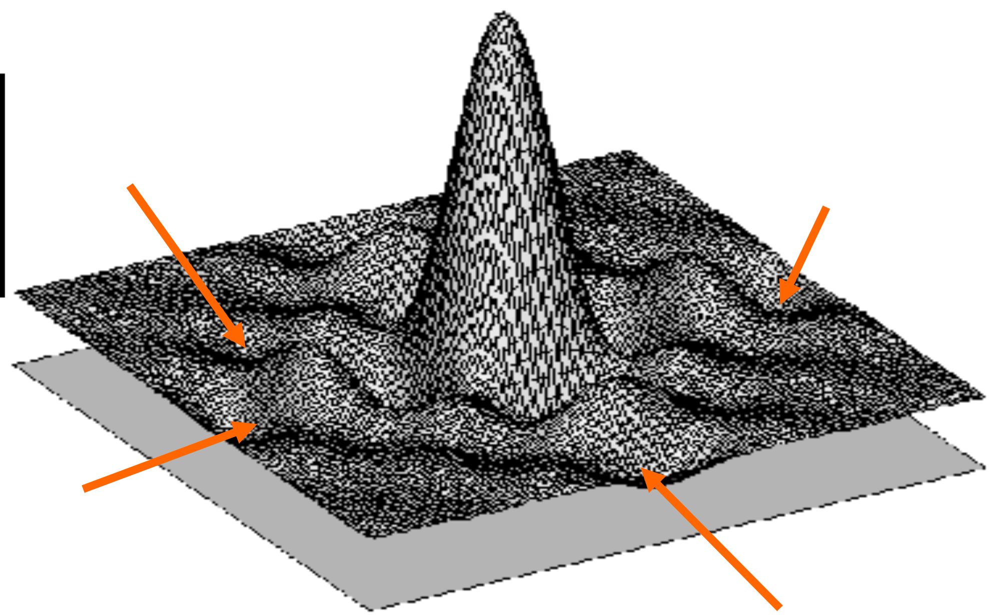
Goal: remove low-signal/noise part of the spectrum

Approach 1: Multiply the Fourier domain by a mask

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

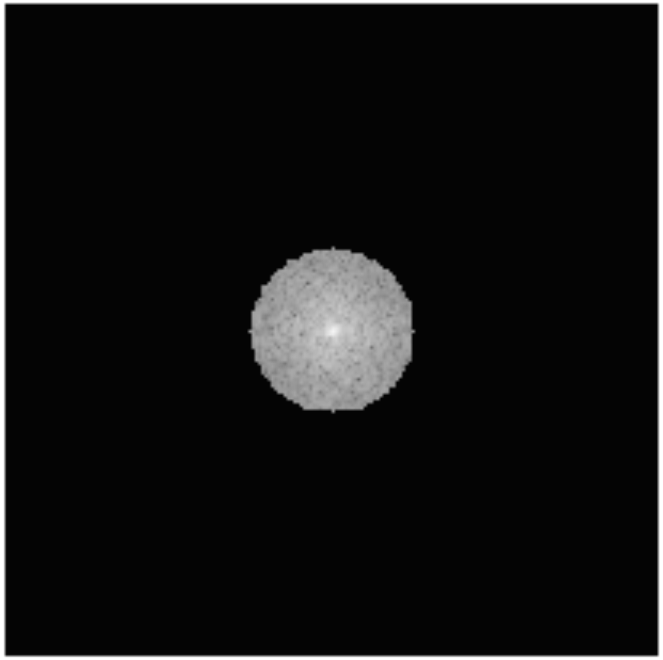






Computer Vision

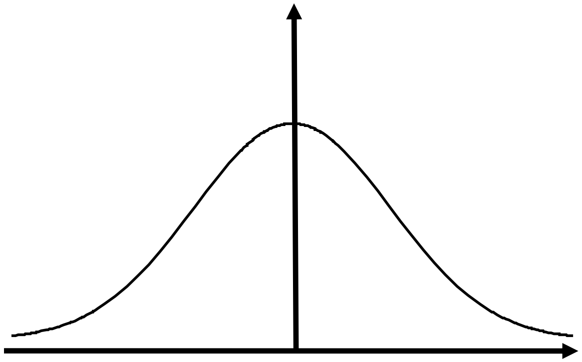
# Illustration of rippling



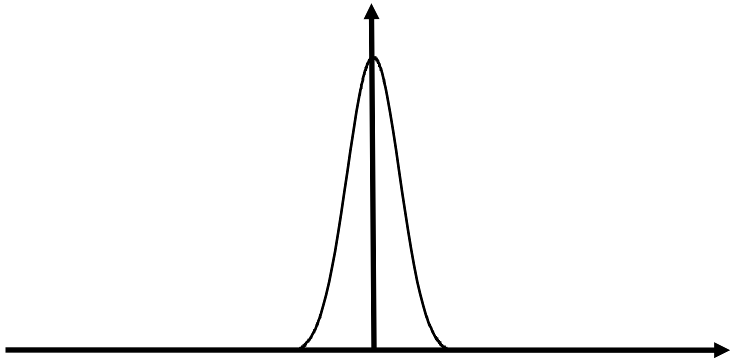
# Approach 2: Low-pass convolution filters

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

Computer Vision

# Averaging

One of the most straight forward convolution filters: averaging filters

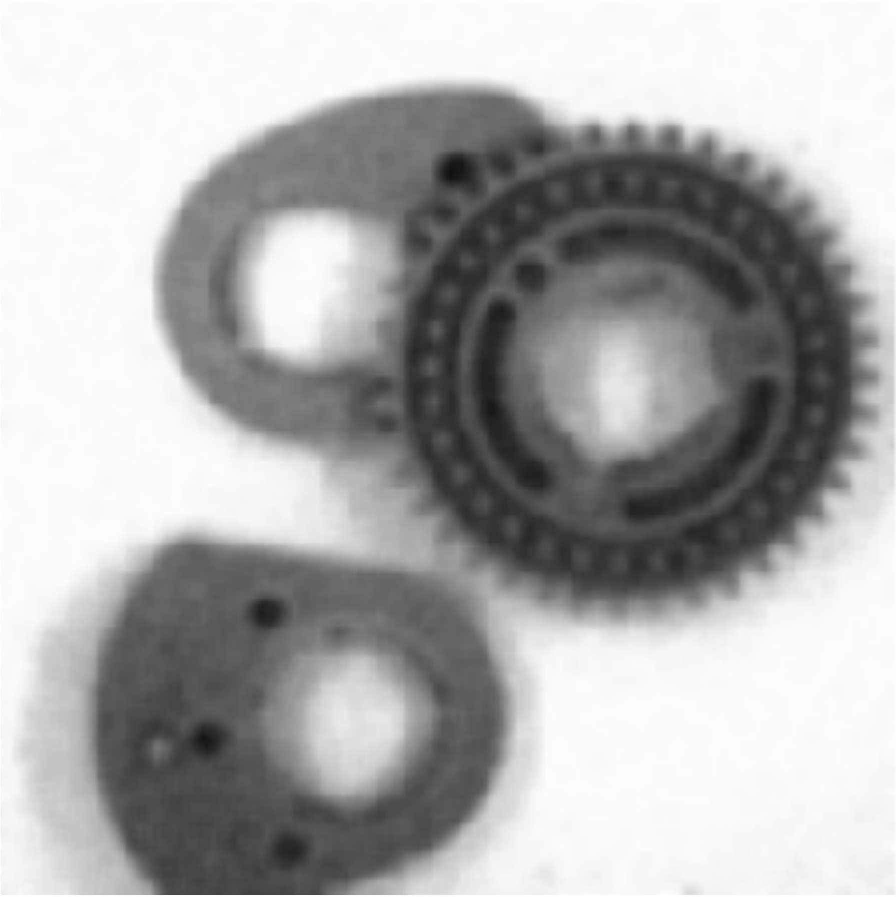
$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \qquad \frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

Separable: $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = \frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * \frac{1}{3}\begin{bmatrix}1&1&1\end{bmatrix}$

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

Computer Vision

# Example for box averaging



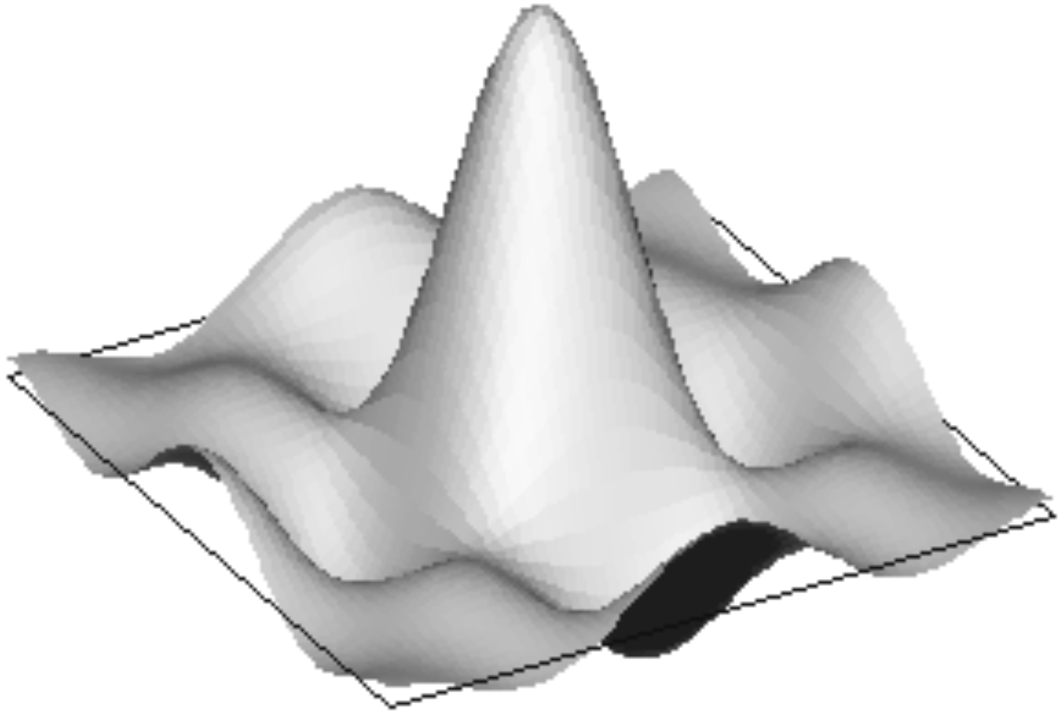
Noise is gone.
Result is blurred!

Computer Vision

# MTFs for averaging

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.

no ripples in either domain required!

# Solution: Binomial filters

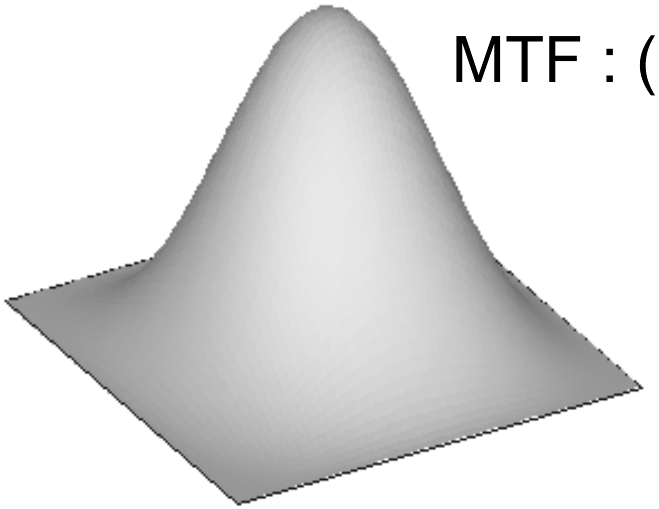
iterative convolutions of (1,1)

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

2D :

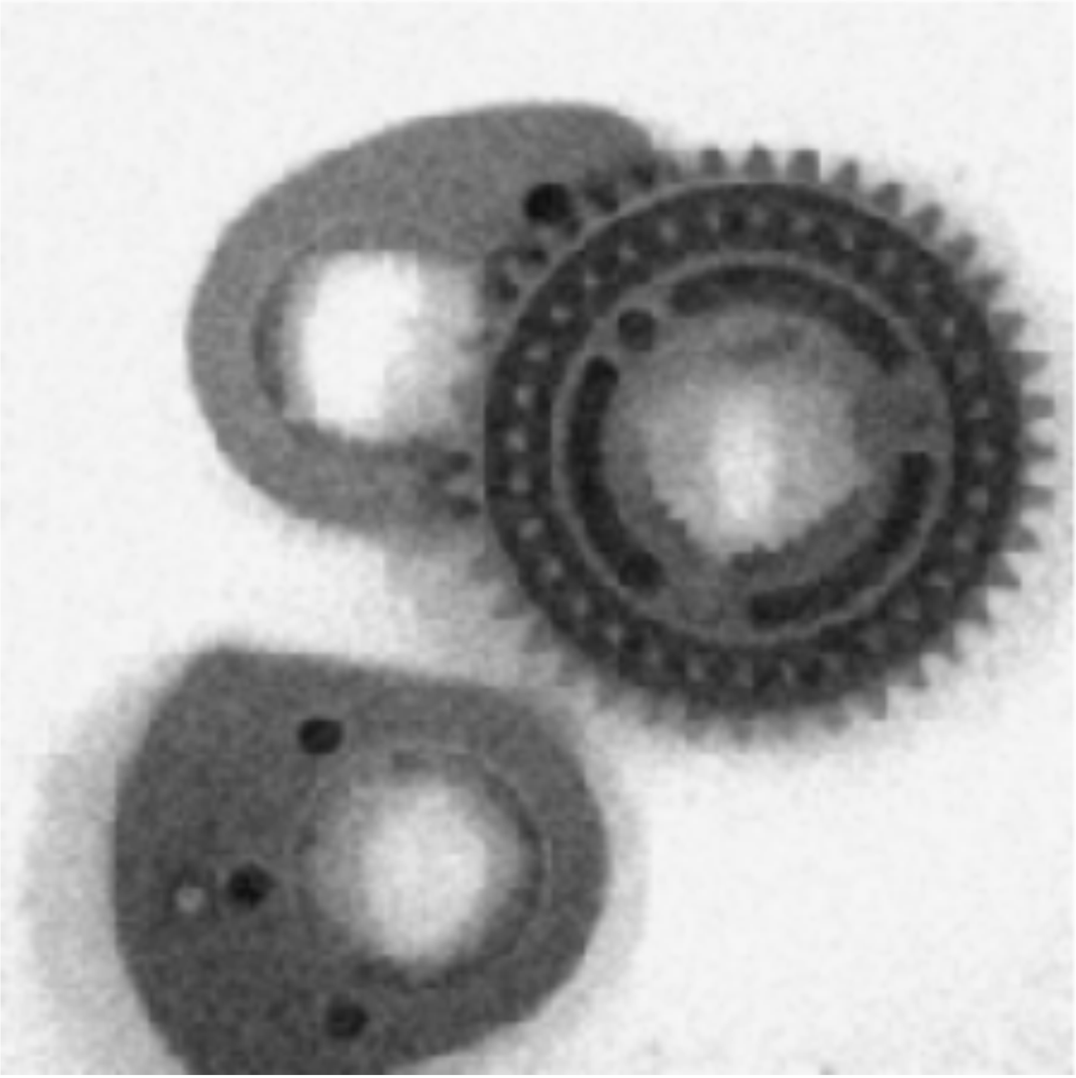
| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

Also separable



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

# Result of binomial filter



Computer Vision

# Limit of iterative binomial filtering

f :

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

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

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

**Gaussian**

# Gaussian smoothing

Gaussian is limit case of binomial filters

noise gone, no ripples, but still blurred…

Actually linear filters cannot solve this problem

# Some implementation issues

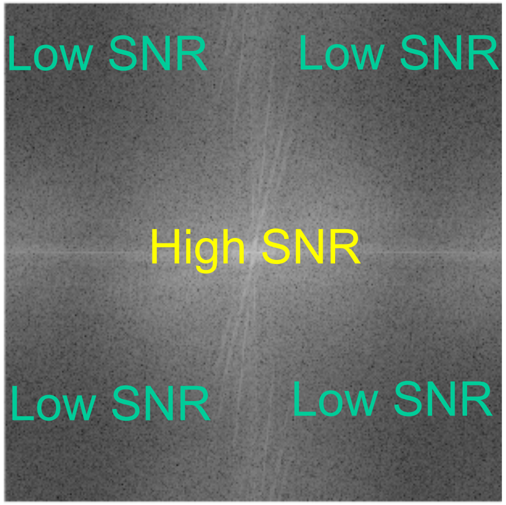
separable filters can be implemented efficiently

large filters through multiplication in the frequency domain

integer mask coefficients increase efficiency powers of 2 can be generated using shift operations

Computer Vision

# Question




Can a linear-shift-invariant systems do a perfect job?

