



Introduction to
Computer Vision

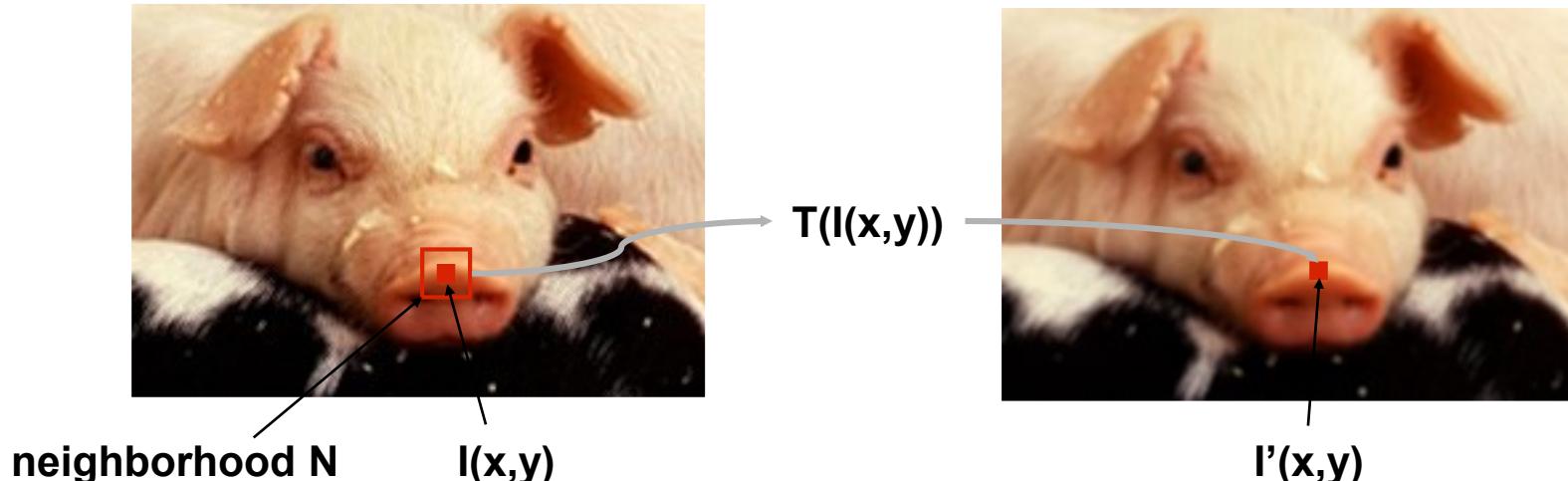
IMAGE PROCESSING

- Image Enhancement
 - Brightness mapping
 - Contrast stretching/enhancement
 - Histogram modification
 - Noise Reduction
 -
- Mathematical Techniques
 - Convolution
 - Mean filtering
 - Gaussian filtering
 - Edge and Line Detection and Extraction
 - Contour Extraction
 - Corner Detection
 - Region Segmentation

- Thresholding
 - threshold selection (manual & automatic)
- Transformations for contrast enhancement
 - linear
 - linear stretching
 - non-linear
 - power law
 - logarithmic
 - equalization
 - CLAHE

- Linear filters: mean and Gaussian
 - convolution operation
- Non-linear filters
 - median
 - anisotropic diffusion filter
 - bilateral filter
 - ...
- Frequency domain filters

- Goal: improve the ‘visual quality’ of the image
 - for human viewing
 - for subsequent processing
- Two typical methods
 - spatial domain techniques....
 - operate directly on image pixels
 - frequency domain techniques....
 - operate on the Fourier transform of the image
- No general theory of ‘visual quality’
 - General assumption: if it looks better, it is better
 - Often not a good assumption



- Transformation T
 - point - pixel to pixel
 - local - local area to pixel
 - global - output value at a specific coordinate depends on all values in the input image. (ex: DFT)
- Local - neighborhoods
 - typically quadrangular
 - typically an odd size: 3x3, 5x5, etc. (why odd size? see later)
 - centered on pixel $I(x,y)$

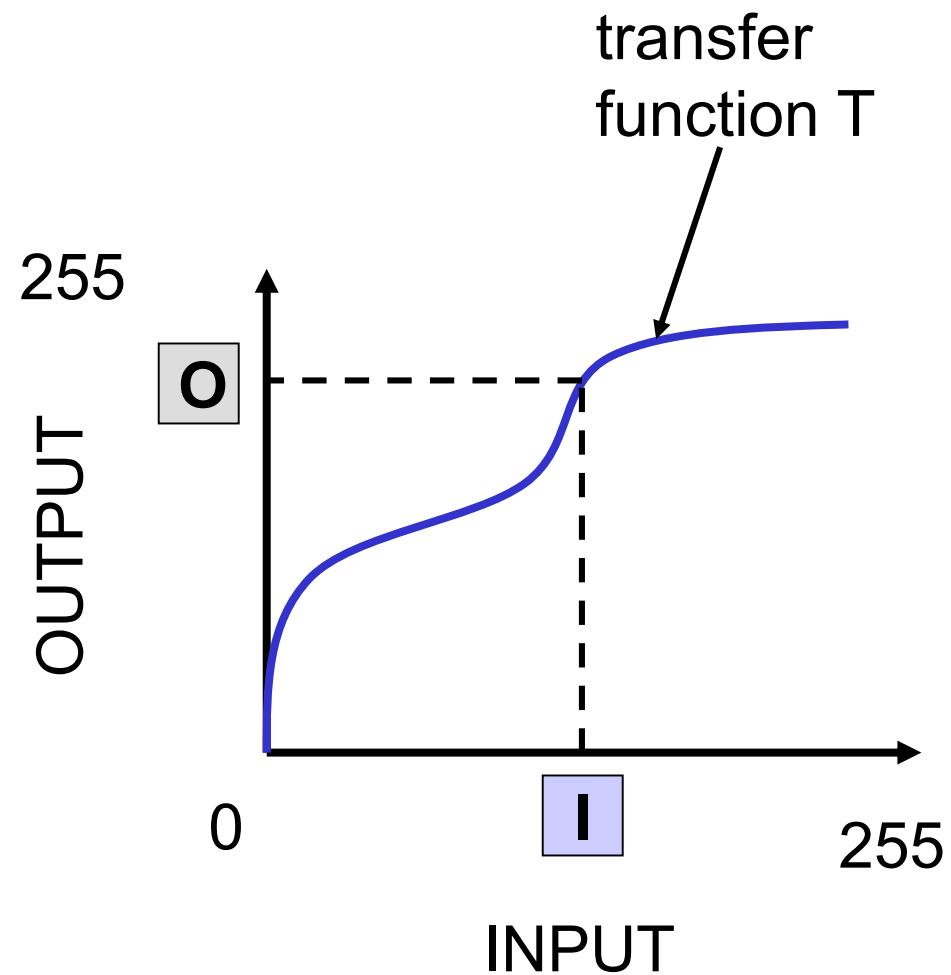
$$I'(x,y) = T(I(x,y))$$

Point transformations

Histogram-based transformations

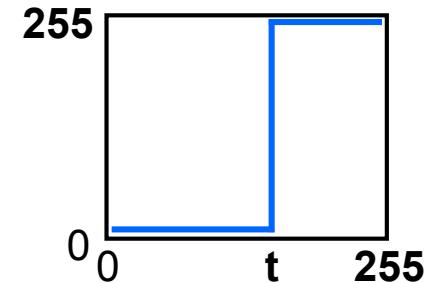
$$O = T(I)$$

Input pixel value, I ,
mapped to output
pixel value, O , via
transfer function T .



- T is a point-to-point transformation
 - only information at $I(x,y)$ used to generate $I'(x,y)$
- Thresholding

$$I'(x,y) = \begin{cases} I_{\max} & \text{if } I(x,y) > t \\ I_{\min} & \text{if } I(x,y) \leq t \end{cases}$$