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

Computer Vision

# Noise suppression

specific methods for specific types of noise

we only consider 2 general options :

- 1. Convolutional linear filters
  low-pass convolution filters

- 2. Non-linear filters
  edge-preserving filters
  fighting blurring!
  - a. median
  - b. anisotropic diffusion

# Median filters : principle

non-linear filter

method :

- 1. rank-order neighbourhood intensities
- 2. take middle value

no new grey levels emerge...

Computer Vision

# Median filters : odd-man-out

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

e.g.

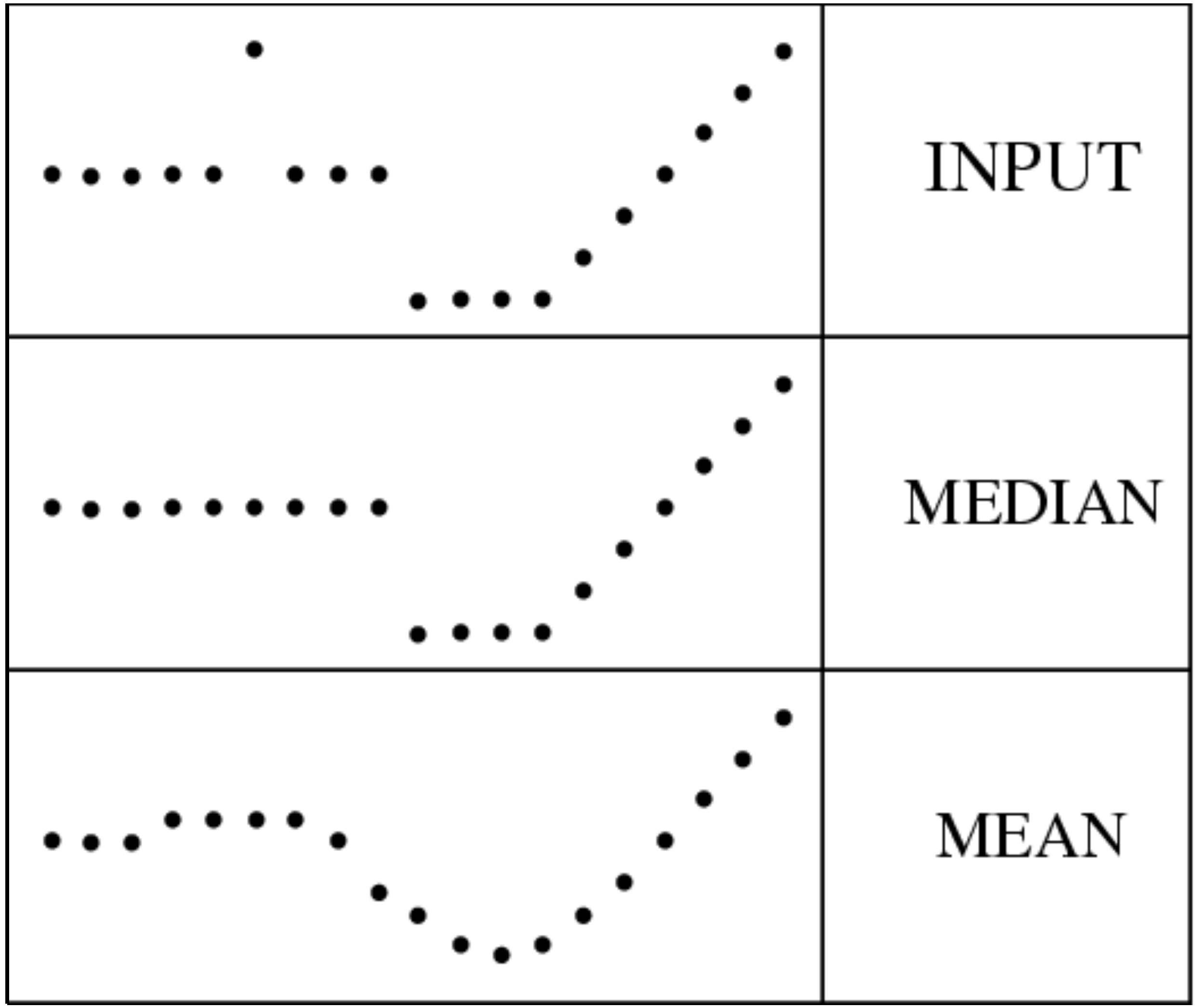
1,1,1,7,1,1,1,1

↓

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

# Median filters : example

filters have width 5 :



# Median filters : analysis

median completely discards the spike,
linear filter always responds to all aspects

median filter preserves discontinuities,
linear filter produces rounding-off effects

DON'T become all too optimistic

Computer Vision

# Median filter : results

3 x 3 median filter :

sharpens edges, destroys edge cusps and protrusions

Computer
Vision

# Median filters : results

Comparison with Gaussian :

e.g. upper lip smoother, eye better preserved

Computer Vision

# Example of median

10 times 3 X 3 median

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

Computer Vision

# Anisotropic diffusion : principle

non-linear filter

method :

- 1. Gaussian smoothing across homogeneous intensity areas
- 2. No smoothing across edges

# The 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!

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

Computer Vision

# Diffusion as Gaussian lowpass 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

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

Computer Vision

# Selection of diffusion coefficient

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

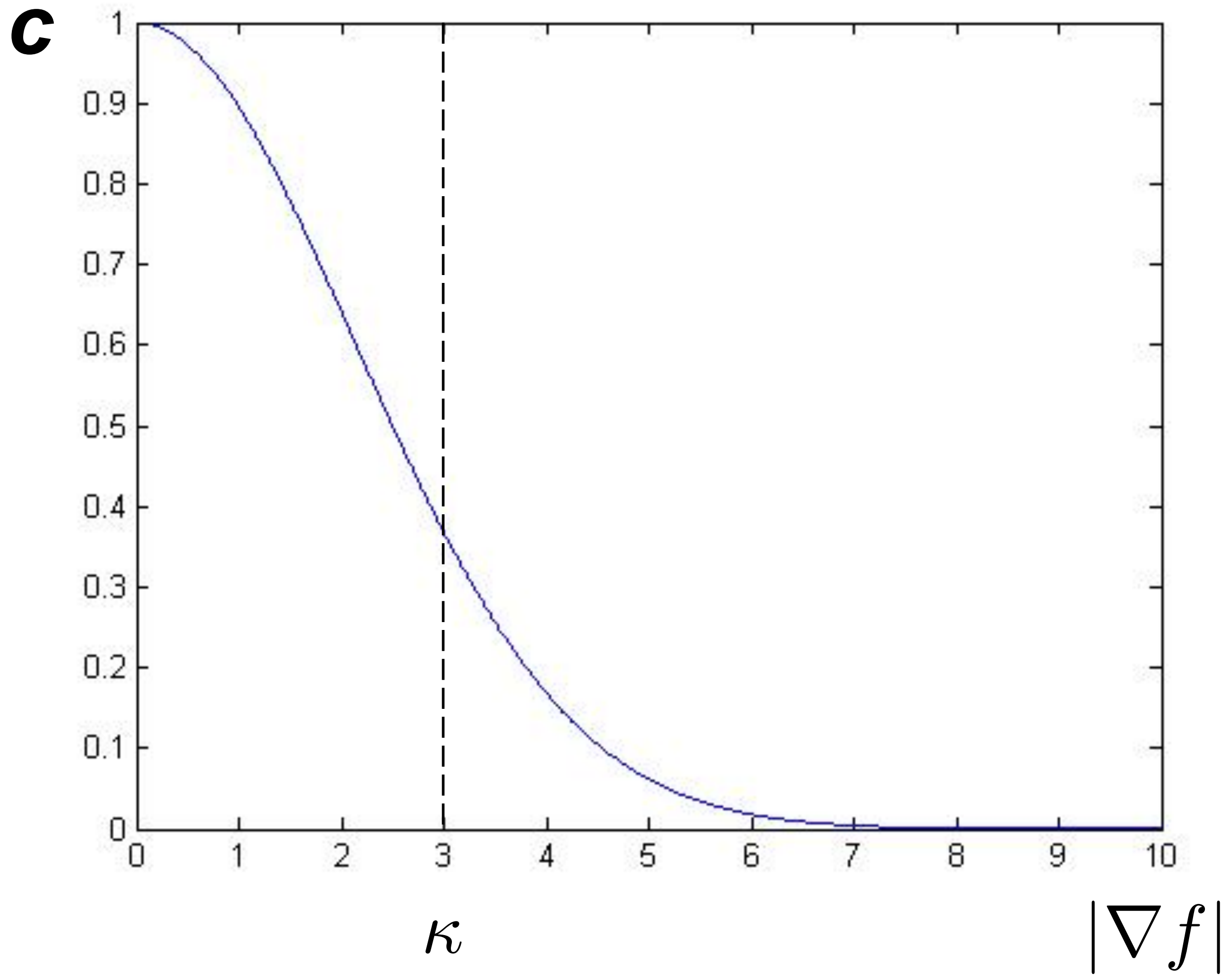
or

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

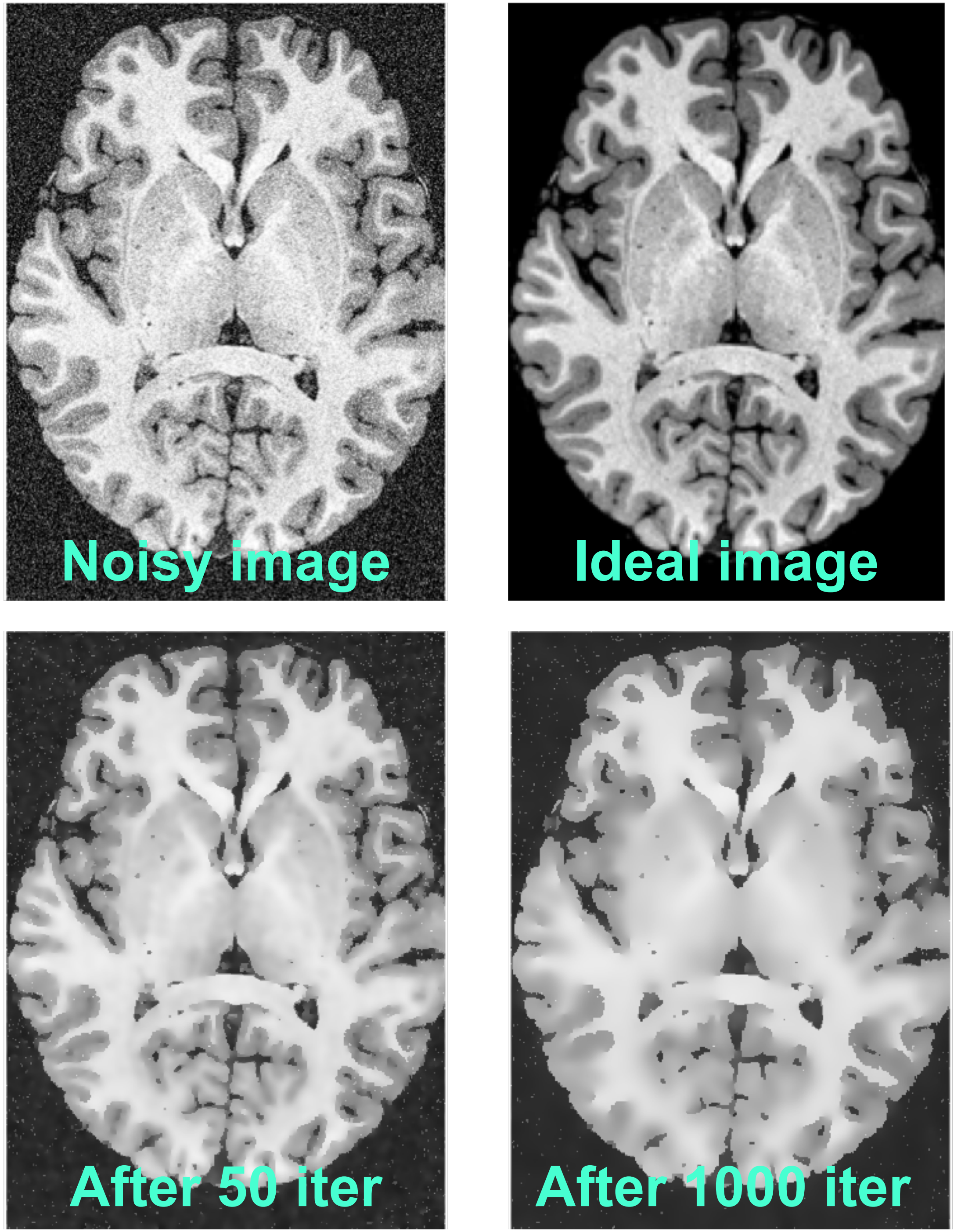
***κ*** controls the contrast to be preserved by smooting actually edge sharpening happens

Computer Vision

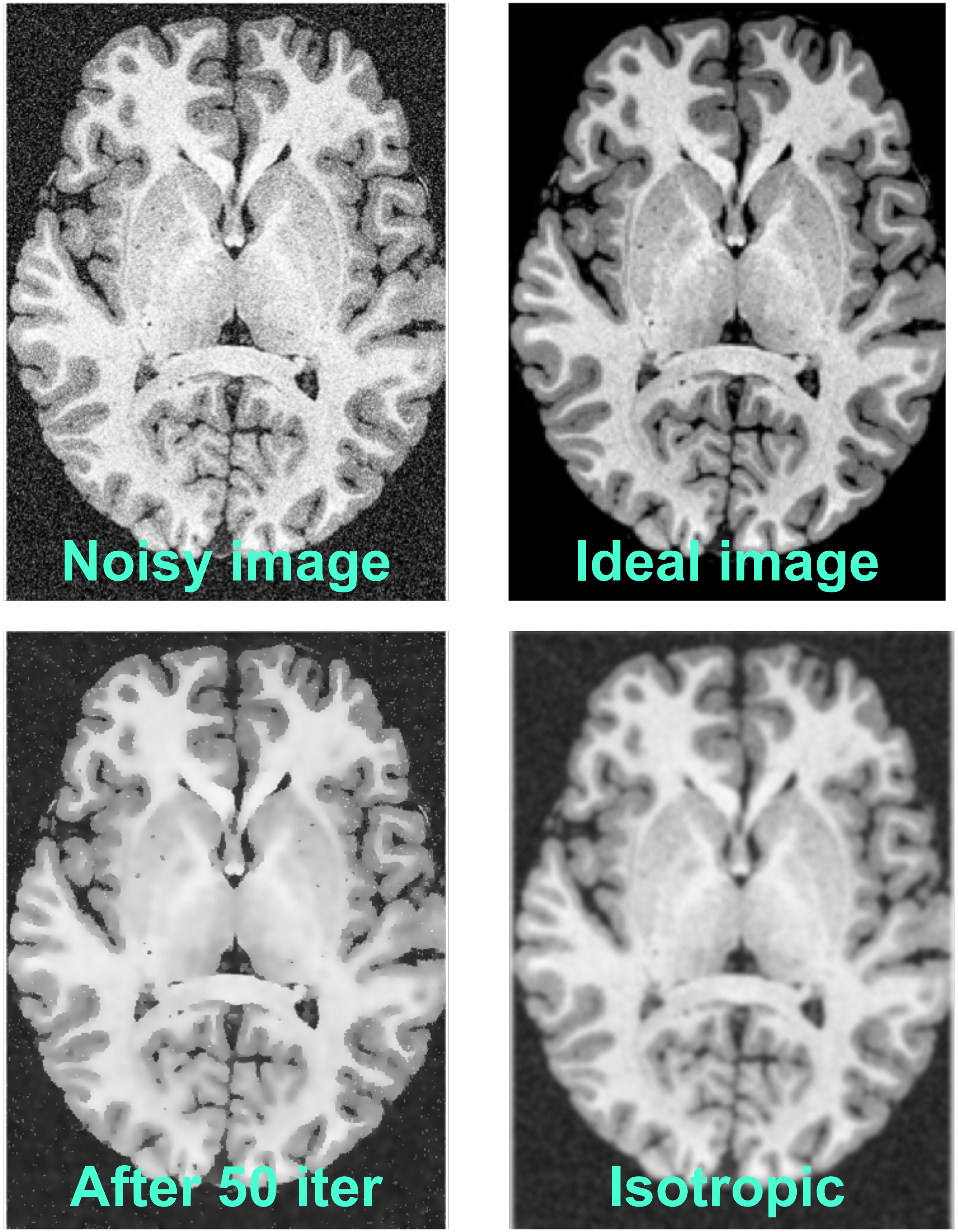
# Dependence on contrast



Computer Vision



Computer Vision



# Let's see what really happens in 1D

Take $c(p) = e^{-p^2}$ with $p(x,t) = \dfrac{\partial f}{\partial x}(x,t)$

$$\frac{\partial f}{\partial t} = \frac{\partial}{\partial x}\left(c(p)\frac{\partial f}{\partial x}\right) = c(p)\frac{\partial^2 f}{\partial x^2} + \left(\frac{\partial c}{\partial p}\right)\left(\frac{\partial p}{\partial x}\right)\frac{\partial f}{\partial x}$$

with $\dfrac{\partial p}{\partial x} = \dfrac{\partial^2 f}{\partial x^2}$ and $\dfrac{\partial f}{\partial x} = p$