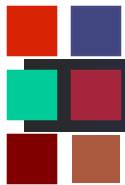
Color image



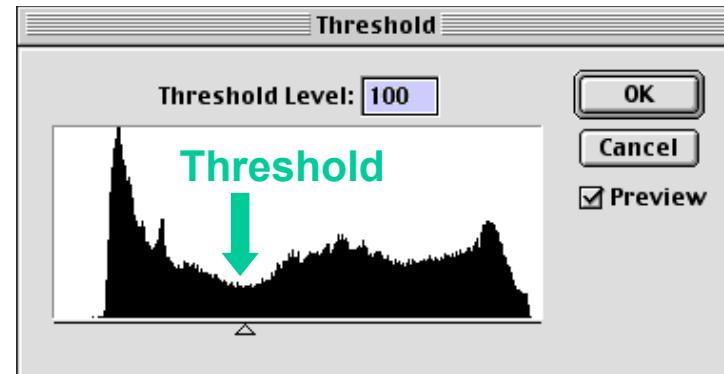
Graylevel image



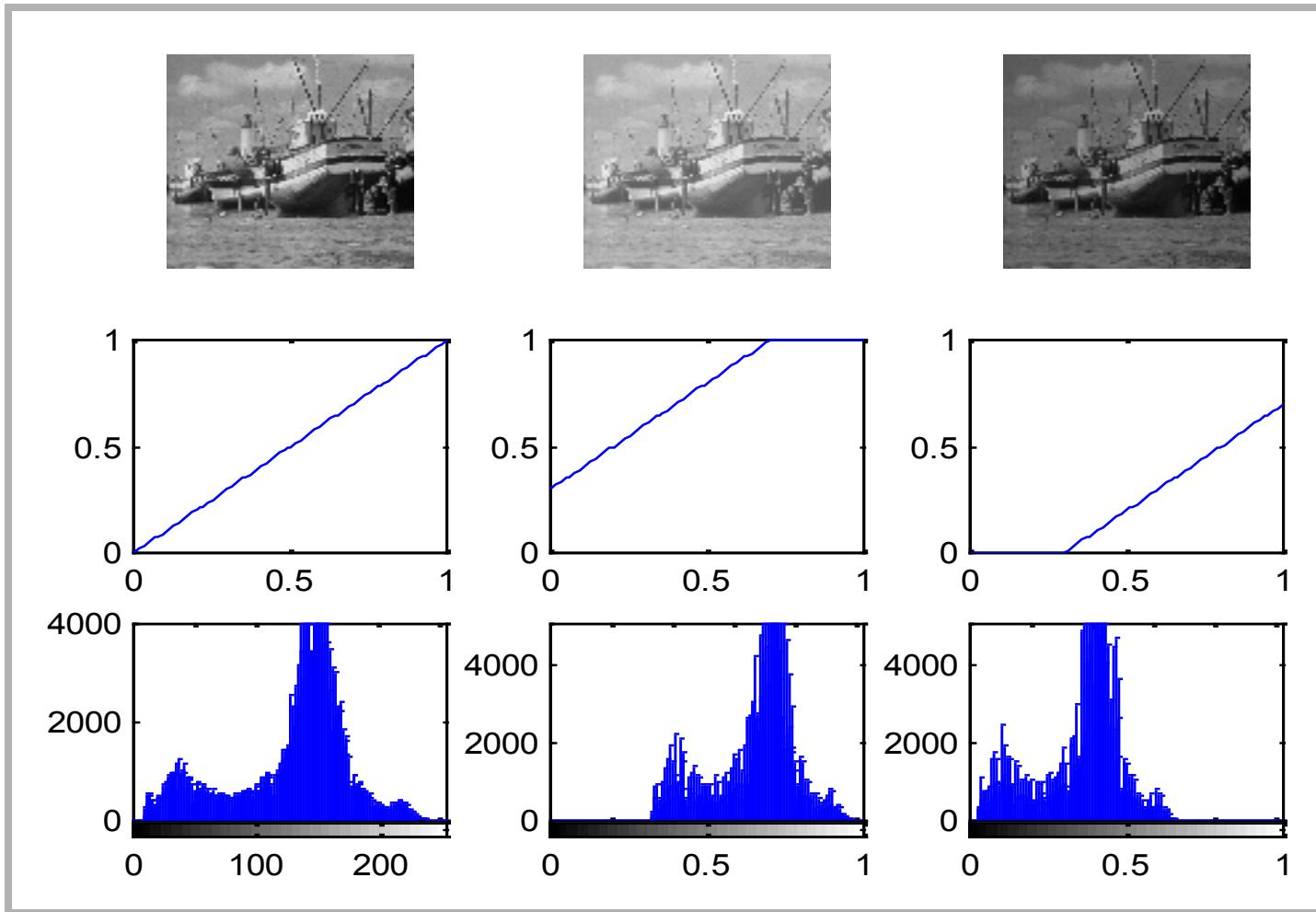
Thresholded graylevel image ($t=89$)

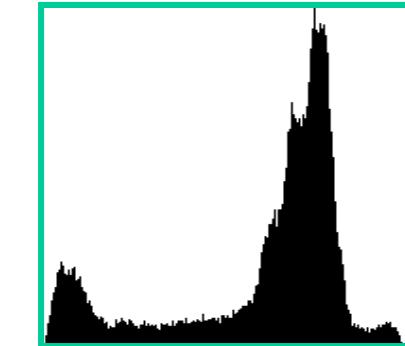
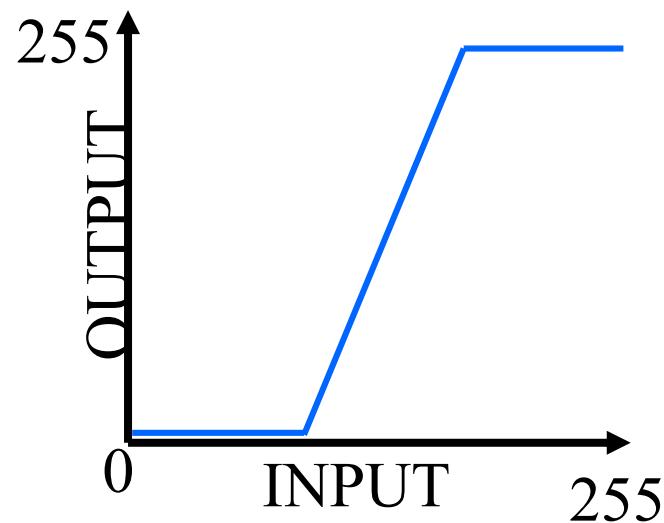
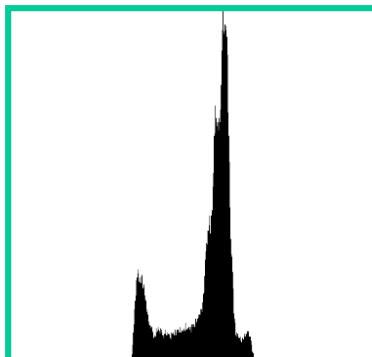


- Arbitrary selection
 - select visually
- Use image histogram



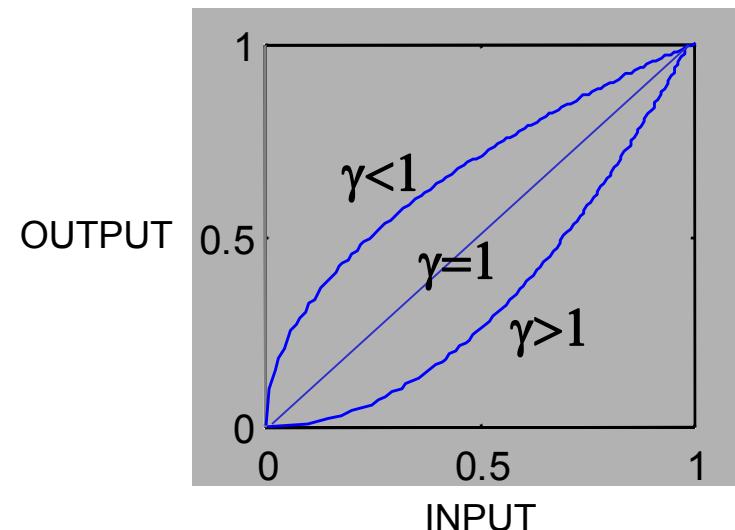
Later, we'll be back to threshold selection methods

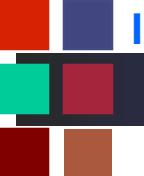




$$O = I^\gamma$$

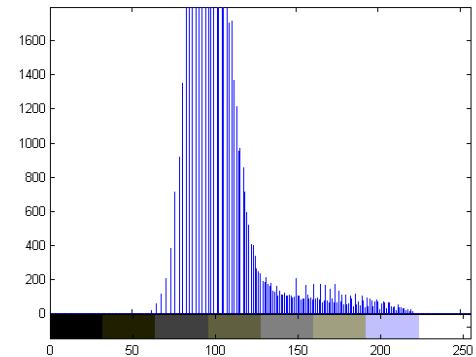
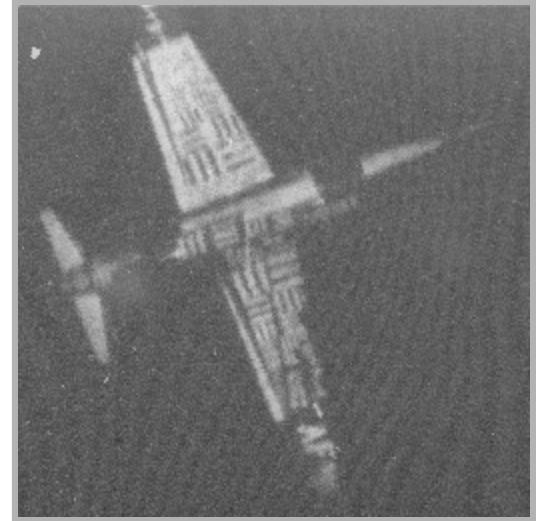
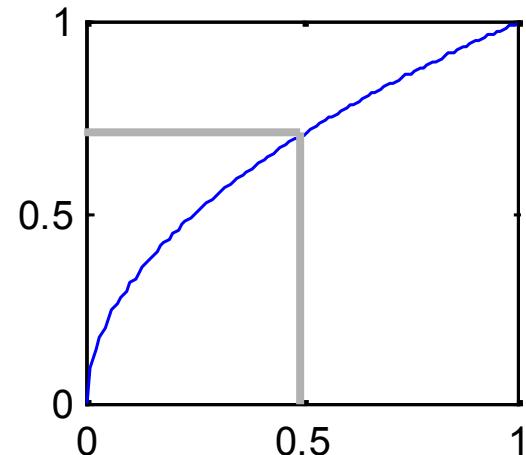
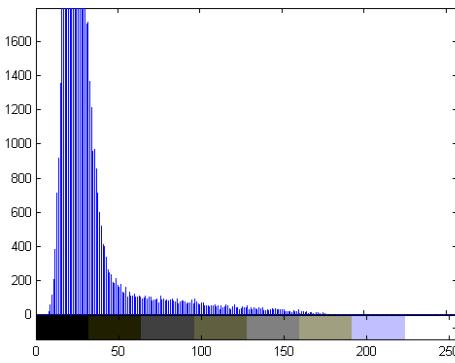
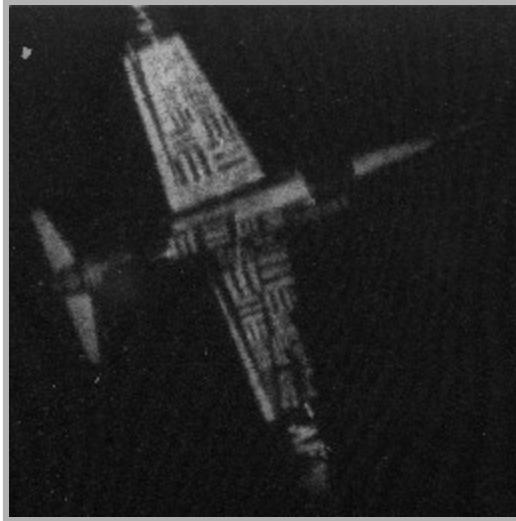
- $\gamma < 1$ to enhance contrast in dark regions
- $\gamma > 1$ to enhance contrast in bright regions.