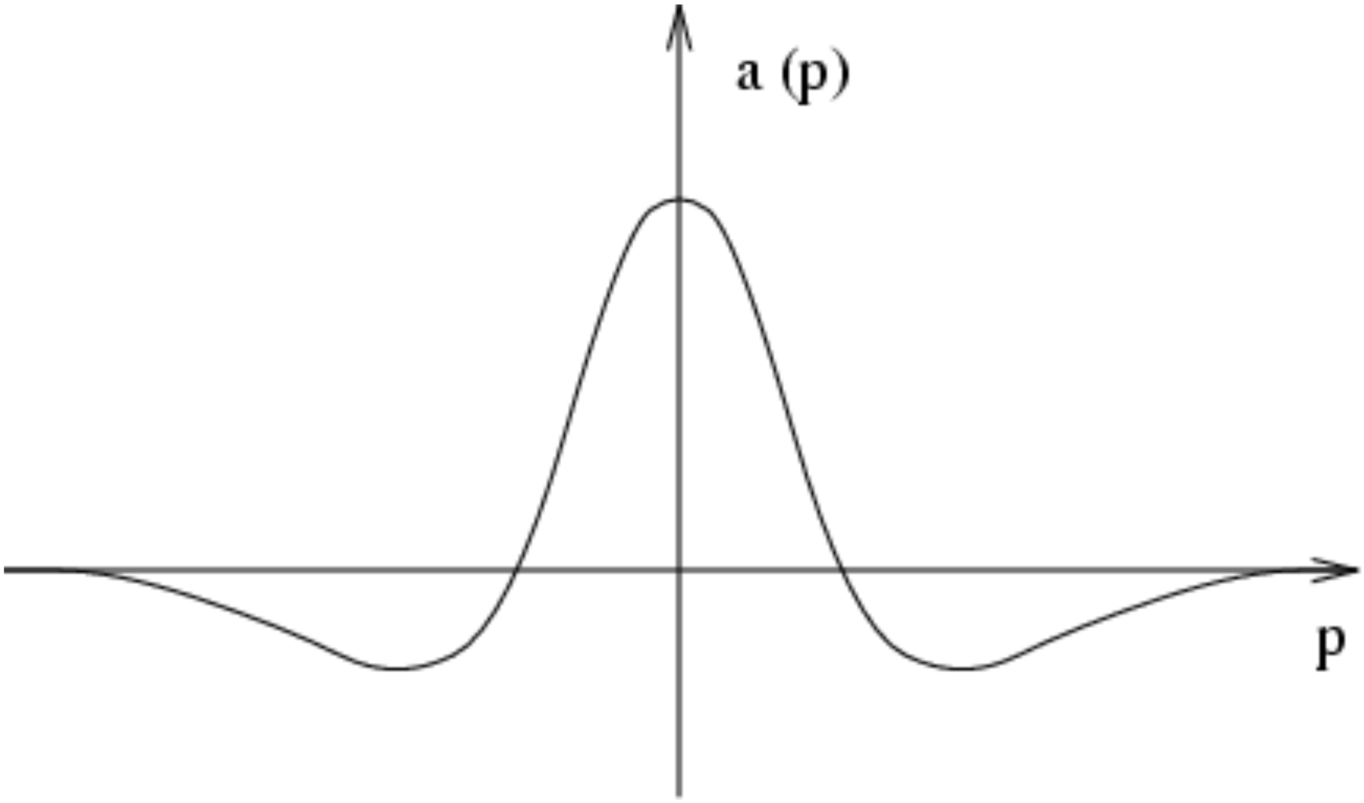
yields $\boxed{\dfrac{\partial f}{\partial t} = a(p)\dfrac{\partial^2 f}{\partial x^2}}$

with $a(p) = c(p) + p\dfrac{\mathrm{d}c}{\mathrm{d}p} = e^{-p^2}\left(1 - 2p^2\right)$

Computer Vision

# Anisotropic diffusion

result : diffusion with gradient dependent sign :



Computer Vision

# 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

# Finite difference approximation of the divergence operator

$$\nabla^2 \approx = \begin{array}{|c|c|c|}\hline & & \\ \hline 1 & -2 & 1 \\ \hline & & \\ \hline\end{array} + \begin{array}{|c|c|c|}\hline & 1 & \\ \hline & -2 & \\ \hline & 1 & \\ \hline\end{array}$$

$$= \begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -4 & 1 \\ \hline & 1 & \\ \hline\end{array}$$

$$= \begin{array}{|c|c|c|}\hline & & \\ \hline 1 & -1 & \\ \hline & & \\ \hline\end{array} \quad \begin{array}{|c|c|c|}\hline & 1 & \\ \hline & -1 & \\ \hline & & \\ \hline\end{array}$$

$$+$$

$$\begin{array}{|c|c|c|}\hline & & \\ \hline & -1 & 1 \\ \hline & & \\ \hline\end{array} \quad \begin{array}{|c|c|c|}\hline & & \\ \hline & -1 & \\ \hline & 1 & \\ \hline\end{array}$$

# Divergence in the presence of c

$$div(c\nabla) = c_1 \begin{bmatrix} & & \\ 1 & -1 & \\ & & \end{bmatrix} + c_2 \begin{bmatrix} & 1 & \\ & -1 & \\ & & \end{bmatrix} + c_3 \begin{bmatrix} & & \\ & -1 & 1 \\ & & \end{bmatrix} + c_4 \begin{bmatrix} & & \\ & -1 & \\ & 1 & \end{bmatrix}$$

Coefficients depend on derivatives of c

# Anisotropic diffusion: Output

End state is
homogeneous

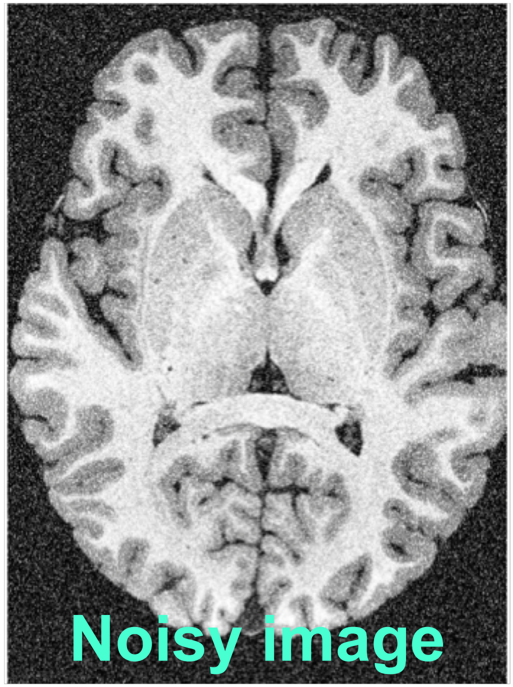
Computer Vision

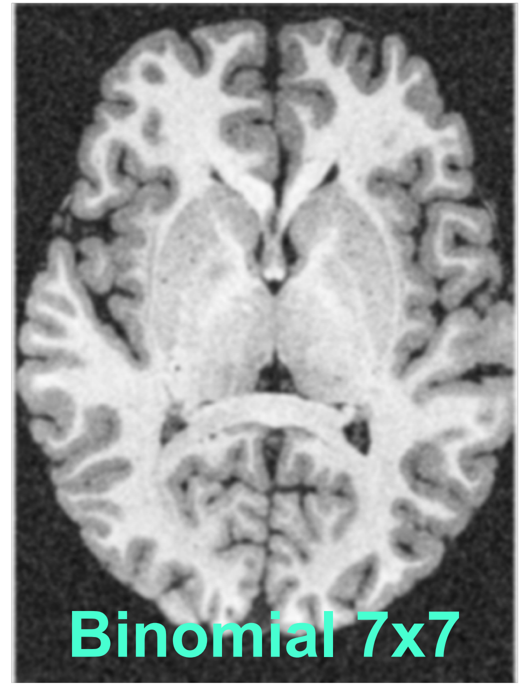
# Restraining the diffusion

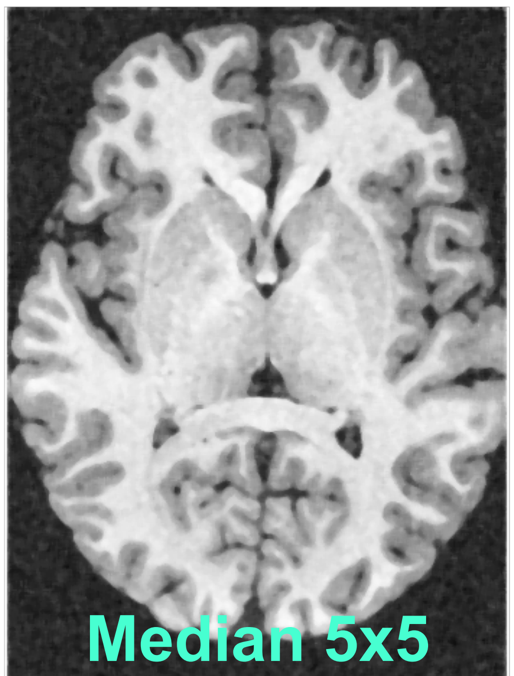
adding restraining force :

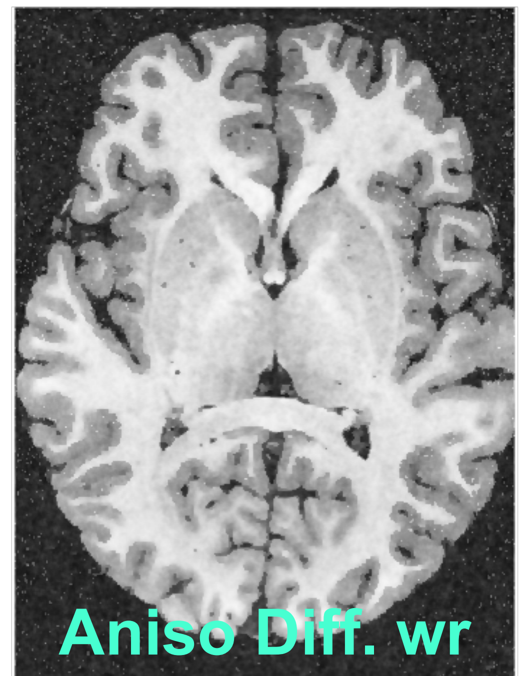
$$\frac{\partial f}{\partial t} = \Delta \cdot (c(|\nabla f|)\nabla f) - \frac{1}{\sigma^2}(f - i)$$

Computer Vision



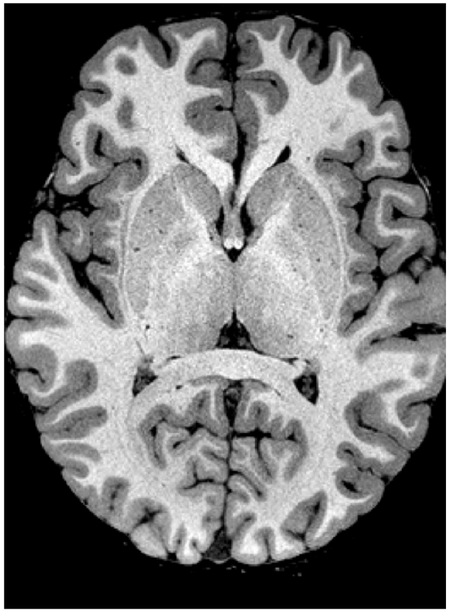



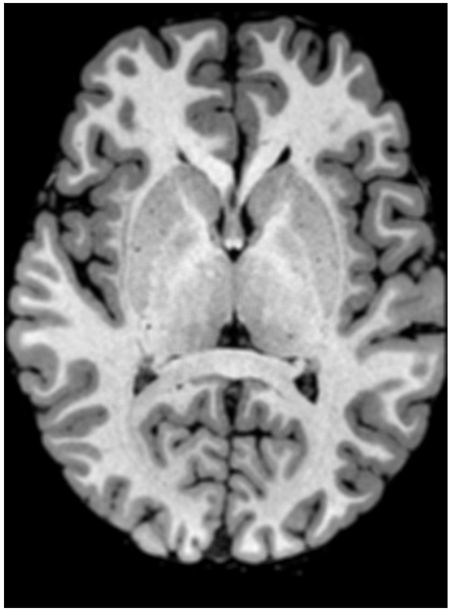
# Overview

1. Preliminaries
   a. Reminders from previous course
   b. Fourier power spectra of images
2. Noise suppression
   a. Convolutional (Linear) filters
   b. Non-linear filters
3. **Image de-blurring**
   **a. Unsharp masking**
   **b. Inverse Filtering**
   **c. Wiener Filters**
4. Contrast enhancement
   a. Histogram Equalization