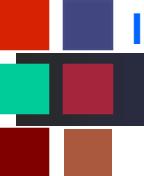




Introduction to Computer Vision

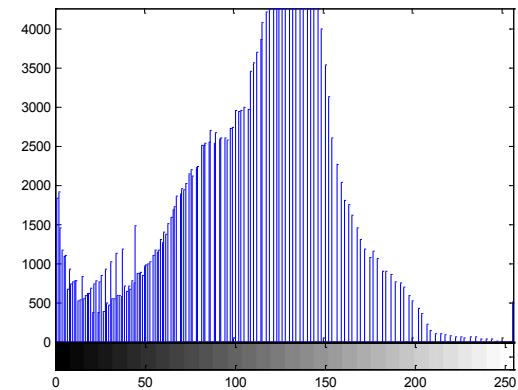
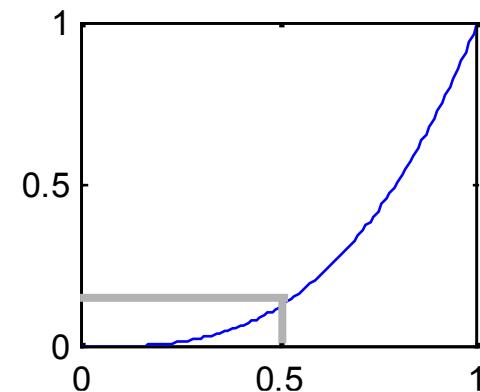
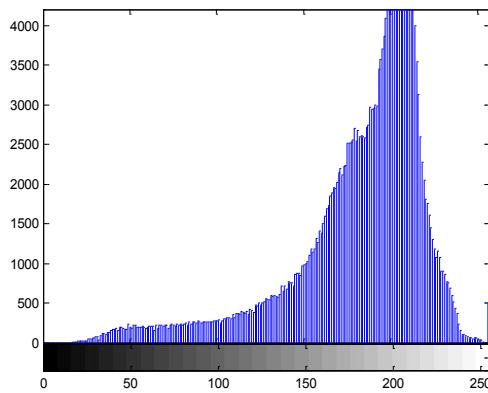
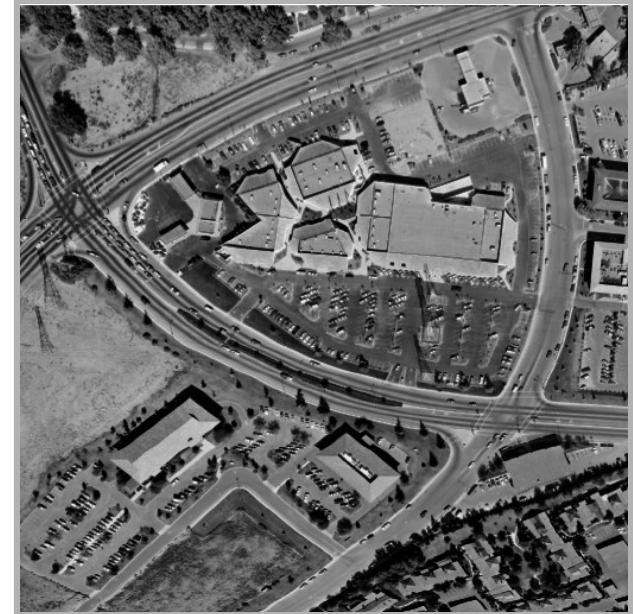
$\gamma=0.5$



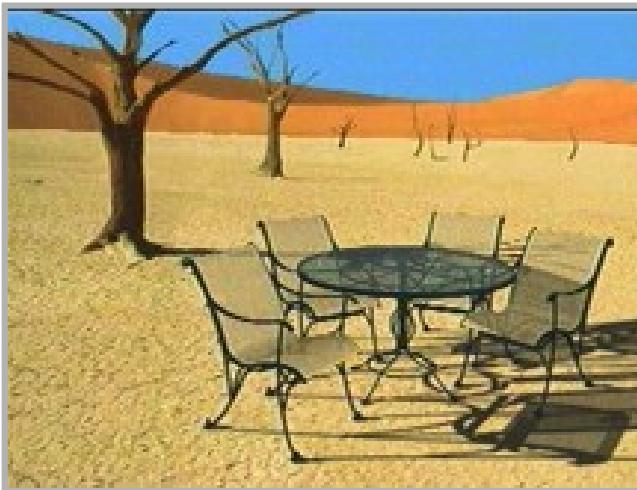


Introduction to Computer Vision

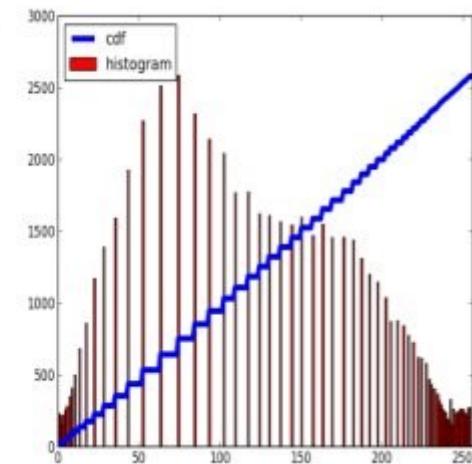
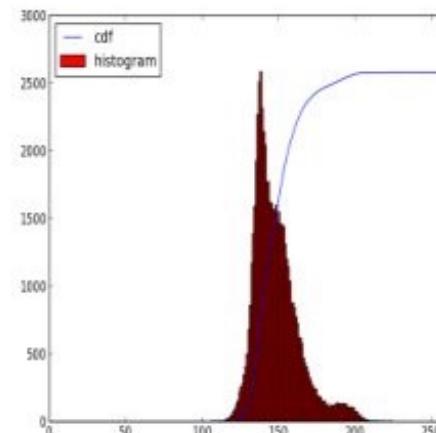
$\gamma=3.0$



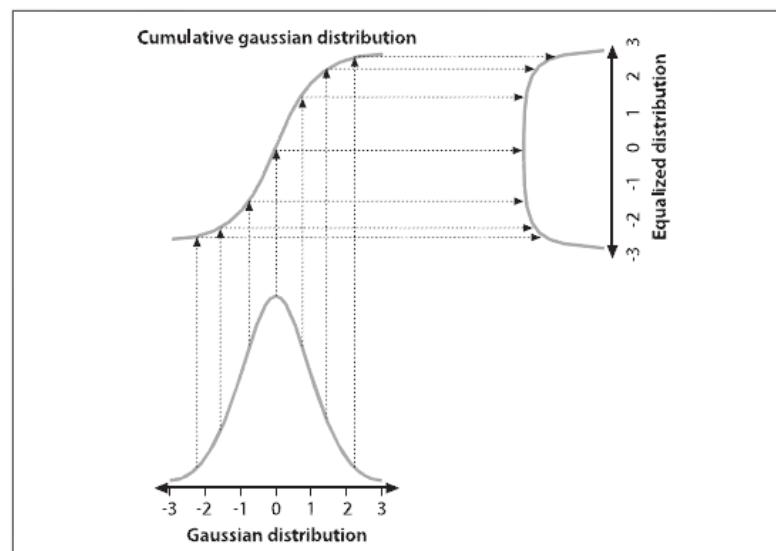
- Technique can be applied to color images
 - same curve to all color bands
 - different curves to separate color bands:



- but ... **be careful**, colors may become distorted...!
- Point transformations are usually applied using **LUT's (Look Up Tables)**



source: <http://docs.opencv.org/>



After equalization
the cumulative
distribution function
is almost linear

Using the cumulative distribution function
to equalize a Gaussian distribution



Original



Histogram
stretching



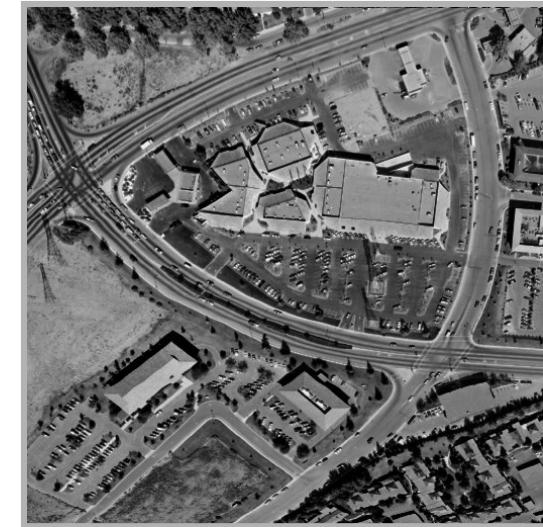
Histogram
equalization

- The contrast enhancement is better after the histogram equalization, which more easily detects structures located in the shade.
- In fact any strongly represented gray-level is stretched while any weakly represented graylevel is merged with other close levels

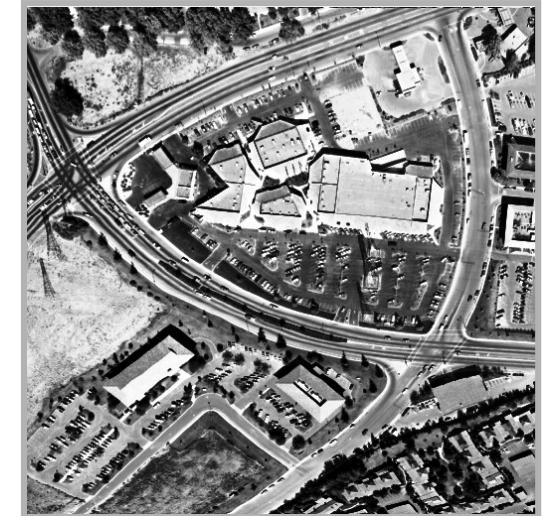
Original



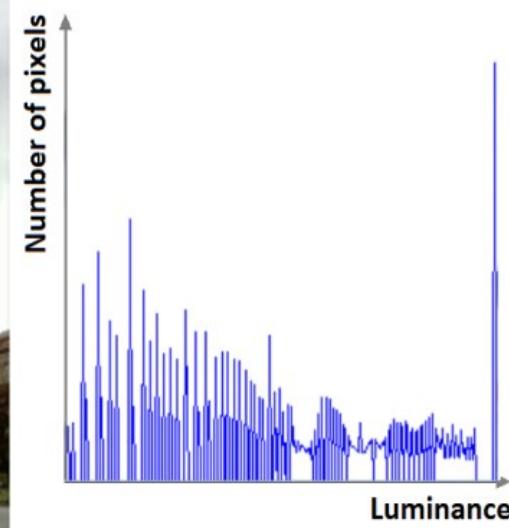
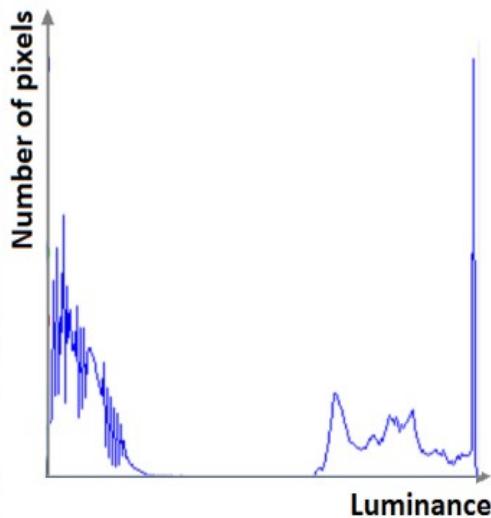
$\gamma > 1$



Histogram
equalization

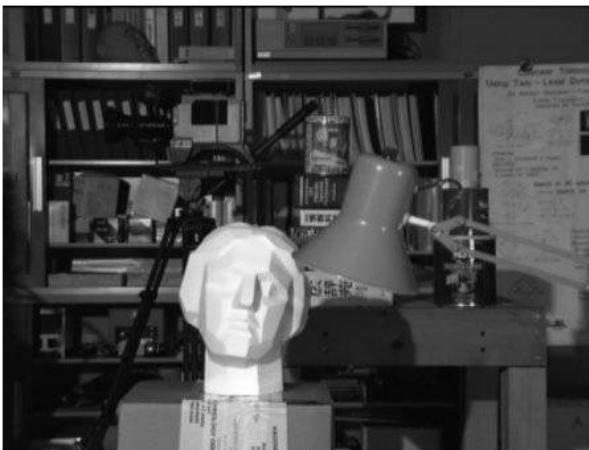


- In color images, only the luminance channel is usually equalized as otherwise the colors can become distorted.

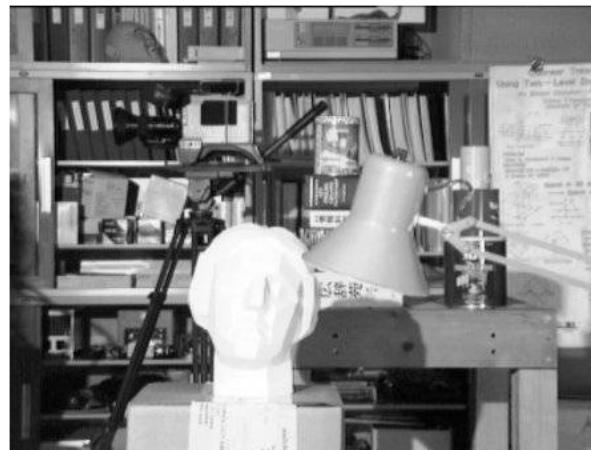


- Acquisition process degrades image
- Brightness and contrast enhancement implemented by pixel operations
- No one algorithm universally useful
- $\gamma > 1$ enhances contrast in bright images
- $\gamma < 1$ enhances contrast in dark images
- Transfer function for histogram equalization proportional to cumulative histogram
- It is essential a process of trial-and-error to determine whether a particular type of images will benefit from histogram transformation operations.

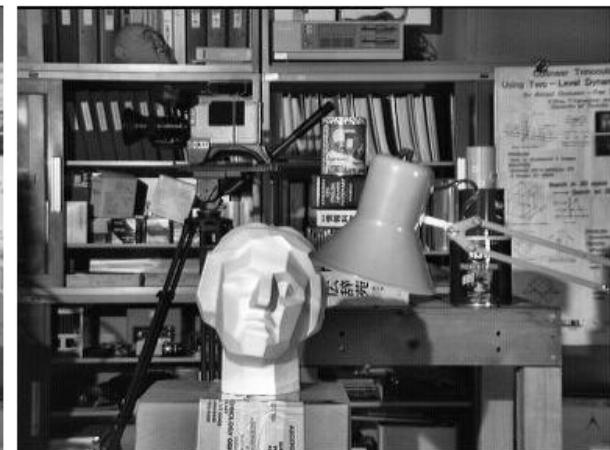
- **CLAHE – Contrast Limited Adaptive Histogram Equalization**
 - Considering the global contrast of the image is not a good idea, in some cases ... (look at the face of the statue, in the middle image)
 - For some images, it might be preferable to apply different kinds of equalization in different regions.
 - Instead of computing a single curve, the image is divided into $M \times M$ pixel non-overlapped sub-blocks and separate histogram equalization is performed in each sub-block.
 - To avoid blocking artifacts (i.e., intensity discontinuities at block boundaries) in the resulting image, the equalization functions are smoothly interpolated as we move between blocks.
 - This technique is known as adaptive histogram equalization (AHE) and its contrast limited (gain-limited, to avoid noise amplification) version is known as CLAHE.



Original image



After global equalization
(notice the effect on the face of the statue)



After CLAHE



Introduction to
Computer Vision

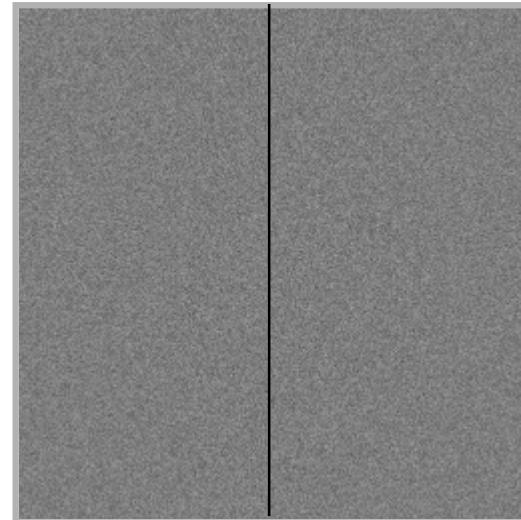
Filtering

- What is noise?
- How is noise reduction performed?
 - Noise reduction from first principles
 - Neighbourhood operators
 - linear filters (low pass)
 - non-linear filters (median)



image

+



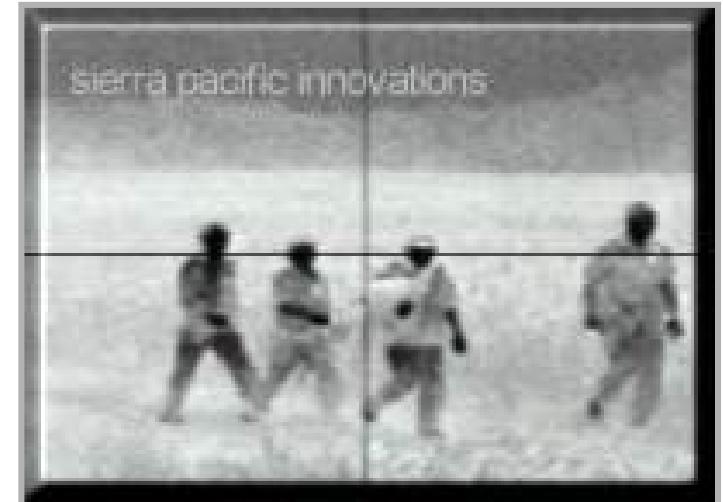
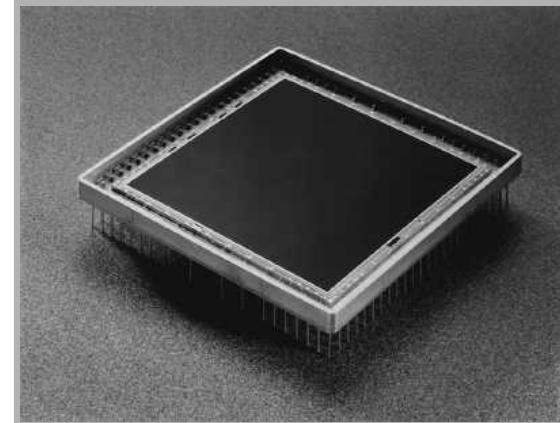
noise

=



‘grainy’ image

- Sources of noise = CCD chip.
- Electronic signal fluctuations in detector.
 - Caused by thermal energy.
 - Worse for infra-red sensors.
- Other electronics
- Transmission (analog)



Radiation from the long wavelength IR band is used in most infrared imaging applications



- Plot of image brightness.
- Noise models:

- additive

- $f(i,j) = g(i,j) + v(i,j)$
 - Gaussian
 - salt and pepper

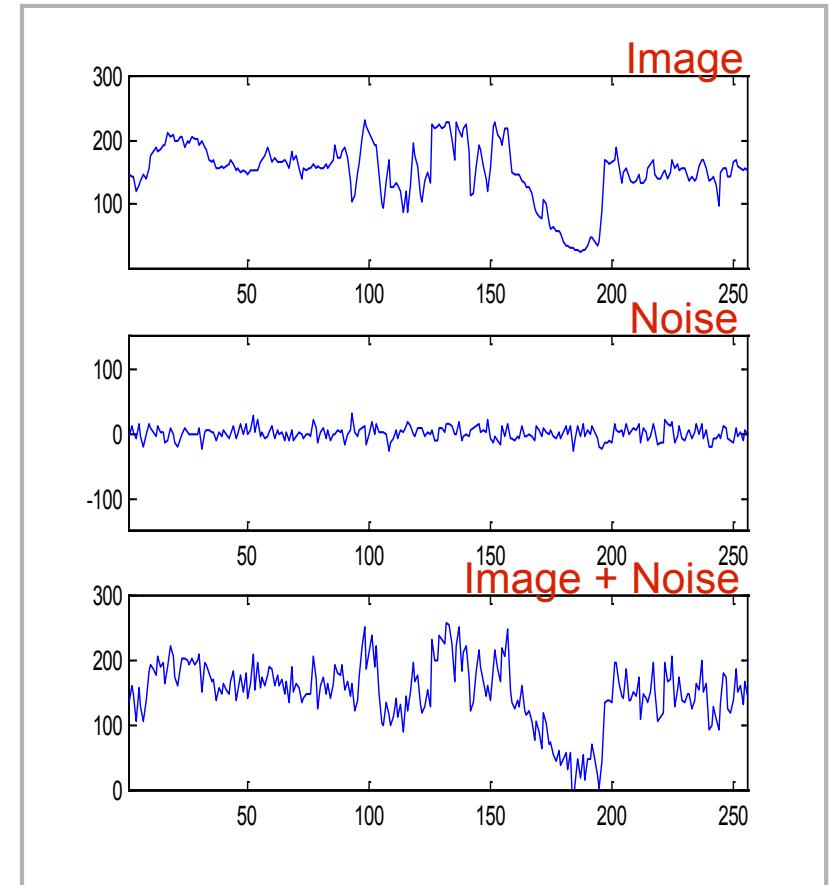
- multiplicative

- $f(i,j) = g(i,j) + g(i,j).v(i,j)$

where

- $f(i,j)$ - acquired signal (noisy)
- $g(i,j)$ - uncorrupted signal
- $v(i,j)$ – noise component

- Noise fluctuations are rapid
 - high frequency.