Deblurring



Original Image
What we want



Blurred image
What we observe

# Unsharp masking

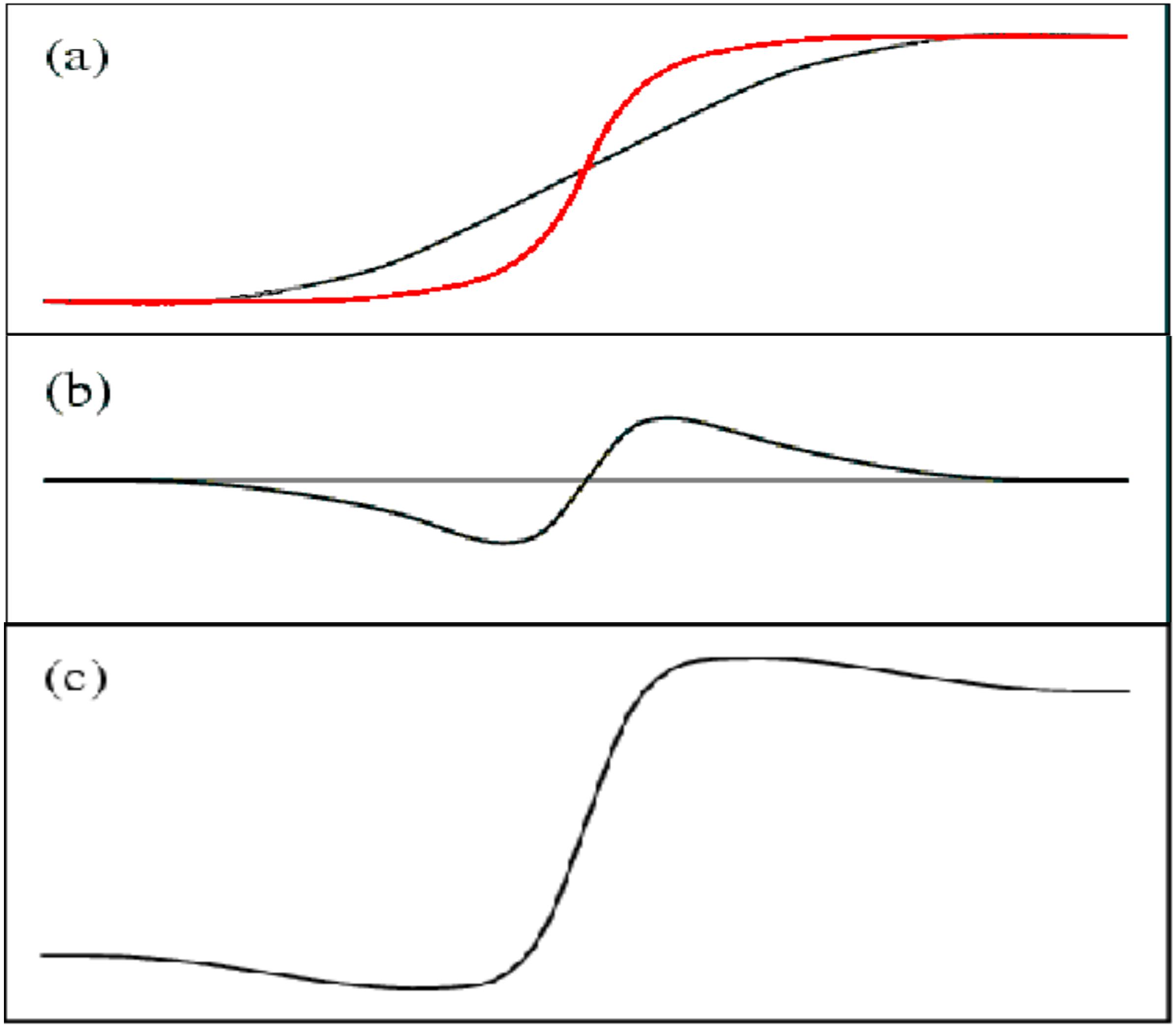
simple but effective method

image independent

linear

used e.g. in photocopiers and scanners

# Unsharp masking : sketch




red = original
black = smoothed

orginal – smooth

original + difference

Computer Vision

# 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*

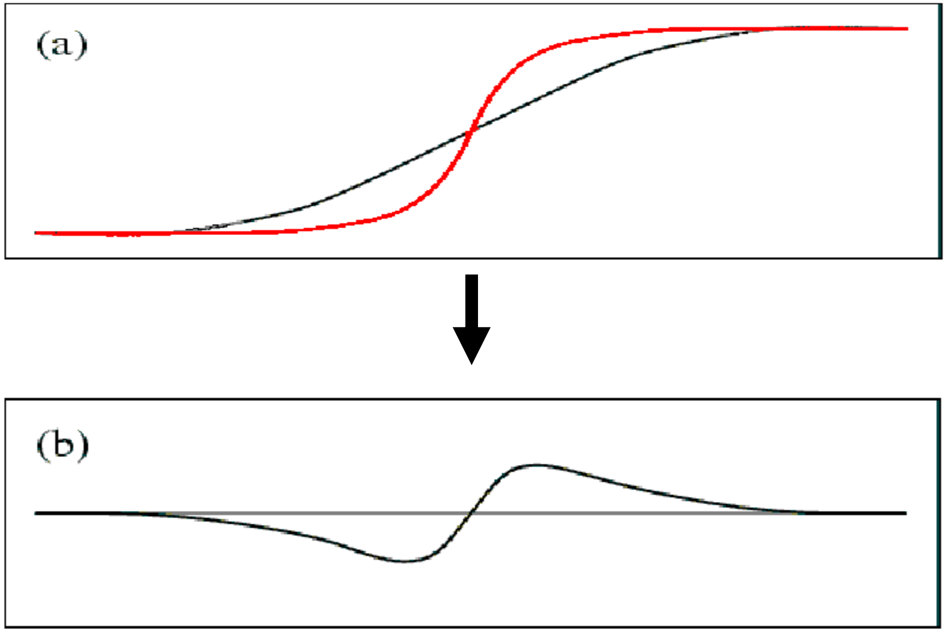
$$o = i - k\nabla^2 i$$

with *k* a well-chosen constant

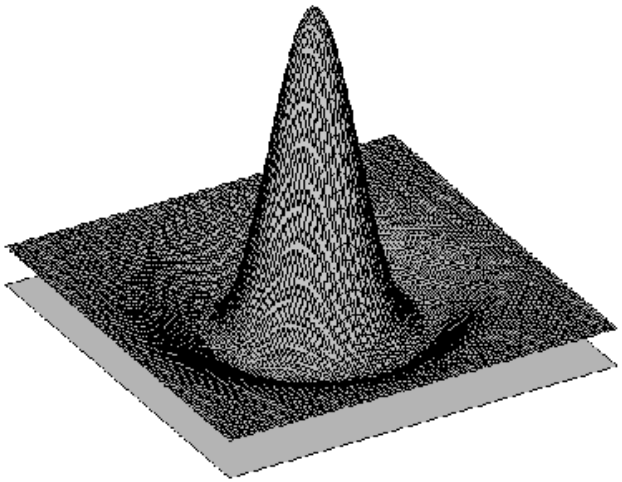
# Need to estimate $\nabla^2 i(x,y)$

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

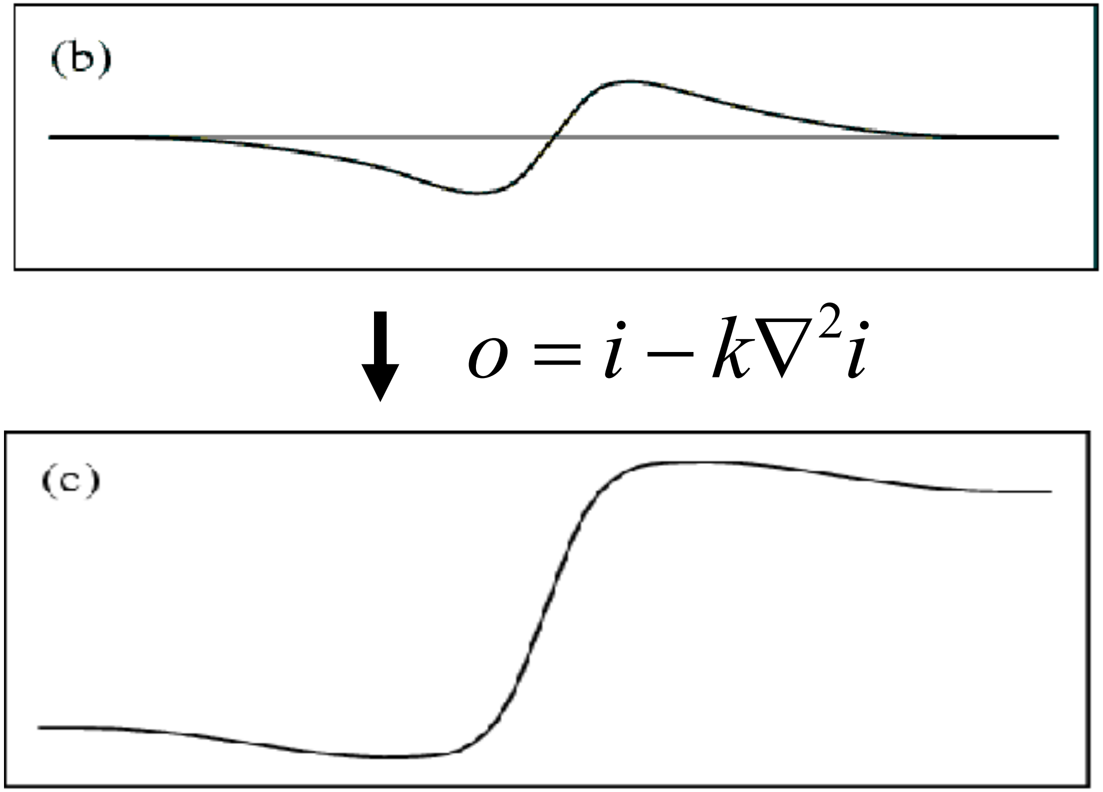
*Our 1D example:*




*Convolution mask in 2D:*



# Unsharp masking: Analysis




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

The edge profile becomes steeper, giving a sharper impression

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

Computer Vision

# Unsharp masking : images

# Inverse filtering

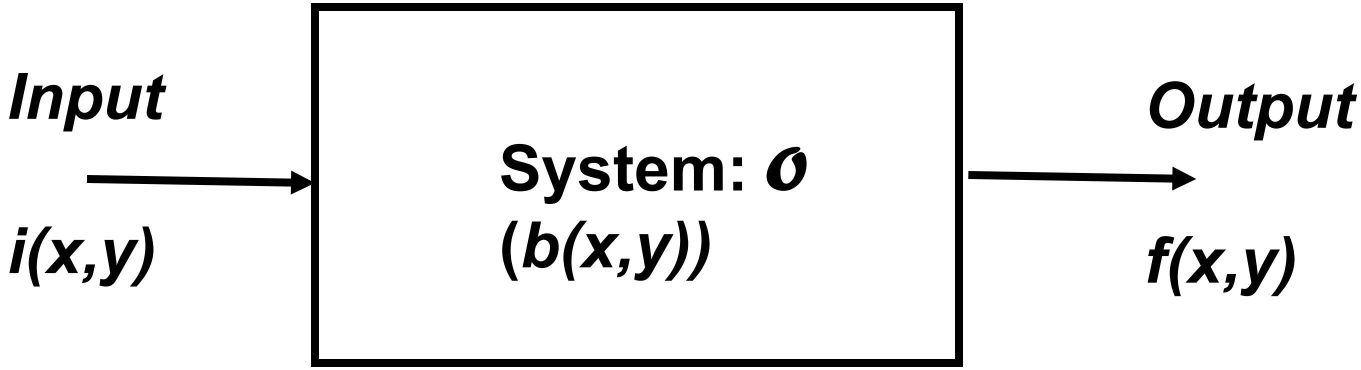
Relies on system view of image processing

Frequency domain technique

Defined through Modulation Transfer Function

Links to theoretically optimal approaches

# A system view on image restoration




*i,b* known *f*=?: simulation, smoothing
*i,f* known *b*=?: system identification
*b,f* known *i*=?: image restoration

for de-blurring: b is the blurring filter

# 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

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

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

Computer Vision

# Inverse filtering : formal derivation

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

For additive noise after filtering

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

Result of inverse filter

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

# Problems of inverse filtering

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

- Frequencies with $B\ (u,v)$ = 0

  Information fully lost during filtering

  Cannot be recovered

  Inverse filter is ill-defined

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

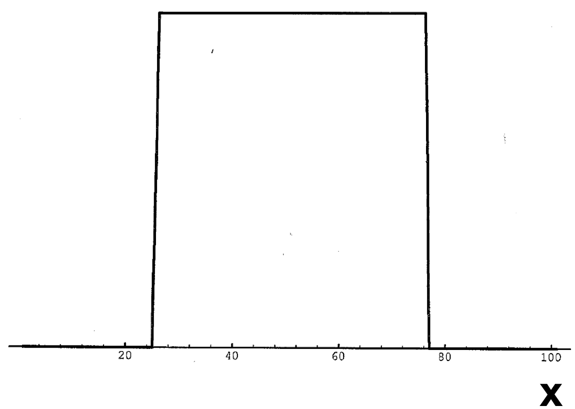
- Also problem with noise added after filtering

  $B(u,v)$ is low -> $1/B(u,v)$ is high,

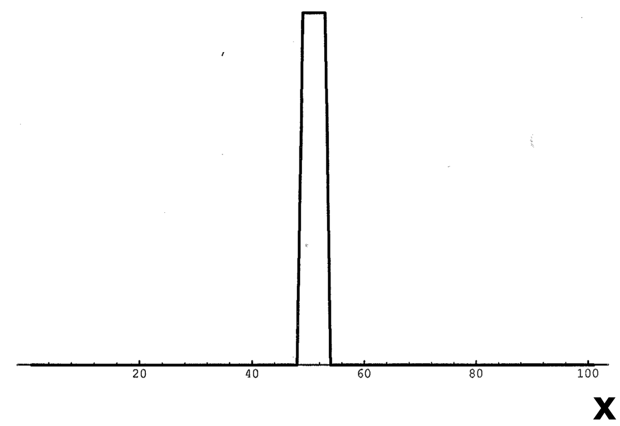
  VERY strong noise amplification

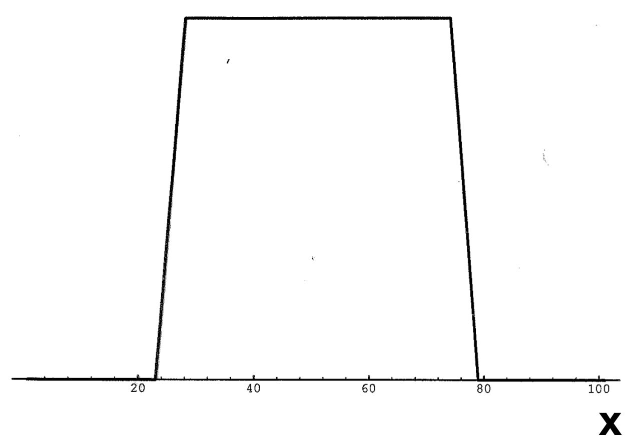
Computer Vision

# 1D Example



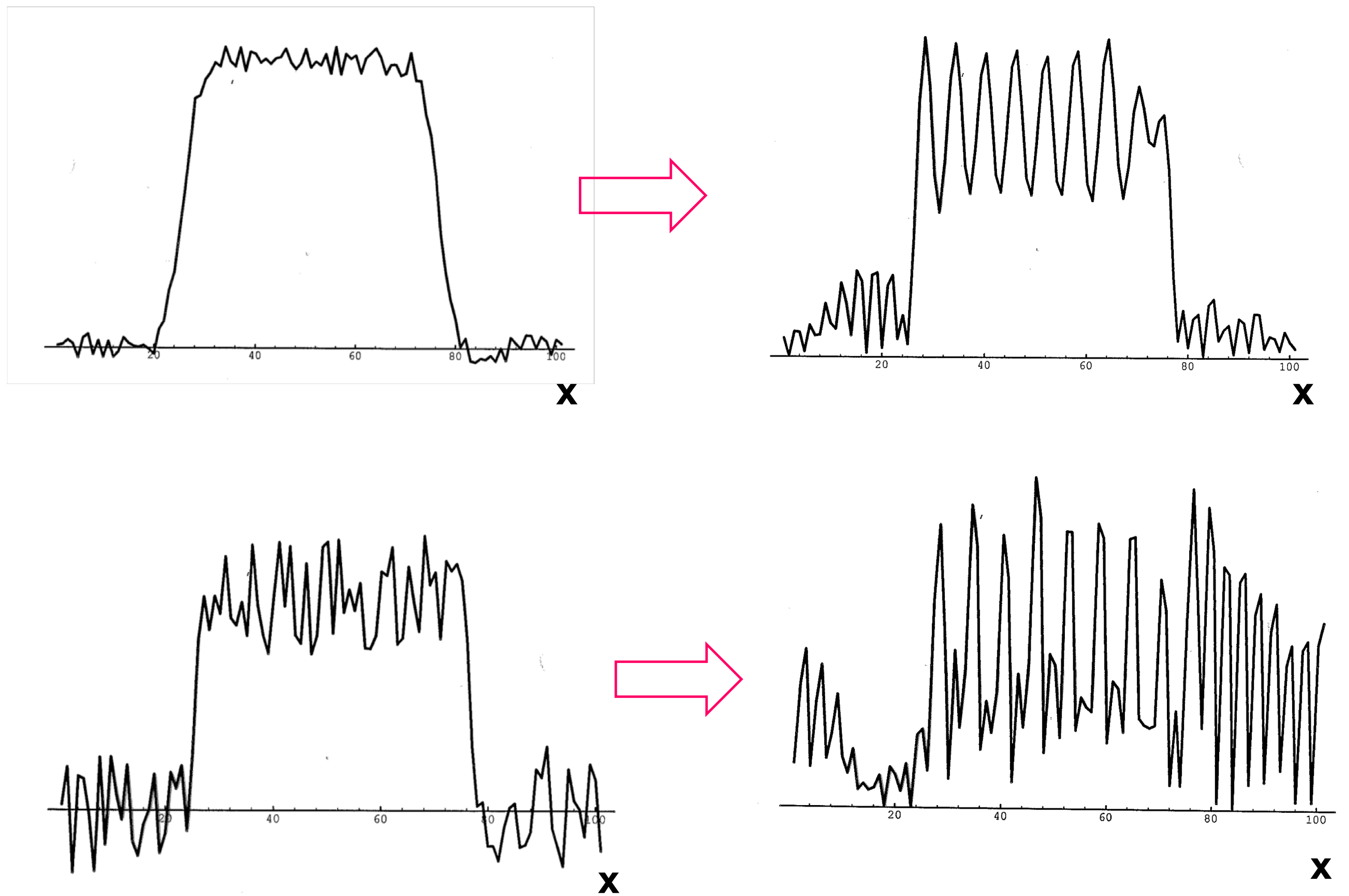

*



Computer Vision

# Restoration of noisy signals



# Inverse filtering : 2D example

we will apply the method to a Gaussian smoothed example ($\sigma$ = 16 pixels)

# Inverse filtering : 2D example

noise leads to spurious high frequencies

# The Wiener Filter

Looking for the optimal filter to do the deblurring

Take into account the noise to avoid amplification

Optimization formulation

Filter is given analytically in the Fourier Domain

# Cross-correlation

Signals *a(x,y), b(x,y),* cross-correlation

$$\phi_{ab} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} a(\xi - x, \eta - y) b(\xi, \eta) d\xi d\eta$$

Correlation between image a and b at different shifts

Difference with convolution: no mirroring
Auto-correlation: $\phi_{aa}(x,y)$: symmetric, global maximum at (*0,0*)

Power spectrum and auto-correlation are linked!