- Impulse noise
 - Noise is maximum or minimum values

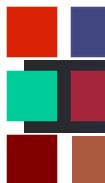


source: *A Practical Introduction to Computer Vision with OpenCV*, by Kenneth Dawson-Howe

- Good approximation to real noise
- Distribution is Gaussian (mean & standard deviation)

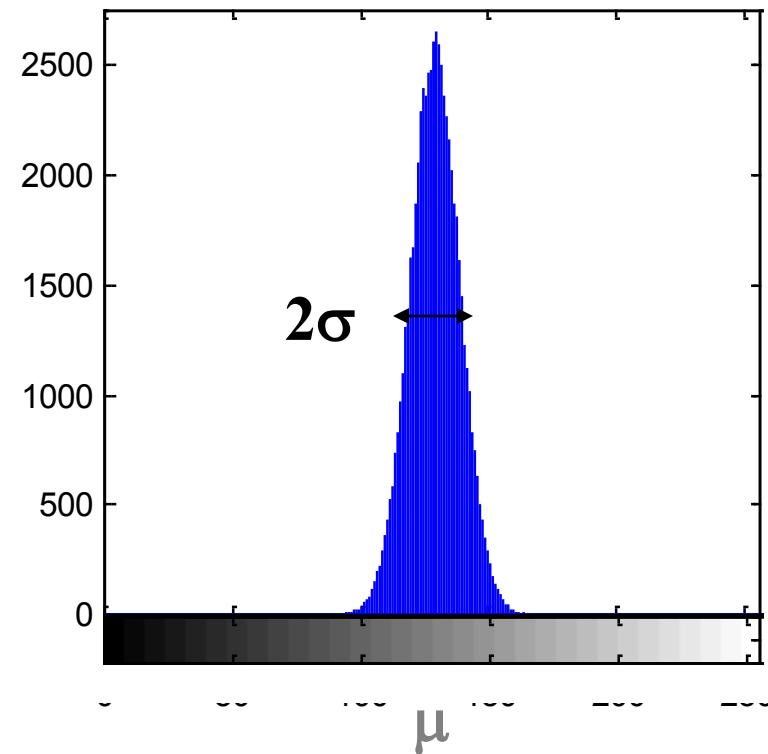


source: *A Practical Introduction to Computer Vision with OpenCV*, by Kenneth Dawson-Howe



- Plot noise histogram
- Typical noise distribution is normal or Gaussian
- Mean(noise) $\mu = 0$
- Standard deviation σ

$$\eta(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{2} \frac{(x-\mu)^2}{\sigma^2}\right)$$



- Noise varies above and below uncorrupted image.

Image

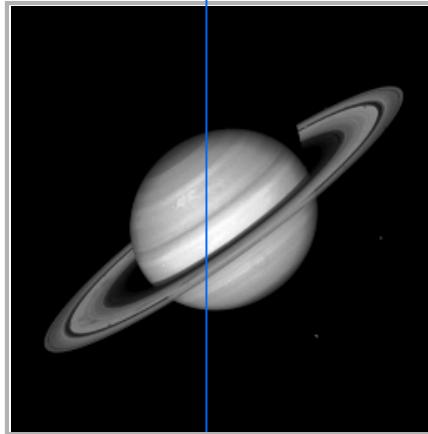
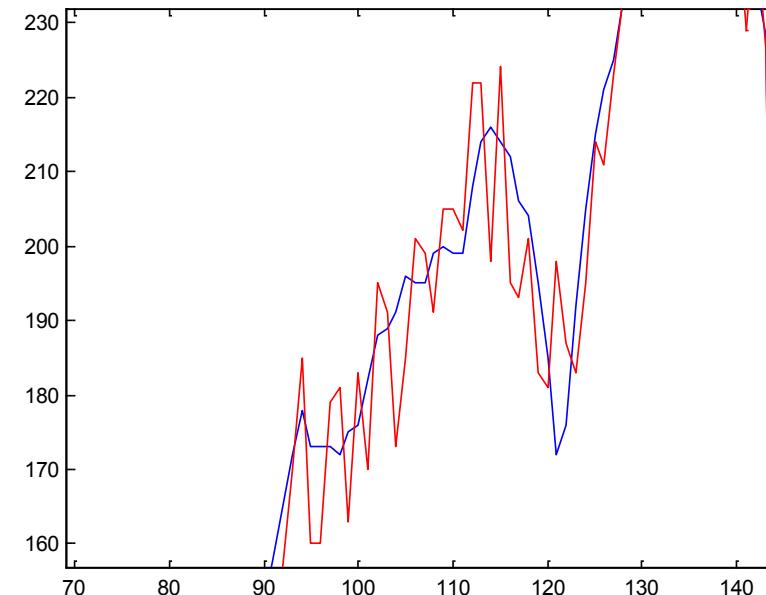
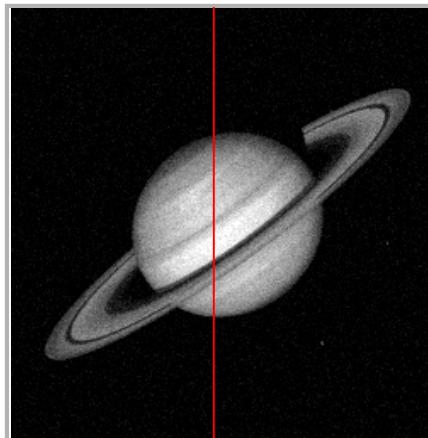


Image
+
Noise



- Removing (?) or reducing noise...
- Linear smoothing transformations
 - frame averaging (=> acquire several frames of static scene)
 - local averaging
 - Gaussian smoothing
- Non-linear transformations
 - median filter
 - rotating mask

Image Data:

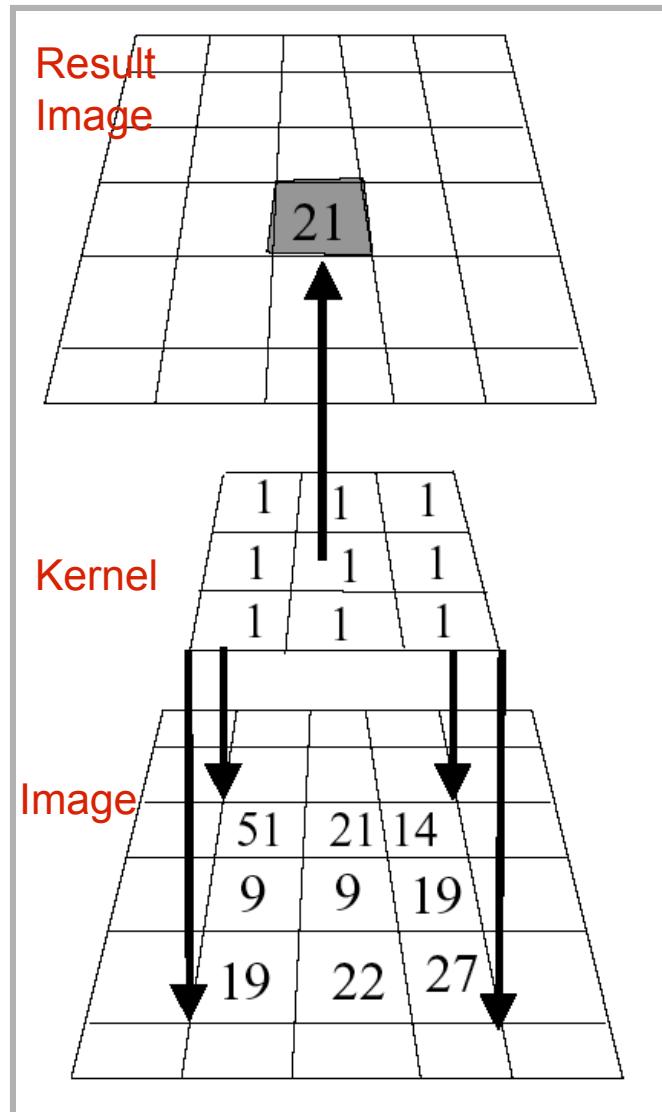
10	12	40	16	19	10
14	22	52	10	55	41
10	14	51	21	14	10
32	22	9	9	19	14
41	18	9	22	27	11
10	7	8	8	4	5

Mask / Filter / Kernel:

1	1	1	1
0	1	1	1
-1	1	1	1
-1	0	1	

$$F \circ I(x, y) = \sum_{j=-N}^N \sum_{i=-N}^N F(i, j) I(x+i, y+j)$$

Filter Image



- Kernel is aligned with pixel in image, multiplicative sum is computed, normalized,
- and stored in result image.
- Process is repeated across image.
- What happens when kernel is near edge of input image?

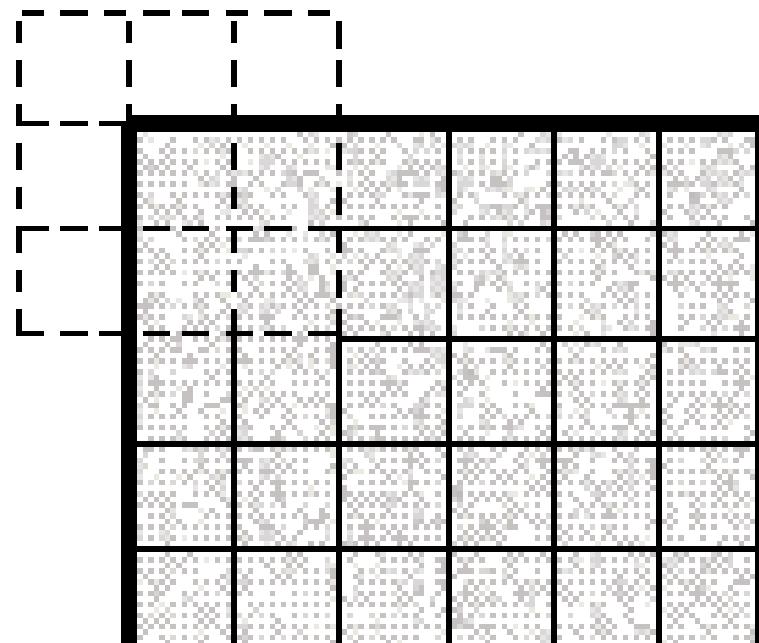
$$\begin{bmatrix}
 1*51 + 1*21 + 1*14 + \\
 1*8 + 1*9 + 1*19 + \\
 1*19 + 1*22 + 1*27
 \end{bmatrix} \quad 1/9 = 21$$

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

← note the factor

- missing samples are zero
- missing samples are gray
- copying last lines
- reflected indexing (mirror)
- circular indexing (periodic)
- reduce size of resulting image

• OpenCV
allows some control on how to do this;
see `CopyMakeBorder()`: copies the source
2D array into the interior of the destination
array and makes a border of the specified
type around the copied area



8	3	4	5
7	6	4	5
4	5	7	8
6	5	5	6

I

8	8	3	4	5	5
8	8	3	4	5	5
7	7	6	4	5	5
4	4	5	7	8	8
6	6	5	5	6	6
6	6	5	5	6	6

I with padded boundaries

6.44	5.22	4.33	4.67
5.78	5.33	5.22	5.67
5.56	5.44	5.67	6.00
5.22	5.33	5.78	6.33

J = FoI

- A **convolution** operation is a correlation where the filter is flipped both horizontally and vertically before being applied to the image:

$$F * I(x, y) = \sum_{j=-N}^N \sum_{i=-N}^N F(i, j)I(x-i, y-j)$$

- Suppose F is a Gaussian or mean kernel.
How does convolution differ from cross-correlation?