Wiener-Kintschin Theorem

$$\mathcal{F}(\phi_{aa}) = \Phi_{aa}(u,v) = |A(u,v)|^2 = A^*(u,v)A(u,v)$$

# The Wiener filter: optimal filter

Looking for the output ($o$) being most similar to the desired signal ($d$, usually the original input $i$)

This means:

$$\begin{aligned} j &= h * i + n \\ o &= h' * j \\ E &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left(o(x,y) - d(x,y)\right)^2 dxdy \end{aligned}$$

Can be solved analytically, the resulting filter in the frequency domain is

$$Wf(H) = H'(u,v) = \frac{H(u,v)\Phi_{ii}}{H^*(u,v)H(u,v)\Phi_{ii} + \Phi_{nn}}$$

where $\frac{\Phi_{ii}}{\Phi_{nn}}$ is the signal-to-noise ratio (SNR)

Computer Vision

# Behaviour of the Wiener filter

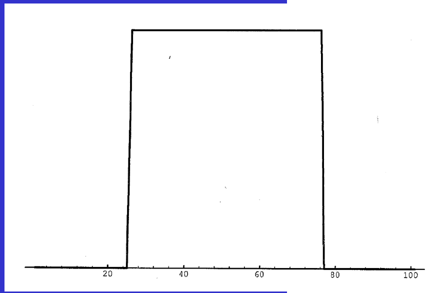
$$Wf(H) = H'(u,v) = \frac{H(u,v)}{H^*(u,v)H(u,v) + 1/\mathrm{SNR}}$$

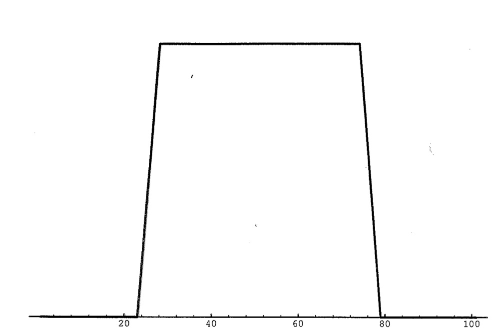
$$\mathrm{SNR} = \frac{\Phi_{ii}}{\Phi_{nn}}$$

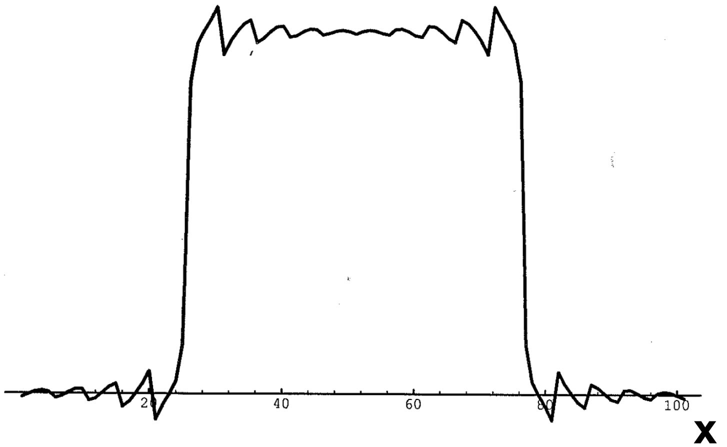
- $$H(u,v) = 0 \implies Wf(H) = 0$$ ✓
- $$\mathrm{SNR} \to \infty \implies 1/\mathrm{SNR} \to 0$$ $$Wf(H) \to \frac{1}{H}$$ ✓
- $$\mathrm{SNR} \to 0 \implies 1/\mathrm{SNR} \to \infty$$ $$Wf(H) \to 0$$ ✓

Computer Vision

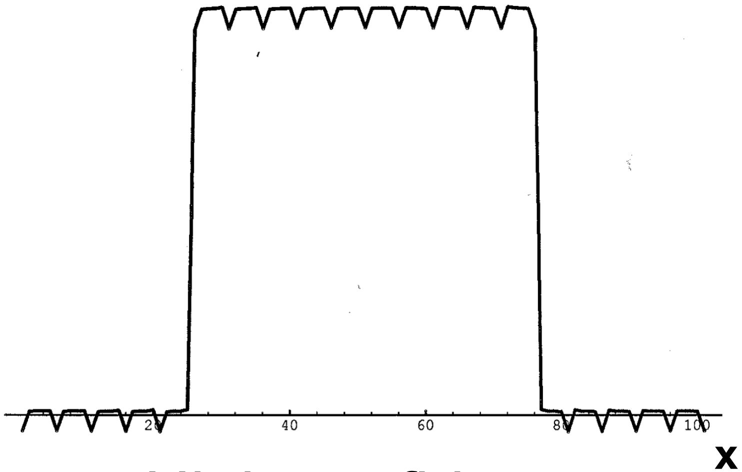
# Wiener filter: Noiseless reconstruction



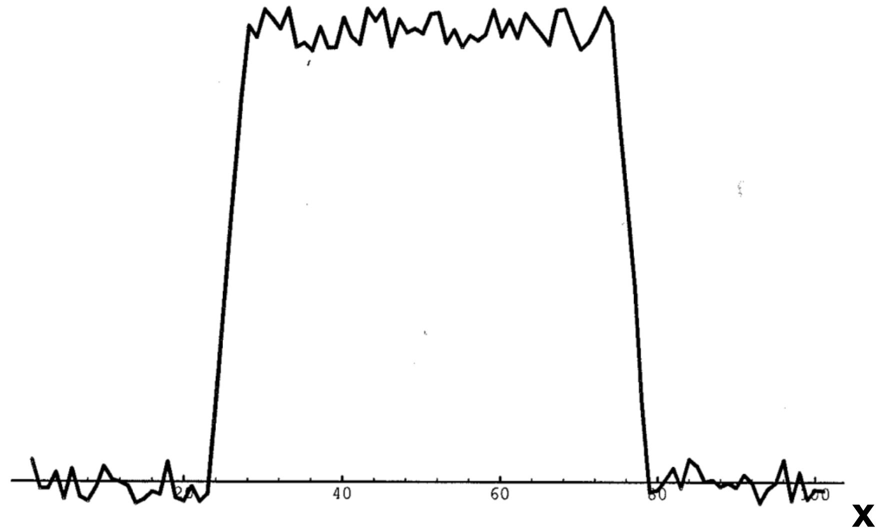



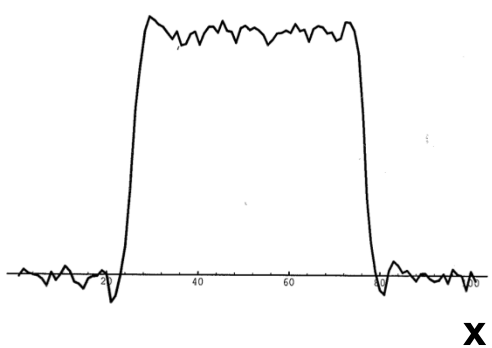

Medium confidence



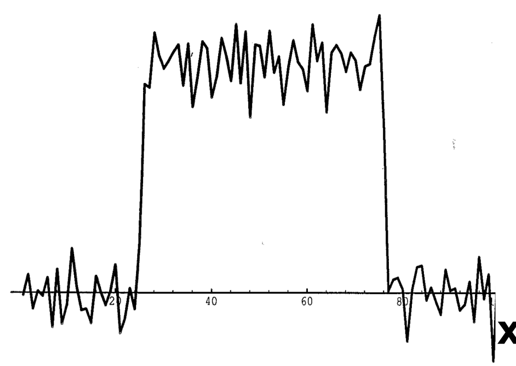

High confidence

# Wiener filter: Noisy reconstruction

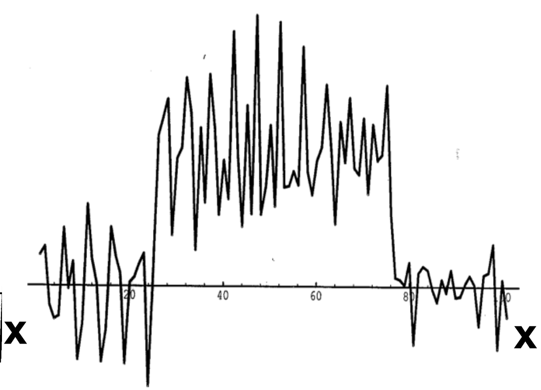



Low confidence




Medium confidence
Correct SNR




High confidence

# Wiener filtering : example

spurious high freq. eliminated, conservative

# Wiener filter: problems of application

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

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

- Conservative

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

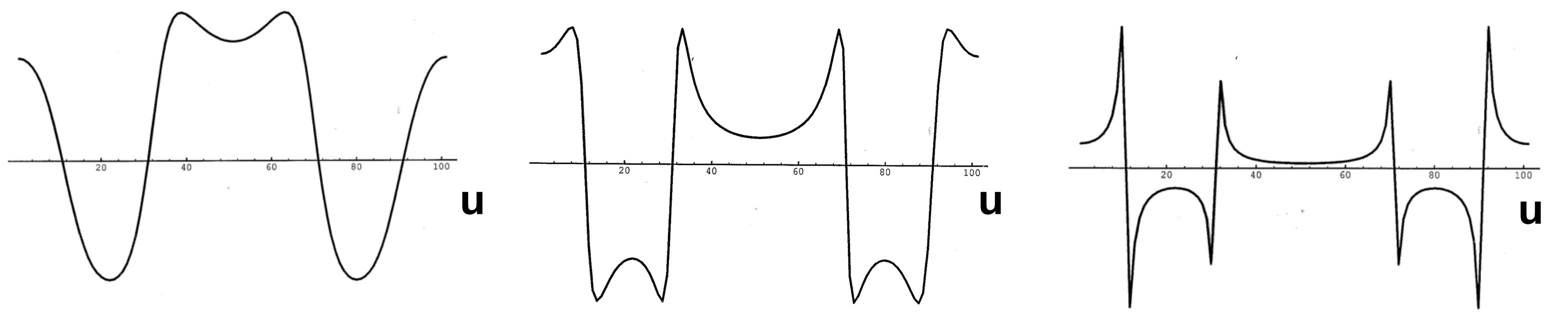
- $SNR = \Phi_{ii}(u,v)/\Phi_{nn}(u,v)$ depends on $I(u,v)$
  strictly speaking is unknown
  power spectrum is not very characteristic

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

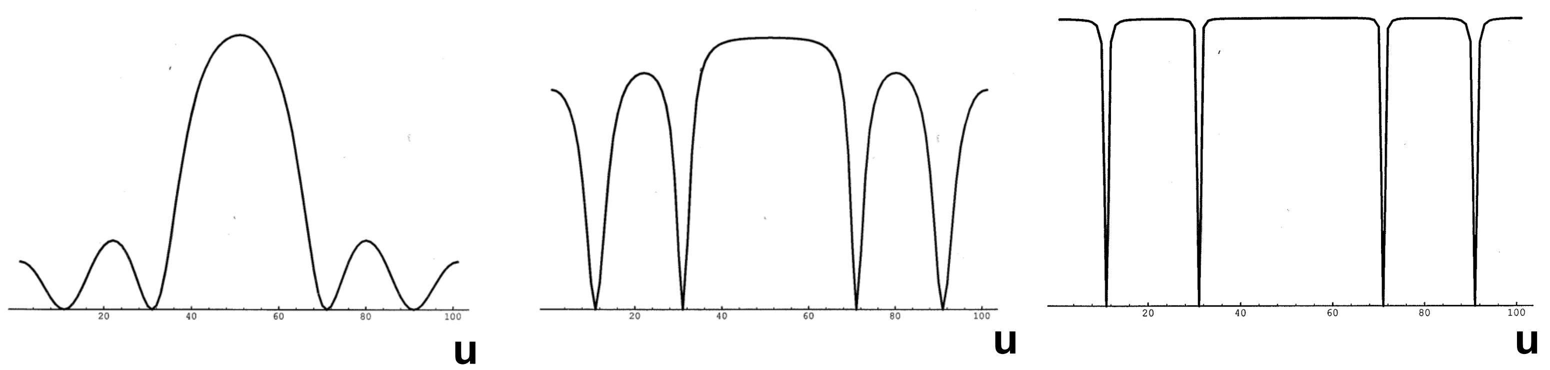
Computer Vision

# Wiener filter: the effective filter

Wiener filter




Effective filter

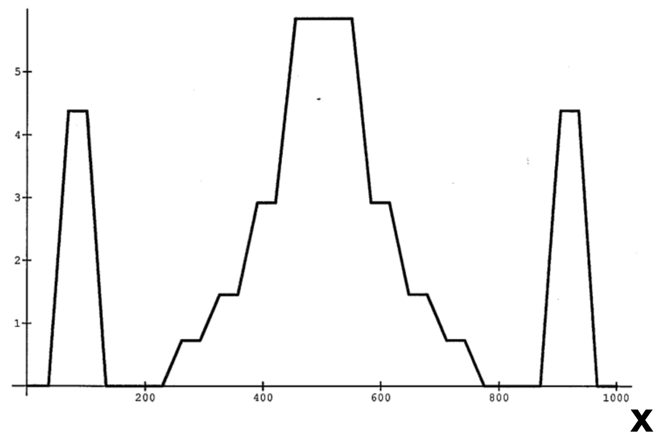



Low SNR Medium SNR High SNR

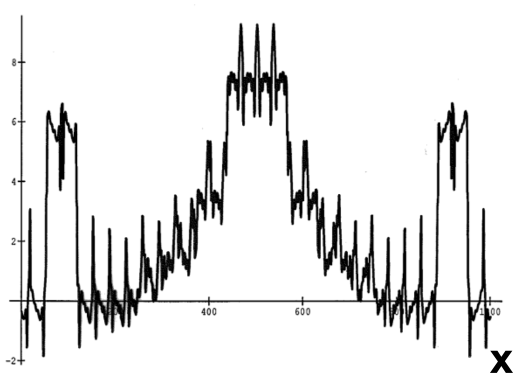
Computer Vision

# Wiener filter: Knowledge of PSF

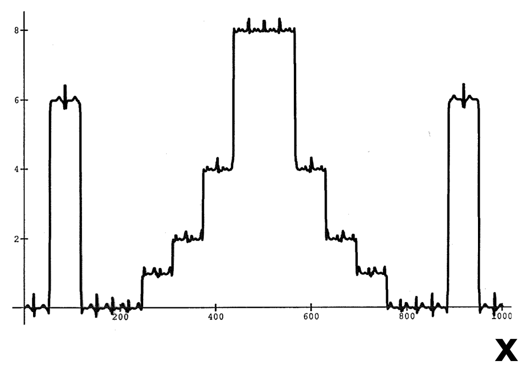
Signal blurred with *rect(x/16)*



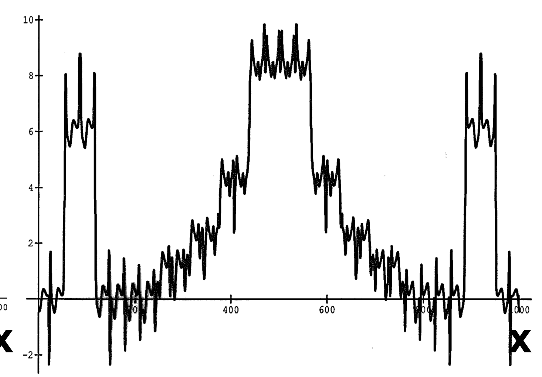

Deblurring by Wiener filter using




*rect(x/16.5)*



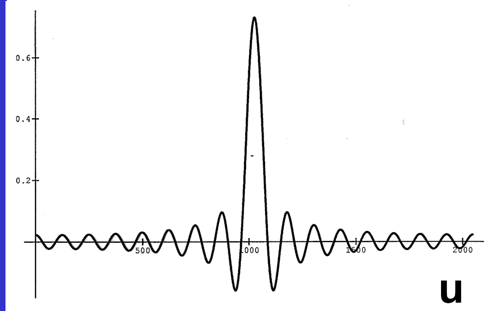

*rect(x/16)*



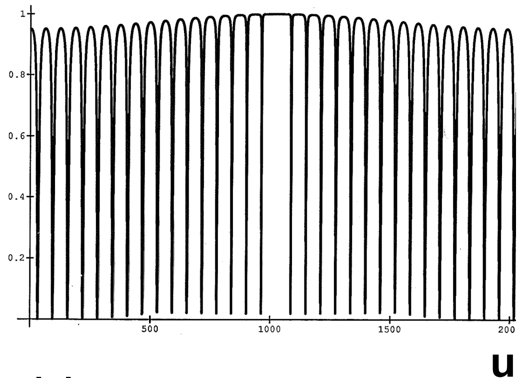

*rect(x/15.5)*

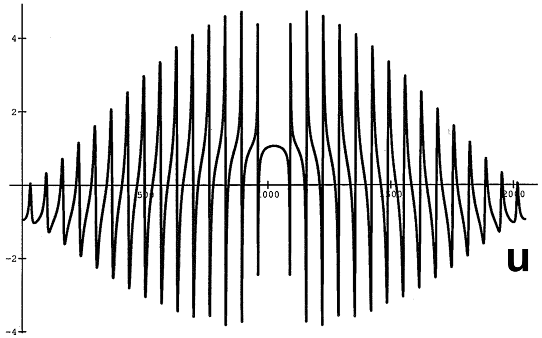
# Wiener filter: Knowledge of PSF

Blurring kernel
*rect(x/16)*

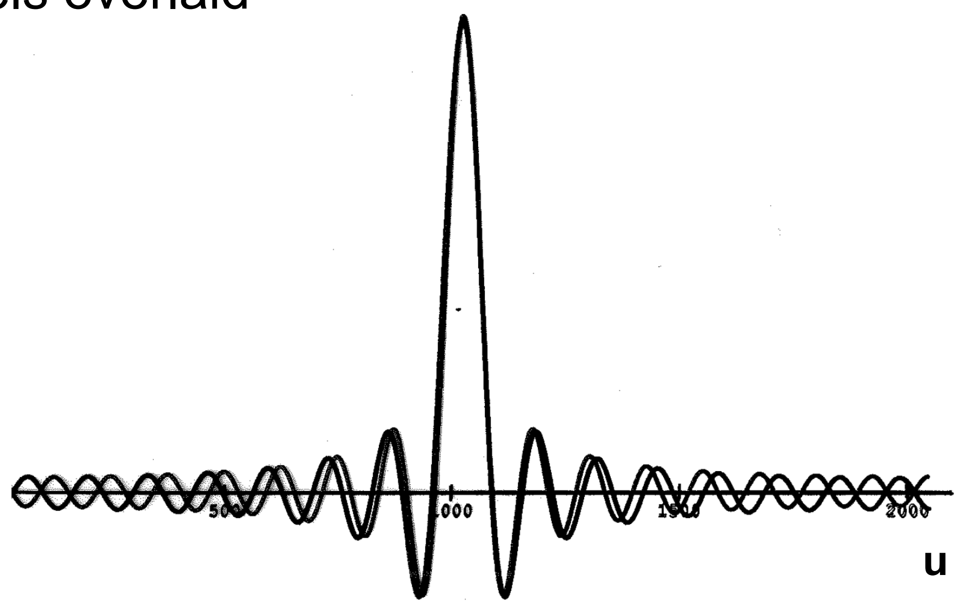



Effective Filter
*rect(x/16)* **Wf(x/16)*   *rect(x/16)* **Wf(x/16.5)*



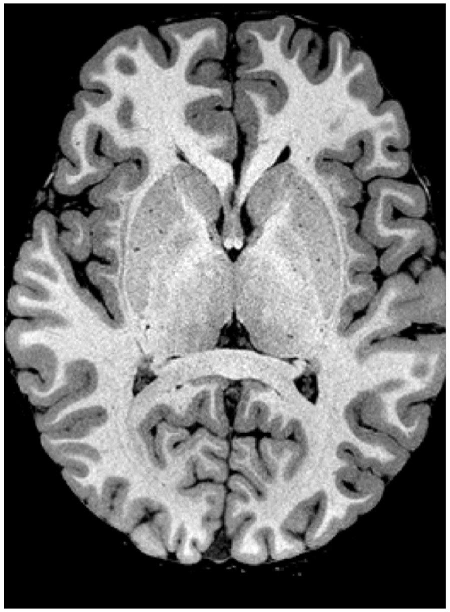

Filter kernels overlaid



# Overview

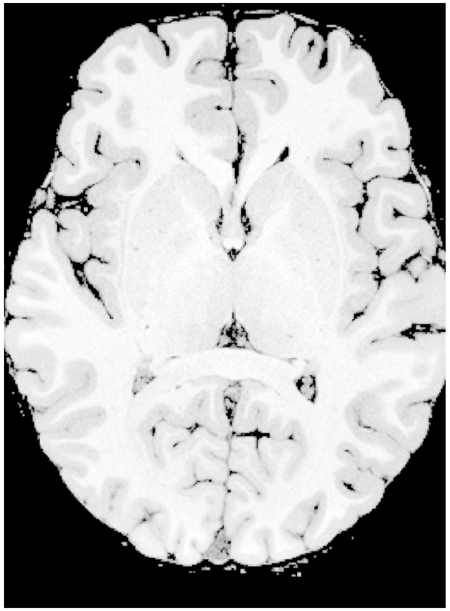
1. Preliminaries
    a. Reminders from previous course
    b. Fourier power spectra of images
2. Noise suppression
    a. Convolutional (Linear) filters
    b. Non-linear filters
3. Image de-blurring
    a. Unsharp masking
    b. Inverse Filtering
    c. Wiener Filters
4. **Contrast enhancement**
    a. **Histogram Equalization**

Computer Vision



Original Image

**Contrast Enhancement**



Observation with Bad Contrast

# Contrast enhancement

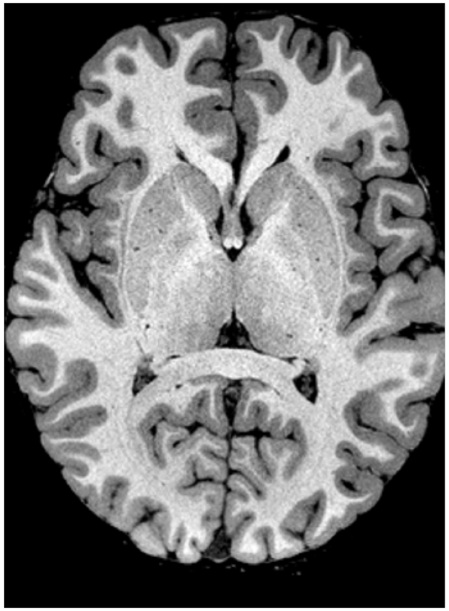
Use 1 : compensating under-, overexposure

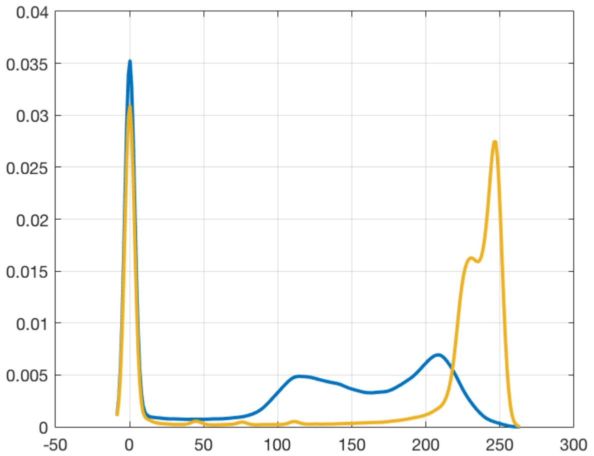
Use 2 : spending intensity range on interesting part of the image