- Extremely important concept in computer vision, image processing, signal processing, etc.
- Lots of related mathematics (**we won't do**)
- General idea: reduce a filtering operation to the repeated application of a mask (or filter kernel) to the image
 - Kernel can be thought of as an NxN image
 - N is usually odd so kernel has a central pixel
- In practice
 - (flip kernel)
 - Align kernel center pixel with an image pixel
 - Pointwise multiply each kernel pixel value with corresponding image pixel value and add results
 - Resulting sum is normalized by kernel weight
 - Result is the value of the pixel centered on the kernel

Correlation

		Padded f						
Origin f		0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	2	3	0
0	0	0	0	0	4	5	6	0
0	0	0	0	0	7	8	9	0

(a)

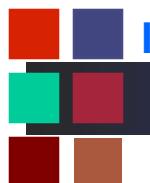
(b)

Initial position for w	Correlation result	Full correlation result
1 2 3	0 0 0 0 0	0 0 0 0 0 0 0 0
4 5 6	0 0 0 0 0	0 0 0 0 0 0 0 0
7 8 9	0 9 8 7 0	0 0 9 8 7 0 0 0
0 0 0 1	0 6 5 4 0	0 0 6 5 4 0 0 0
0 0 0 0	0 3 2 1 0	0 0 3 2 1 0 0 0
0 0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0
0 0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0

(c)

(d)

(e)



Convolution

$$w$$
$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

Rotated w

$$\begin{matrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \quad \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

(f)

Convolution result

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

(g)

Full convolution result

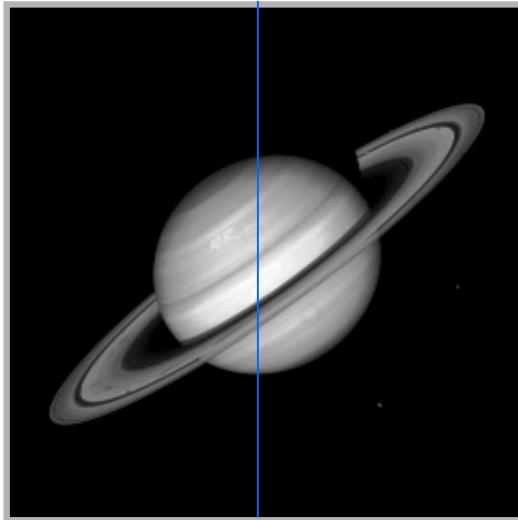
$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 3 & 0 & 0 \\ 0 & 0 & 4 & 5 & 6 & 0 & 0 \\ 0 & 0 & 7 & 8 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

(h)

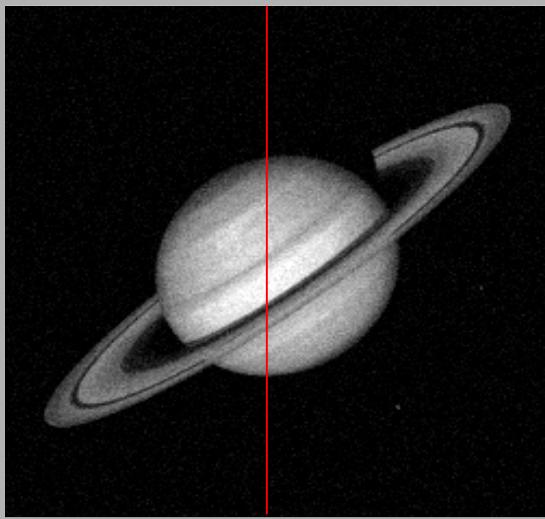
source: <https://towardsdatascience.com/convolution-vs-correlation-af868b6b4fb5>

Notes:

- 1) Correlation and convolution are identical when the filter is symmetric.
- 2) The key difference between the two is that convolution is associative, that is, if F and G are filters, then $F^*(G^*Img) = (F^*G)^*Img$
- 3) In general, people use **convolution** for image processing operations such as smoothing, and they use **correlation** to match a template to an image (**see later**).



Uncorrupted Image



Uncorrupted Image + Noise

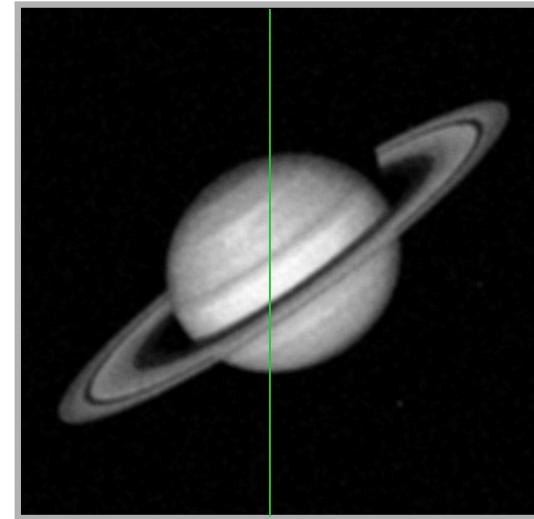
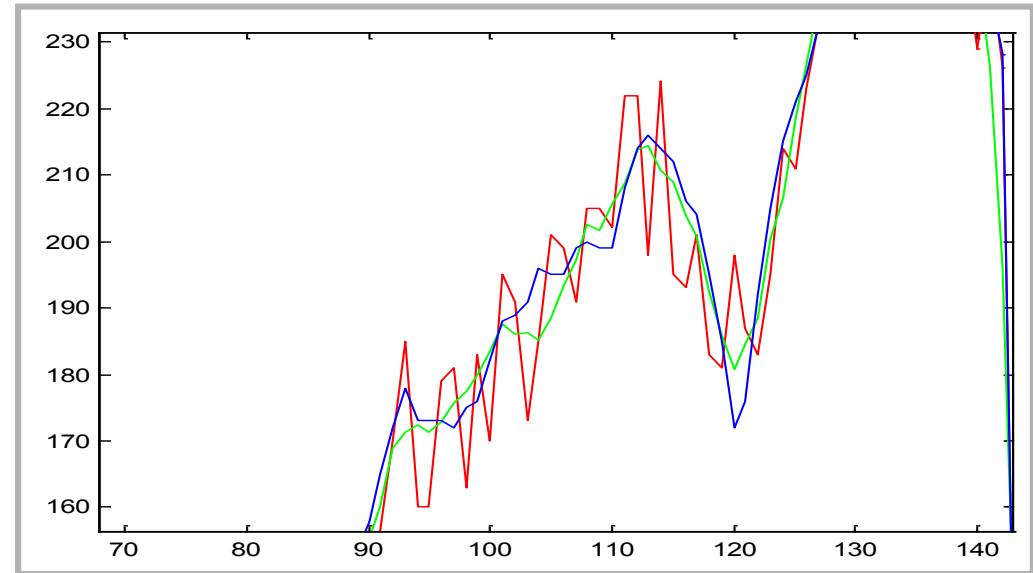


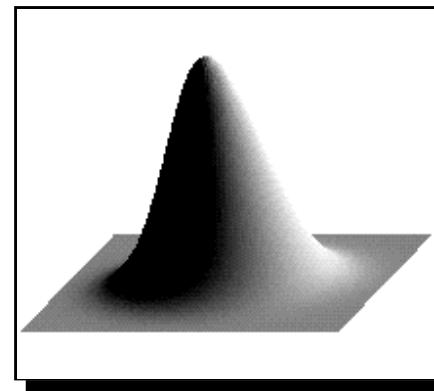
Image + Noise - Blurred

- Technique relies on high frequency noise fluctuations being ‘blocked’ by filter.
Hence, **low-pass** filter.
- Fine detail in image may also be smoothed.
- Balance between
keeping image fine detail and reducing noise.

- Saturn image (*previous slide*) has coarse detail
- Boat image contains fine detail
- Noise reduced but fine detail also smoothed



- Smoothing operator should be
 - 'tunable' in what it leaves behind
 - smooth and localized in image space.
- One operator which satisfies these two constraints is the Gaussian

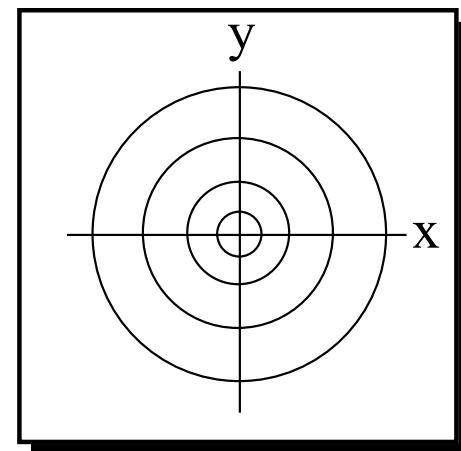


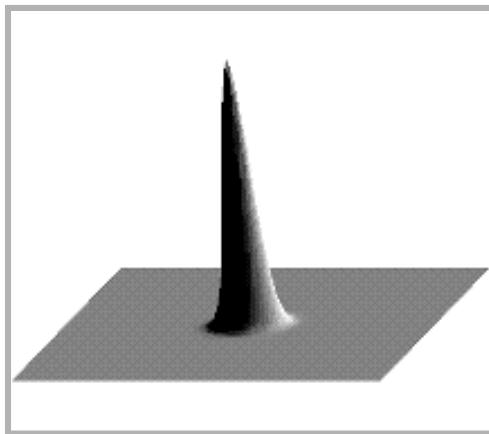
- *OpenCV: Smooth()*

- The two-dimensional Gaussian distribution is defined by:

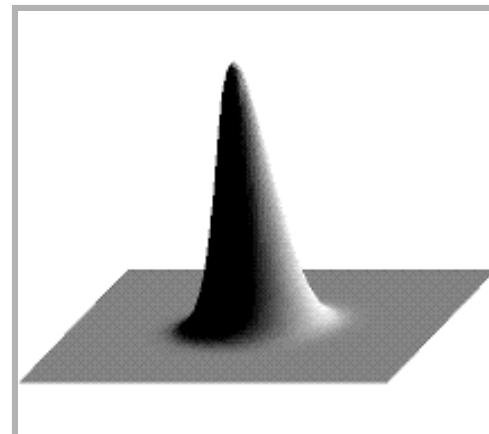
$$G(x,y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left[\frac{(x^2+y^2)}{2\sigma^2}\right]}$$

- From this distribution, can generate smoothing masks whose width depends upon σ :