We'll study *histogram equalisation*

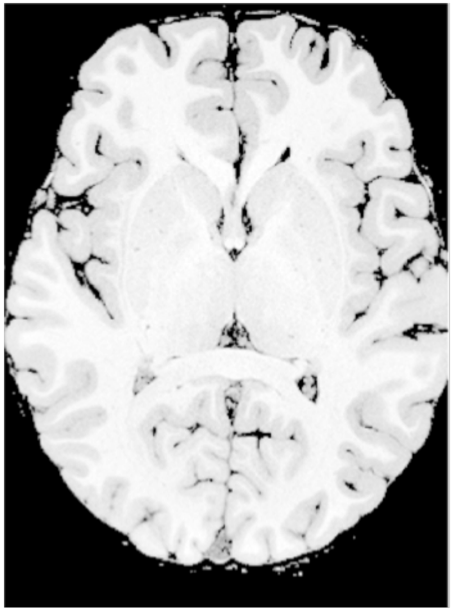
Computer Vision

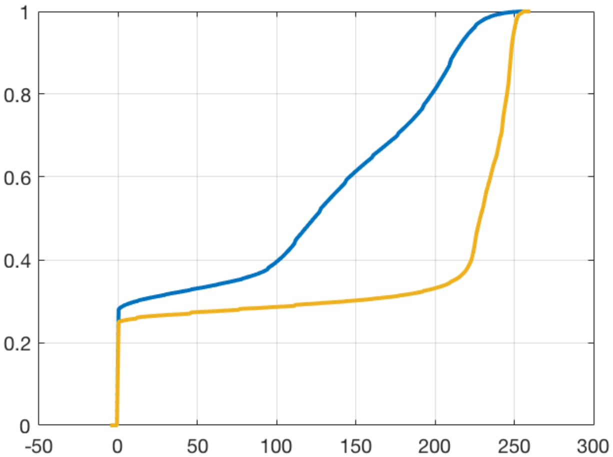
# Intensity distribution






Histogram



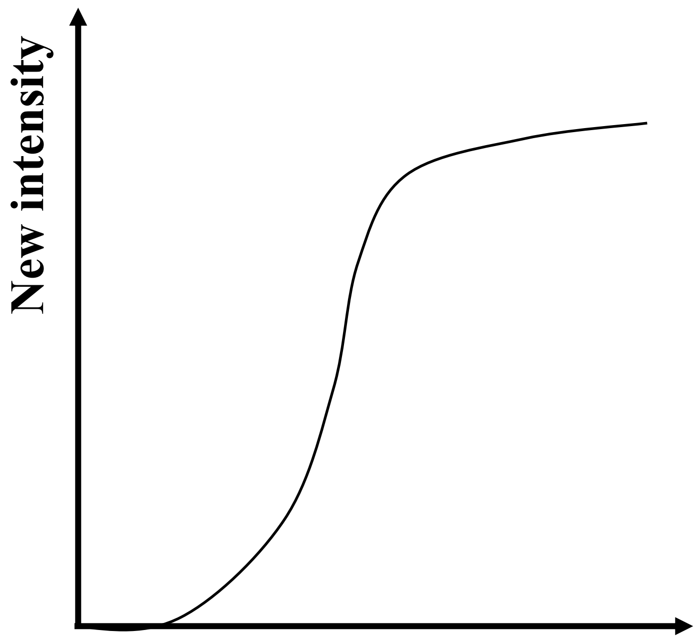



Cumulative histogram

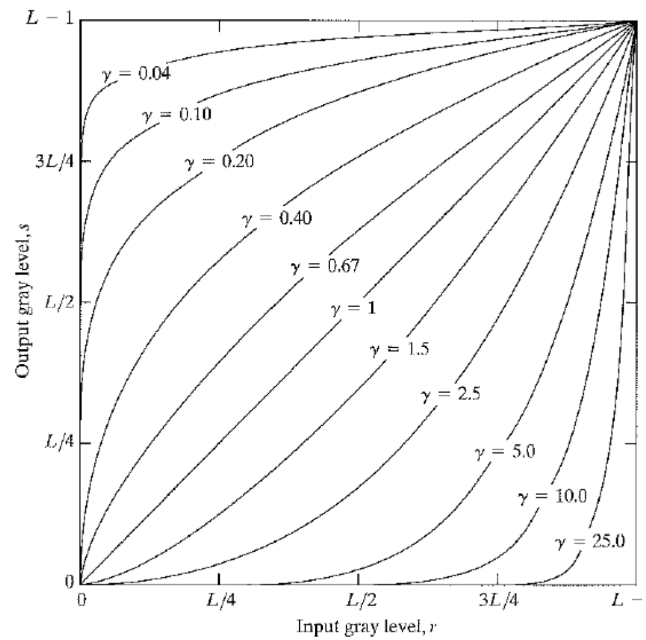
Computer Vision

# Intensity mappings

## Usually monotonic mappings required



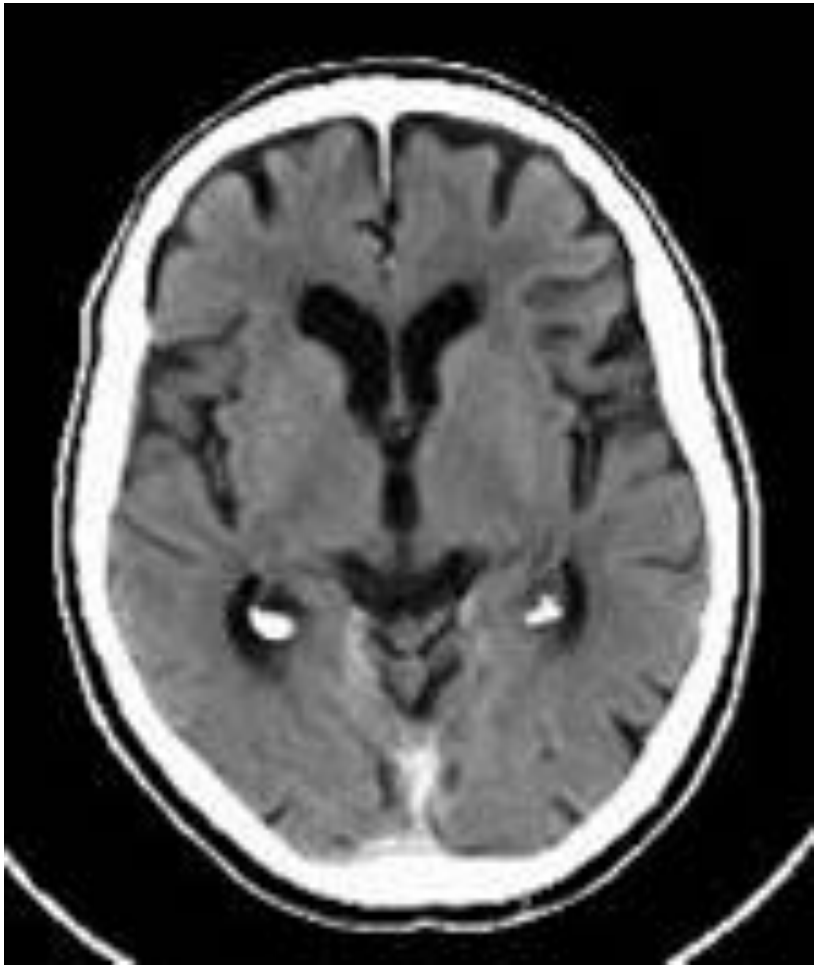

**Generic transformation function**




**Power law transformation**

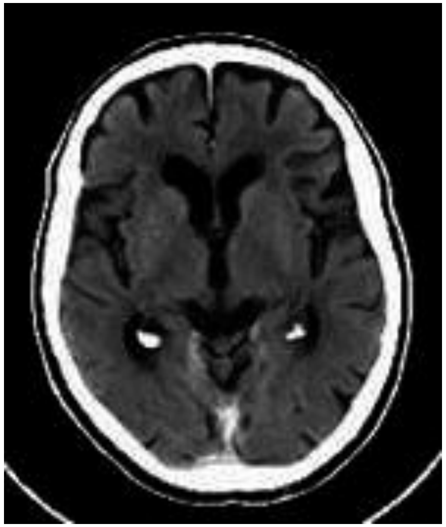
$$I_{new} = I_{old}^{\gamma}$$

# Gamma correction



**Original**

**$\gamma = 2$**

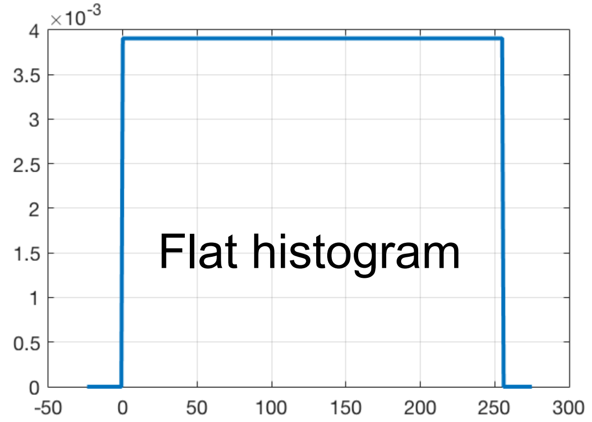


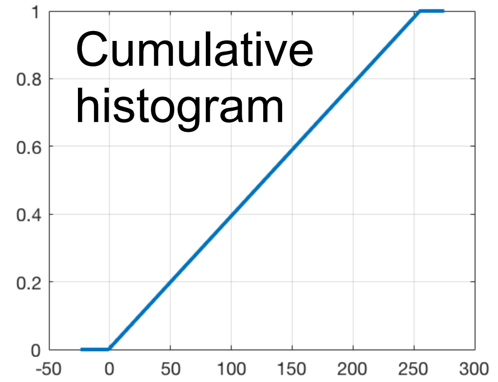
**$\gamma = 0.5$**

Computer Vision

# HISTOGRAM EQUALISATION

**WHAT :** create a flat histogram

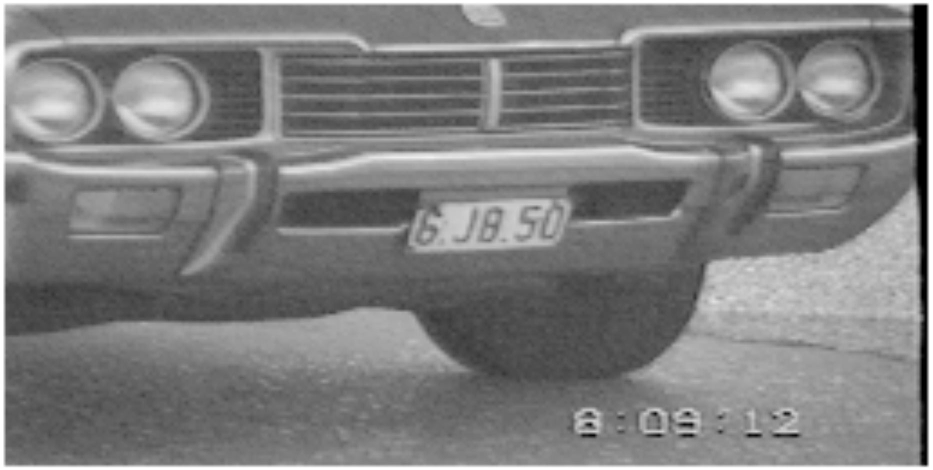



**HOW :** apply an appropriate intensity map depending on the image content

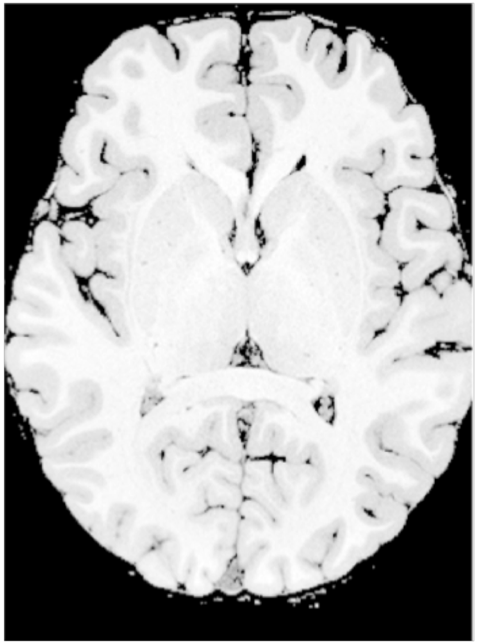
method will be generally applicable

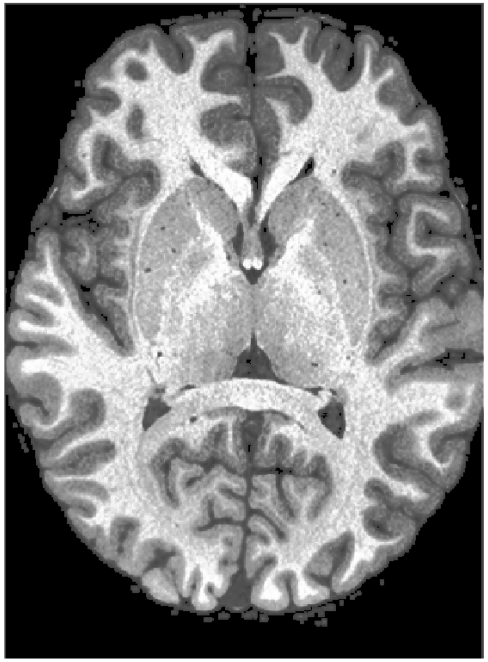
# Histogram equalisation : example




Computer Vision

# Histogram equalisation  : example

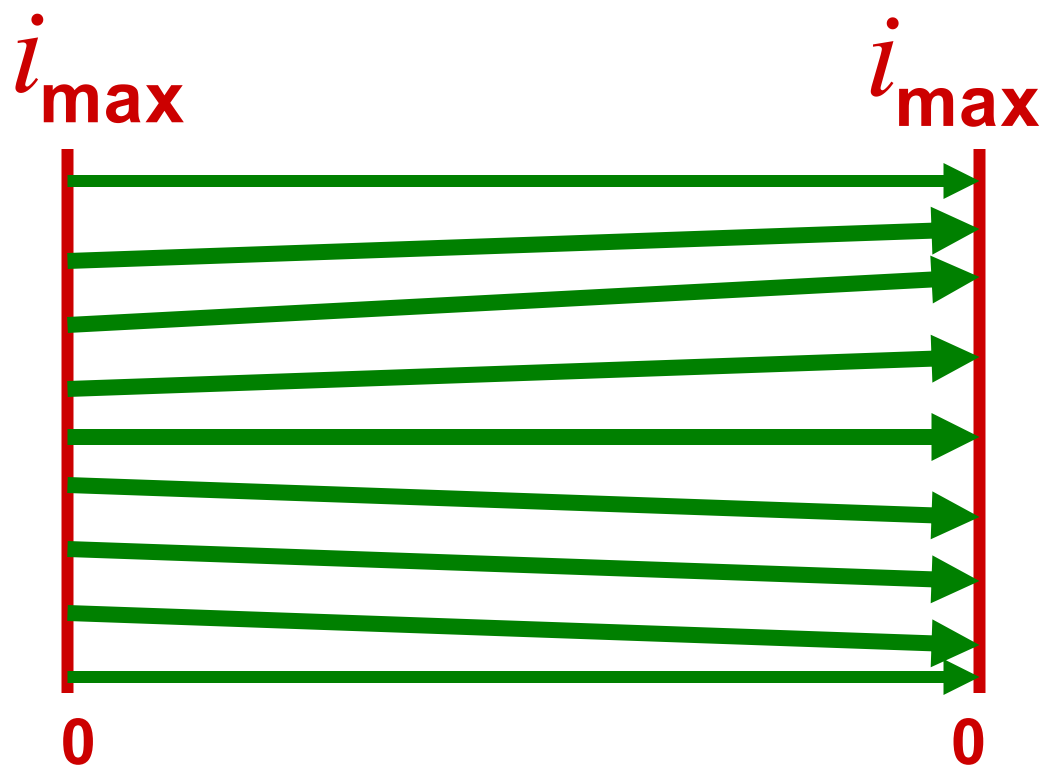




# Histogram equalisation : principle

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



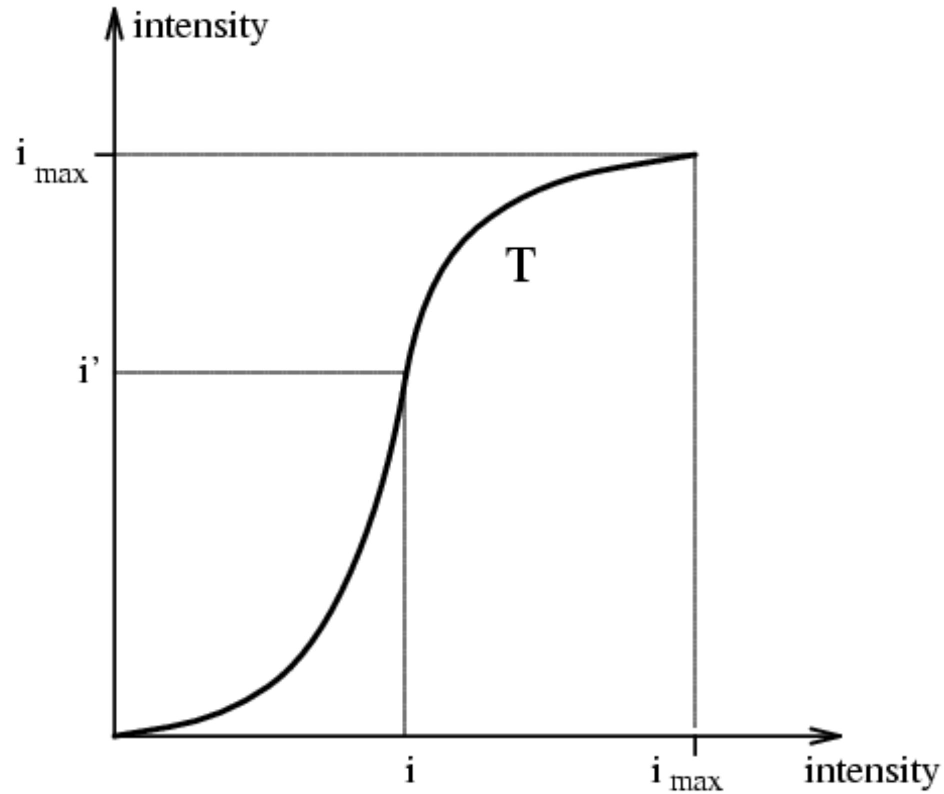
Computer Vision

# Histogram equalisation : algorithm

This mapping is easy to find:

It corresponds to the cumulative intensity probability,

i.e. by integrating the histogram from the left

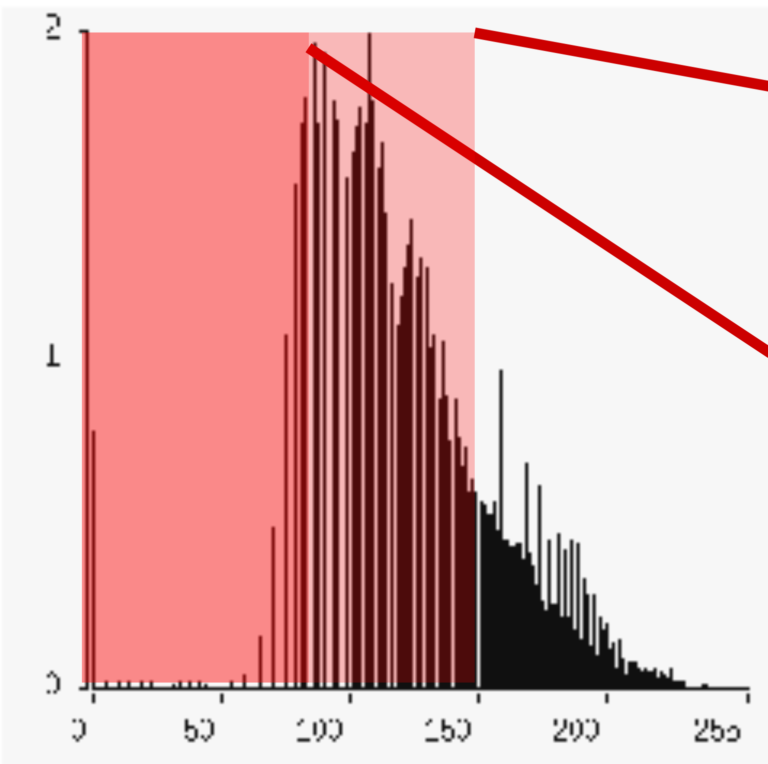


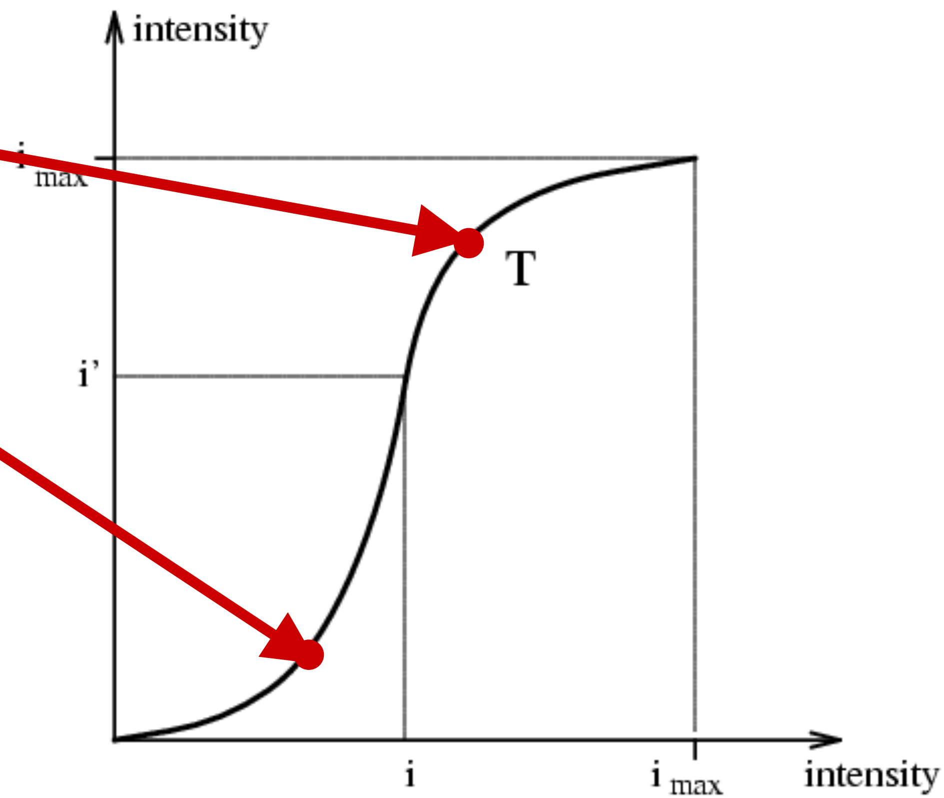
# Histogram equalisation : algorithm

This mapping is easy to find:

It corresponds to the cumulative intensity probability,

i.e. by integrating the histogram from the left



# Histogram equalisation : algorithm

suppose continuous probability density $p(i)$

cumulative probability distribution :

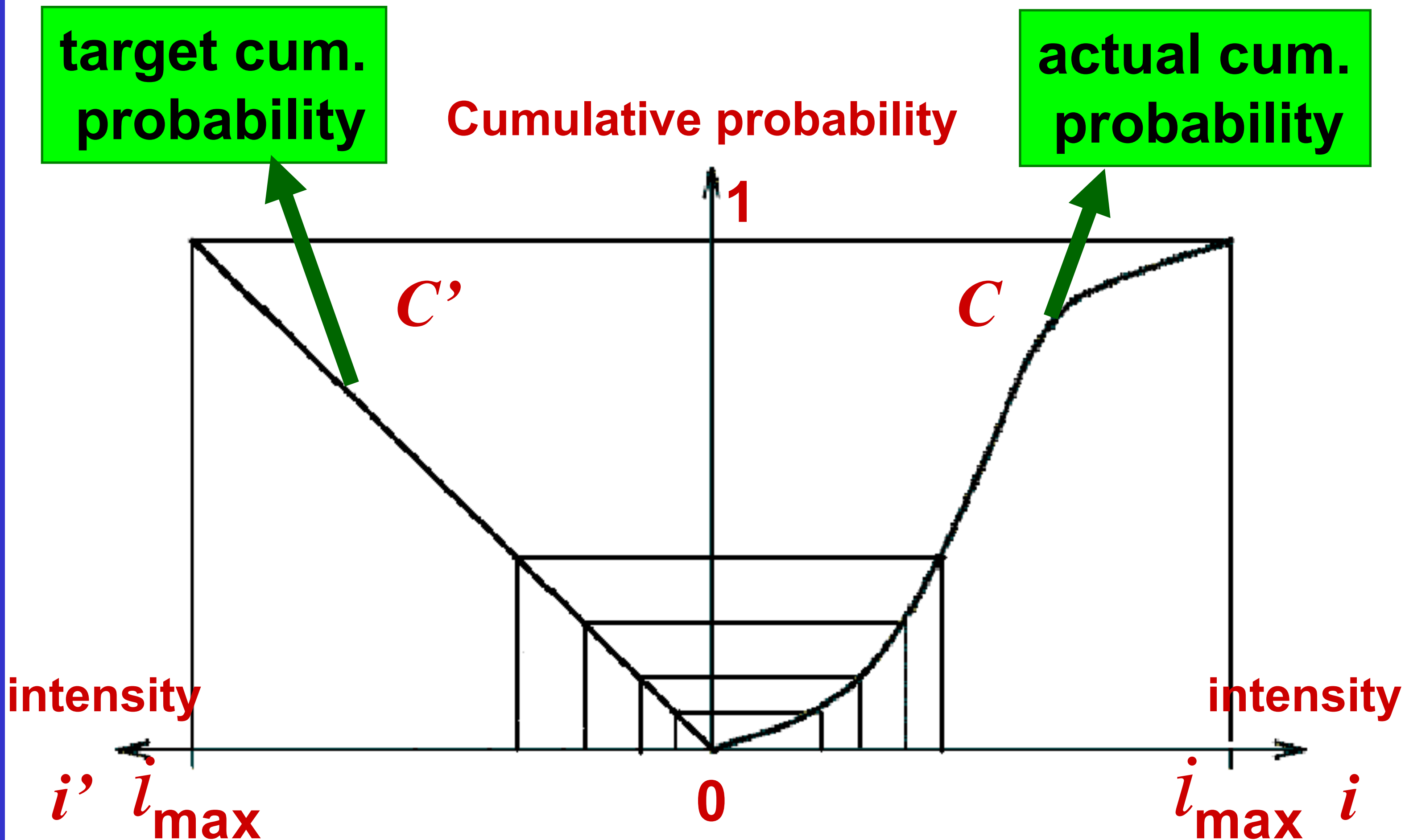
$$P(i) = \int_0^i p(i^*)di^*$$

distribution as our map $T(i)$ :

$$i' = T(i) = i_{\max} \int_0^i p(i^*)di^*$$

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

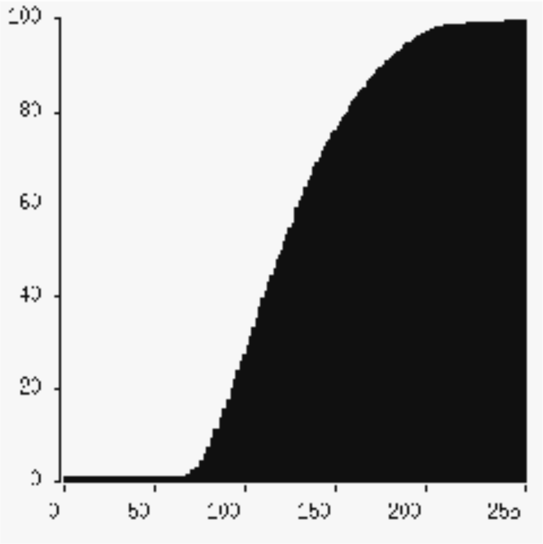
# Histogram equalisation : sketch



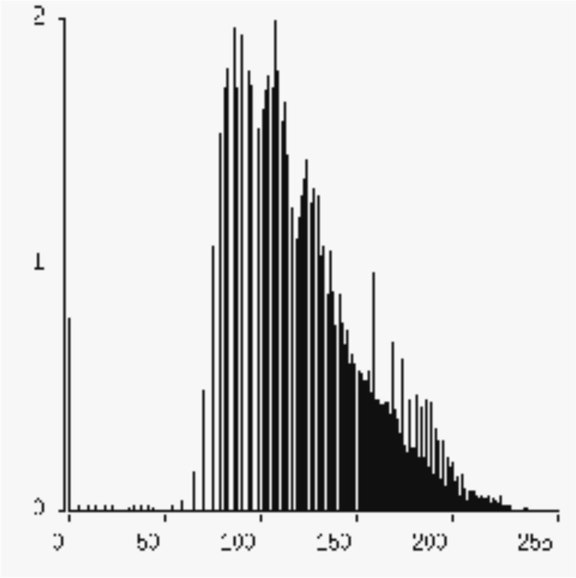

$$i' = T(i) = i_{\max} C(i) = i_{\max} \int_0^i p(i^*) di^*$$

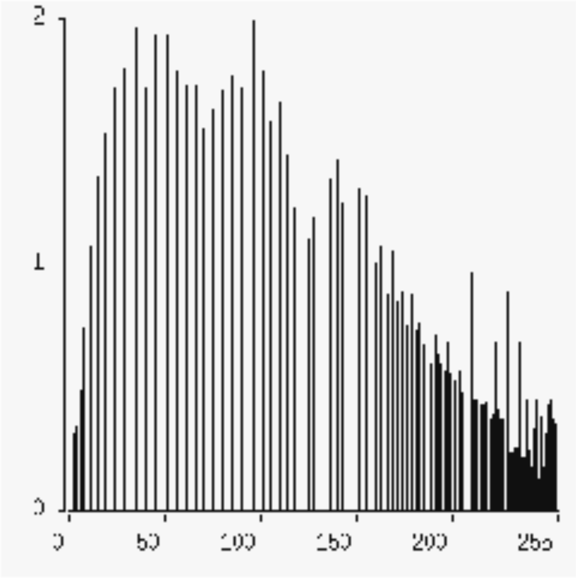
# Histogram equalisation : result

intensity map :




original and flattened histograms :

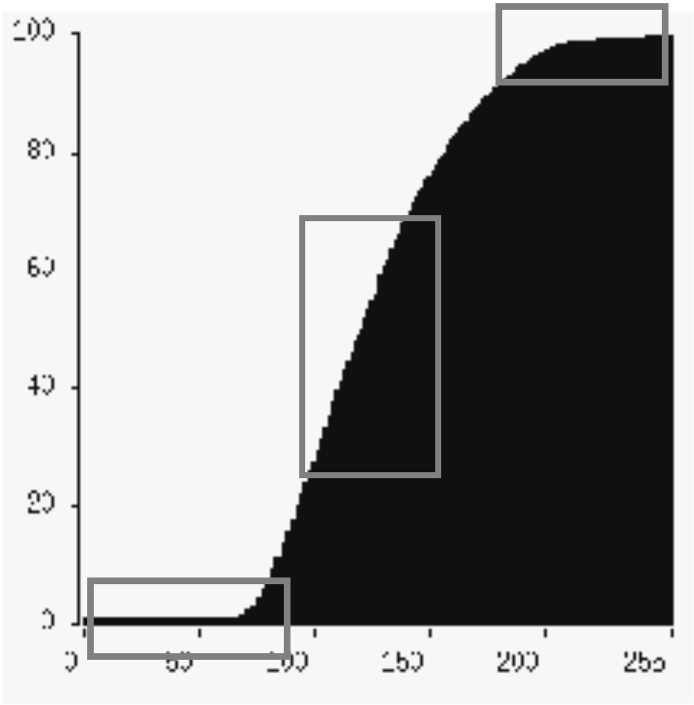


Computer Vision

# Histogram equalisation : analysis

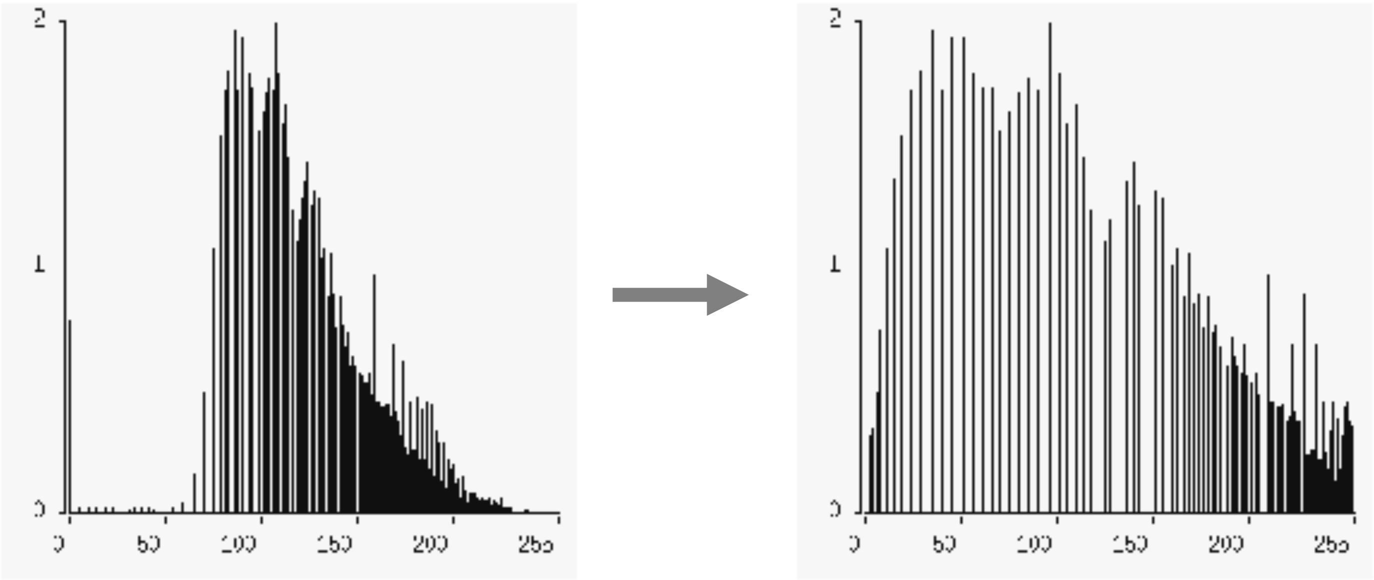
Intervals where many pixels are packed together are expanded




Intervals with only few corresponding pixels are compressed

# Histogram equalisation : analysis

… BUT we don’t obtain a flat histogram

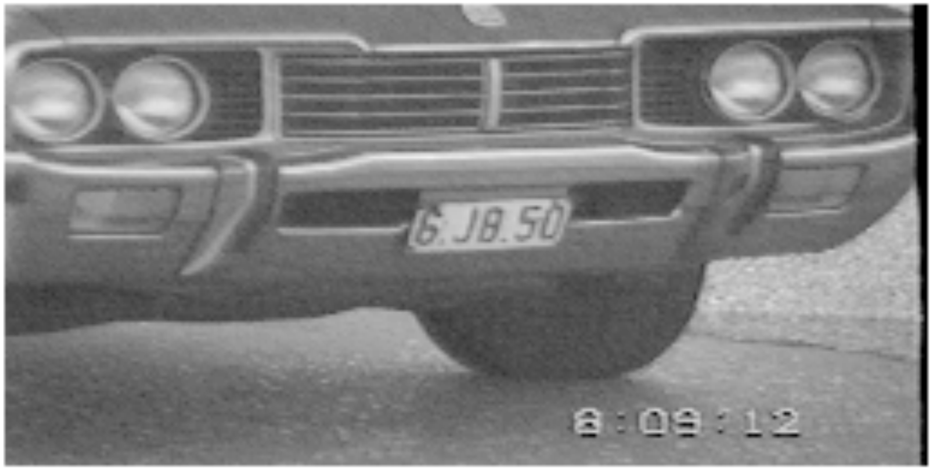



This is due to the **discrete nature** of the input histogram and the equalisation procedure

Jumps in the discretised cumulative probability distribution lead to gaps in the histogram

Computer Vision

# Histogram equalisation : example revisited




# Histogram equalisation : generalisation

Find a map $i' = T(i)$ that yields probability density $p'$

$$C'(i') = \int_0^{i'} p'(w)dw = \int_0^{i} p(v)dv = C(i).$$

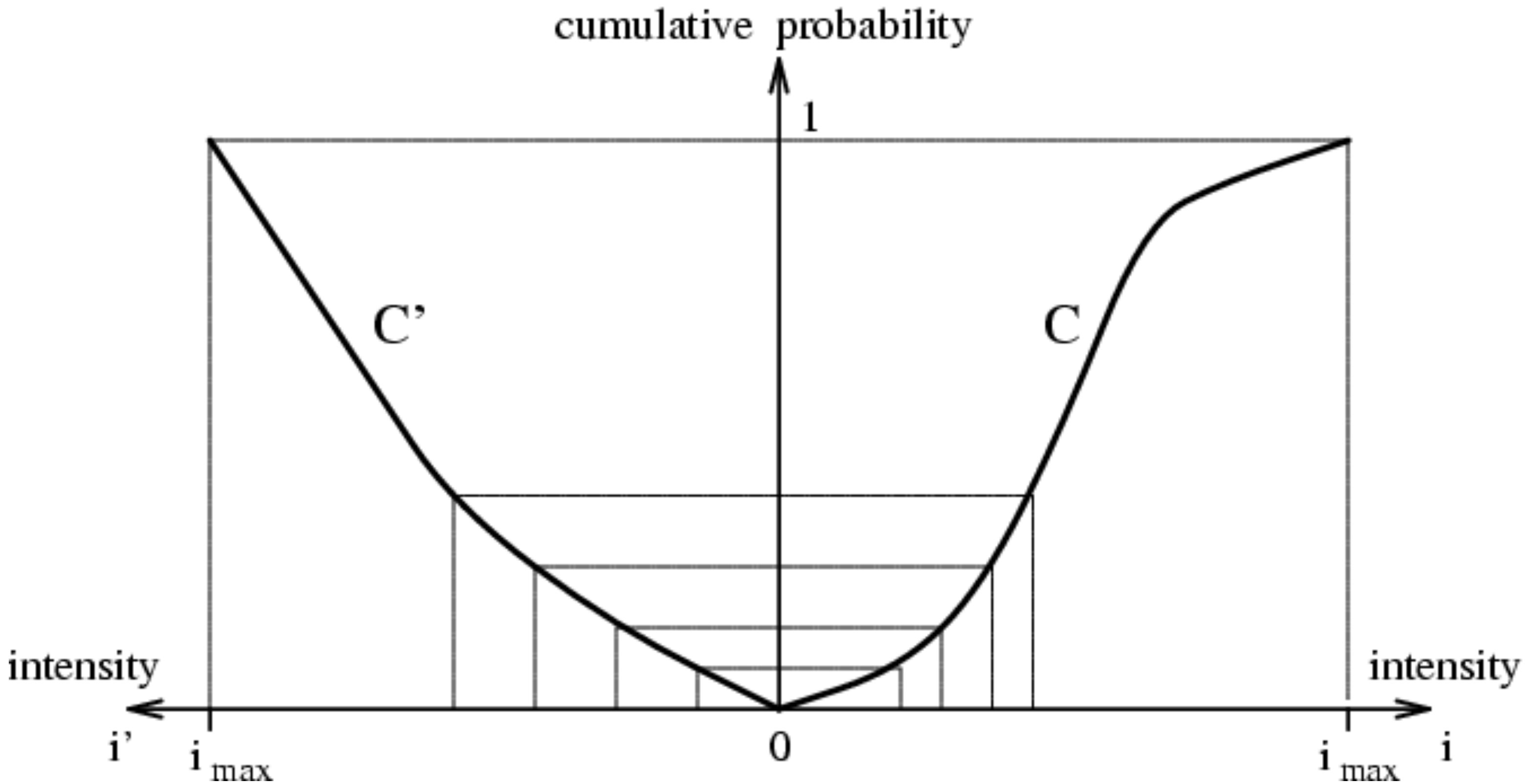
with $C'(i')$ and $C(i)$ the prescribed and original cumulative probability distributions

Thus

$$i' = C'^{-1}(C(i))$$

Computer Vision

# Histogram equalisation : sketch




$$i' = C'^{-1}(C(i))$$