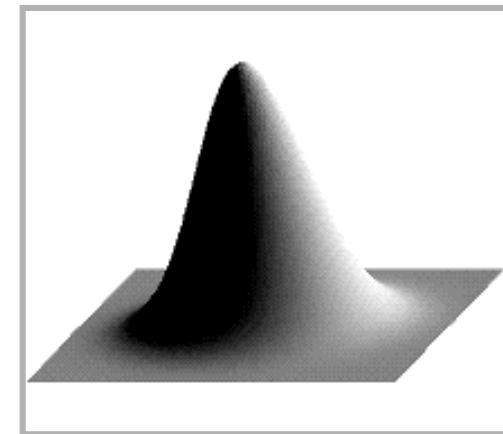




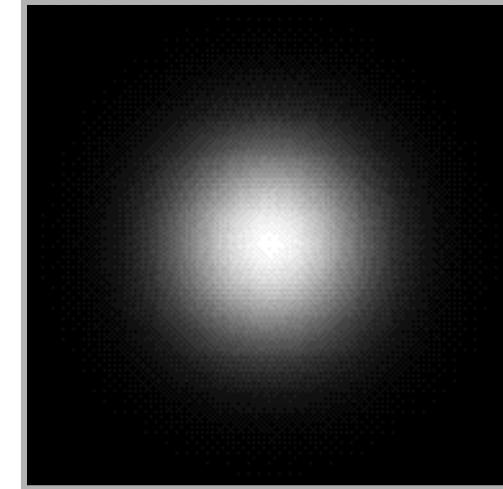
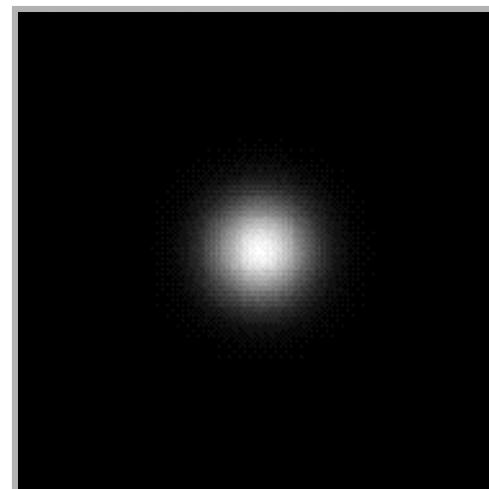
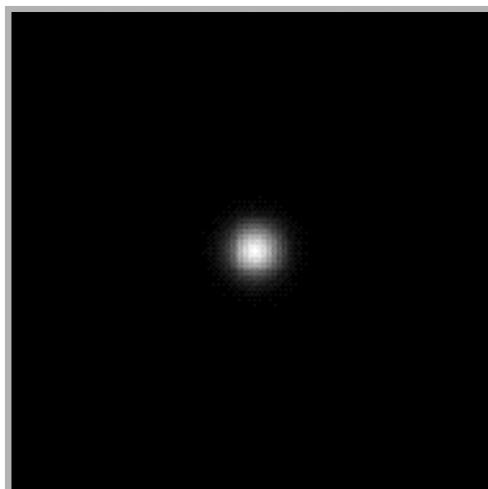
$\sigma^2 = .25$

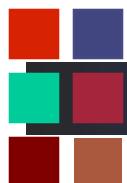


$\sigma^2 = 1.0$



$\sigma^2 = 4.0$





- Choose $\sigma^2 = 2$ and $n = 7$, then:

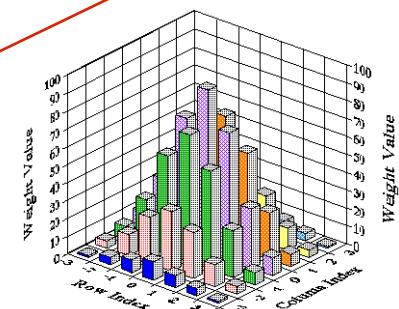
		j						
		-3	-2	-1	0	1	2	3
i	-3	.011	.039	.082	.105	.082	.039	.011
	-2	.039	.135	.287	.368	.287	.135	.039
	-1	.082	.287	.606	.779	.606	.287	.082
	0	.105	.039	.779	1.000	.779	.368	.105
	1	.082	.287	.606	.779	.606	.287	.082
	2	.039	.135	.287	.368	.287	.135	.039
	3	.011	.039	.082	.105	.082	.039	.011

$$\frac{W(1,2)}{k} = \exp\left(-\frac{1^2 + 2^2}{2*2}\right)$$

To make this value 1,
choose k=91

1	4	7	10	7	4	1
4	12	26	33	26	12	4
7	26	55	71	55	26	7
10	33	71	91	71	33	10
7	26	55	71	55	26	7
4	12	26	33	26	12	4
1	4	7	10	7	4	1

7x7 Gaussian filter





7x7 Gaussian kernel



15x15 Gaussian kernel

- Gaussian is not the only choice, but it has a number of important properties
 - If we convolve a Gaussian with another Gaussian, the result is a Gaussian
 - This is called linear scale space

$$G_{\sigma_1} * G_{\sigma_2} = G_{\sqrt{\sigma_1^2 + \sigma_2^2}}.$$

- Efficiency: separable

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2 + y^2)}{2\sigma^2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)\right) \times \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right)\right), \end{aligned}$$

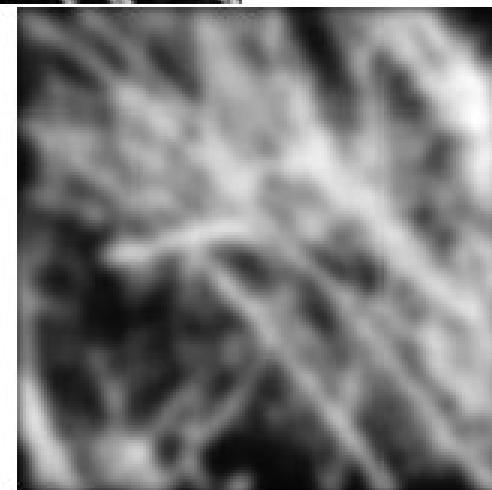
Mean vs. Gaussian filtering



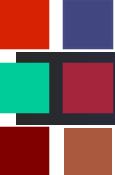
Original image



After mean filtering



After Gaussian filtering



Introduction to Computer Vision

hmmmmm.....

Image



Blurred image



=



- Compute median of points in neighborhood
- Nonlinear filter
 - example of a larger class of filters named "rank-filters"
(ex: min, median, max)
- Has a tendency to blur detail less than averaging
- Works very well for 'shot' or 'salt and pepper' noise

Linear vs Non-linear

Mean(I_1+I_2) ==
Mean (I_1) + Mean(I_2)

Median(I_1+I_2) !=
Median(I_1)+Median(I_2)



Original



Low-pass



Median

Linear vs Non-linear filters:

<https://dsp.stackexchange.com/questions/14241/what-is-the-difference-between-linear-and-non-linear-filters>

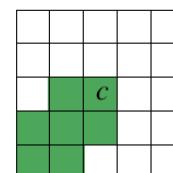
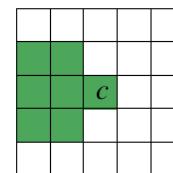
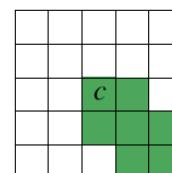
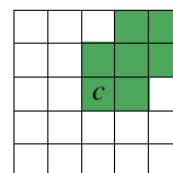
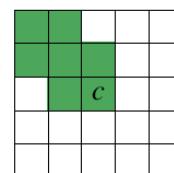
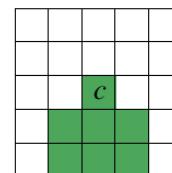
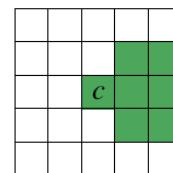
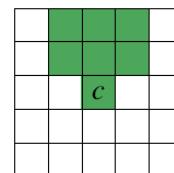
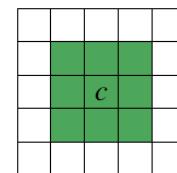
- Mean
 - Linear
 - Signal frequencies shared with noise are lost, resulting in blurring.
 - Impulsive noise is diffused but not removed.
 - It spreads the noise, resulting in blurring.
 - Blurs edges.
- Median
 - Non-linear
 - Does not spread the noise.
 - Can remove spike noise.
 - Preserves (some) edges.
 - Small details (small regions, thin edges, ...) may be lost.
 - Expensive to run.

Smooth (average, blur) an image without disturbing

- sharpness or
- position
- of the edges
- Nagao-Matsuyama filter
- Kuwahara filter
- Anisotropic diffusion filters (Perona & Malik,)
- Bilateral filtering
- ...



- Calculate the variance within nine subwindows of a 5x5 moving window
- Output value is the **mean** of the **subwindow with the smallest variance**
- Nine subwindows used:





- Principle:
 - divide filter mask into four regions (a, b, c, d).
 - in each compute the mean brightness and the variance
 - the output value of the center pixel (abcd) in the window is the **mean value of that region** that has the **smallest variance**.



a	a	ab	b	b
a	a	ab	b	b
ac	ac	abcd	bd	bd
c	c	cd	d	d
c	c	cd	d	d



Kuwahara filter example



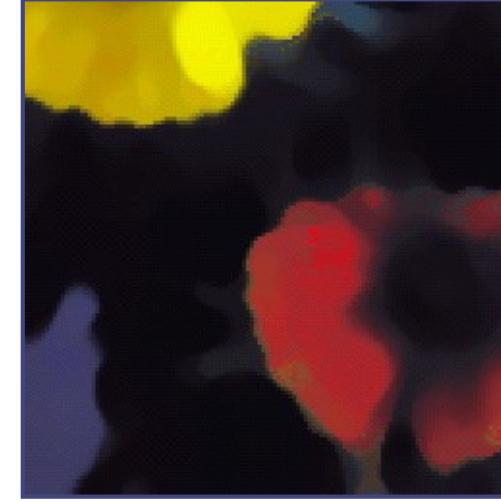
Original



Median (1 iteration)



Kuwahara



Median (10 iterations)

- The anisotropic diffusion is a nonlinear filter that smoothes the intraregions of an image without blurring the strong edges
 - Encourages the **smoothing in homogeneous regions in preference to** smoothing across the boundaries.
- Based on the solution of a partial differential equation, inspired in heat diffusion equation
 - *The algorithm results from the discretization of the non-linear diffusion equation.*
 - *The derivatives are approximated by differences.*
- Diffusion methods average over extended regions by solving partial differential equations, and are therefore inherently iterative.
Iteration may raise issues of stability and, depending on the computational architecture, efficiency.
- This algorithm is also used to detect the edges

Anisotropic diffusion filtering

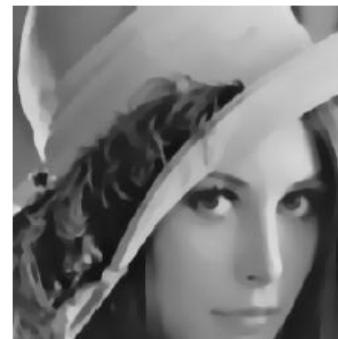


Original



Perona, Malik, 1987

$$\frac{\partial u}{\partial t} = \operatorname{div} \left(\frac{Du}{|Du|^2 + \lambda^2} \right)$$



Rudin, Osher,
Fatemi, 1992

$$\frac{\partial u}{\partial t} = \operatorname{div} \left(\frac{Du}{|Du|} \right)$$



Alvarez, Lions, 1992

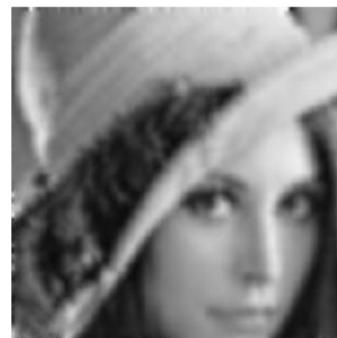
$$\frac{\partial u}{\partial t} = \frac{|Du|}{|k * Du|} \operatorname{div} \left(\frac{Du}{|Du|} \right)$$



Weickert, 1994

$$\frac{\partial u}{\partial t} = D^2 u(d, d)$$

$$d = SEigen(k * (Du \otimes Du))$$



Caselles, Sbert, 1997

$$\frac{\partial u}{\partial t} = \frac{1}{|Du|^2} D^2 u(Du, Du)$$



Zhong Carmona, 1998

$$\frac{\partial u}{\partial t} = D^2 u(d, d)$$

$$d = SEigen(D^2 u)$$

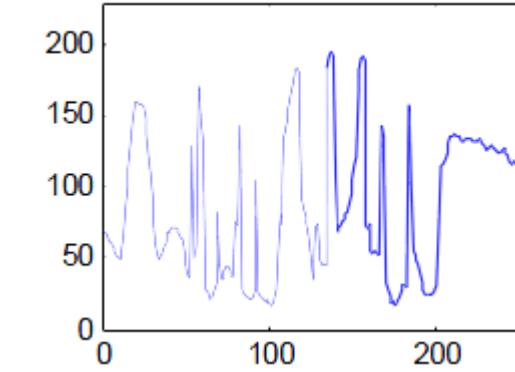
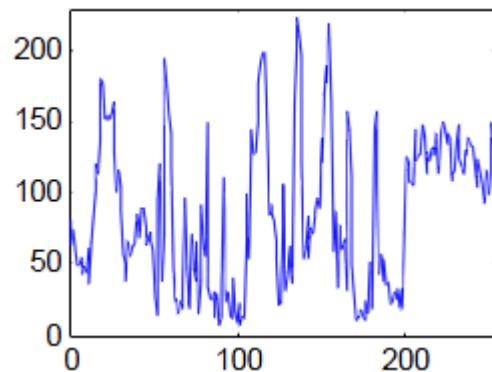
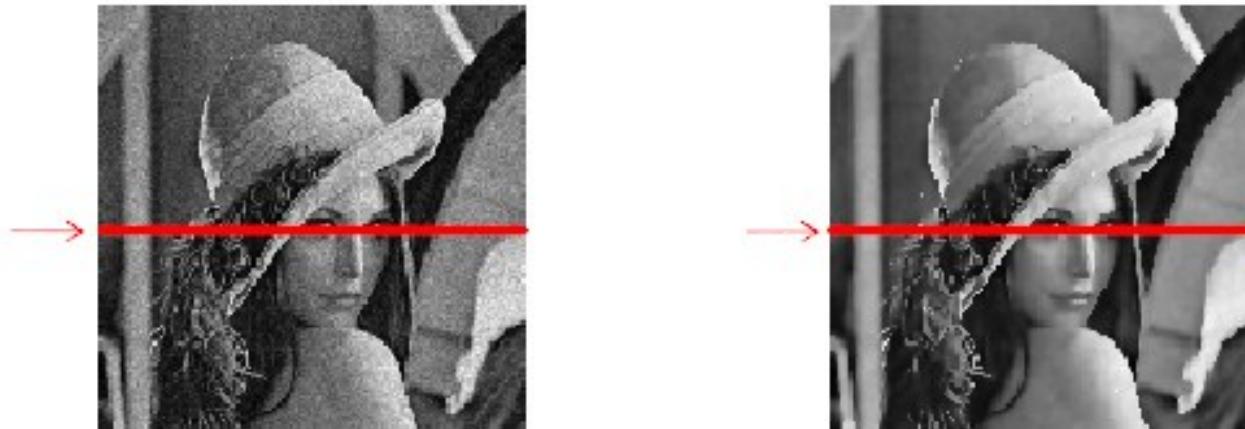


Sochen, Kimmel,
Malladi, 1998

$$\frac{\partial u}{\partial t} = \operatorname{div} \left(\frac{Du}{\sqrt{|Du|^2 + 1}} \right)$$

Note: Original Perona-Malik diffusion process is NOT anisotropic, although they erroneously said it was.

Anisotropic diffusion filtering



- A bilateral filter is an edge-preserving and noise reducing smoothing filter.
- Traditional filtering is **domain filtering**, and enforces closeness by weighing pixel values with weights that fall off with distance.
- Similarly, we define **range filtering**, which averages image values with weights that decay with dissimilarity.
- **Range filters**
 - are nonlinear because their weights depend on image intensity or color
 - preserve edges
 - by themselves, may distort an image's color map
(see C.Tomasi et al., Bilateral Filtering for Gray and Color, ICCV 1998)
- **Bilateral filtering** combines range and domain filtering.
- The intensity value at each pixel in an image is replaced by a **weighted average of intensity values from nearby pixels**.
- This weight is based on a Gaussian distribution.
The weights depend not only on Euclidean distance but also on the radiometric differences (e.g. color intensity).
- It preserves sharp edges by systematically looping through each pixel and attributing weights to the adjacent pixels accordingly.
- *OpenCV: Smooth()*

original



original



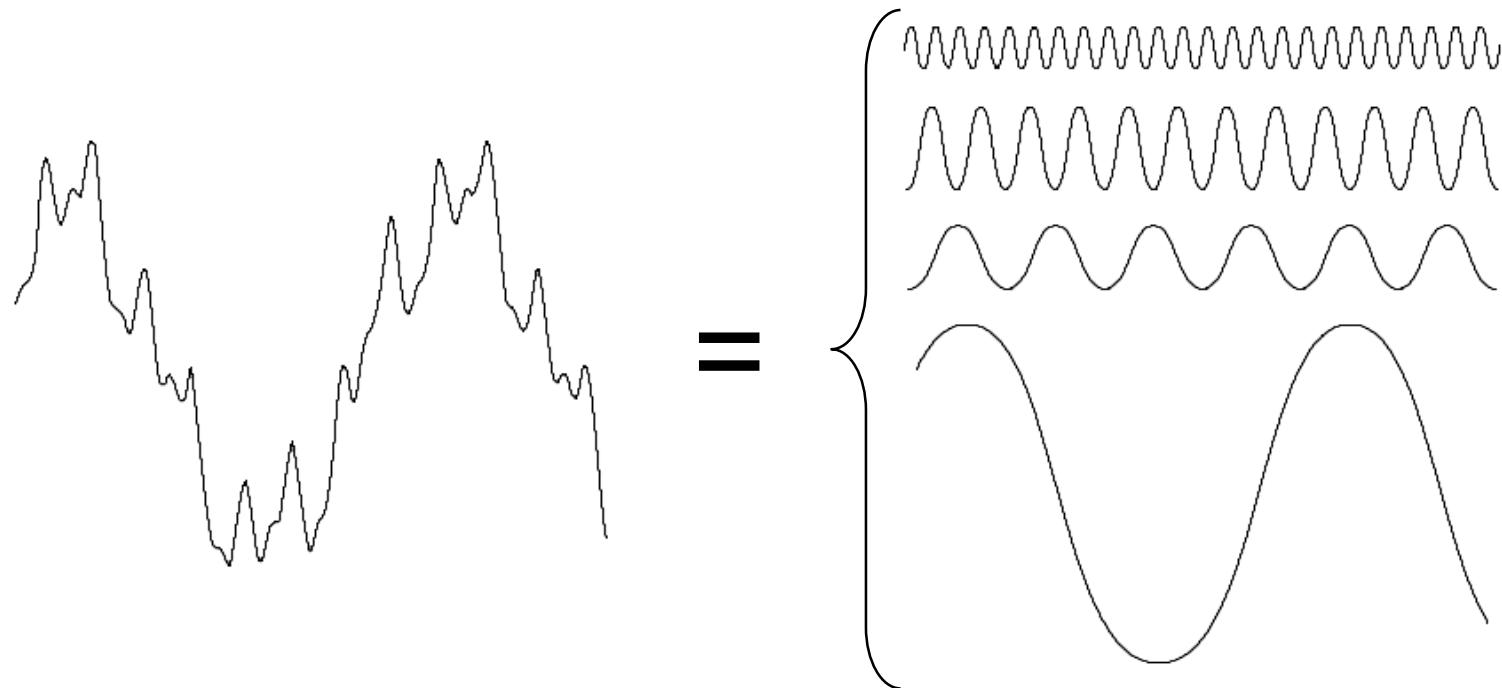
1 iteration



5 iterations



- An image can be represented in the frequency domain, using the Fourier Transform (FT).
- The FT encodes the amplitude and phase of each frequency component.
- The values near the origin of the transformed space are called low-frequency components of the FT, and those distant from the origin are the high-frequency components.
- Convolution in the image domain corresponds to multiplication in the spatial frequency domain.
- Therefore, convolution with large filters, which would normally be expensive in the image domain, can be implemented efficiently using the Fast Fourier Transform (FFT).
- This is an important technique in many image processing applications.



- Any function that periodically repeats itself can be expressed as a sum of sines and cosines of different frequencies each multiplied by a different coefficient – a *Fourier series*

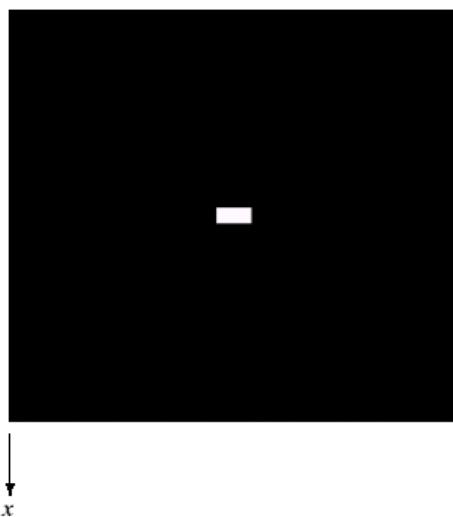
The *Discrete Fourier Transform* of $f(x, y)$,
for $x = 0, 1, 2 \dots M-1$ and $y = 0, 1, 2 \dots N-1$,
denoted by $F(u, v)$, is given by the equation:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)}$$

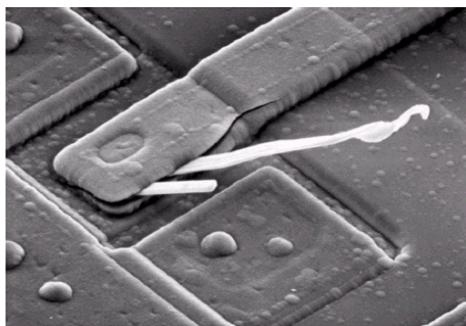
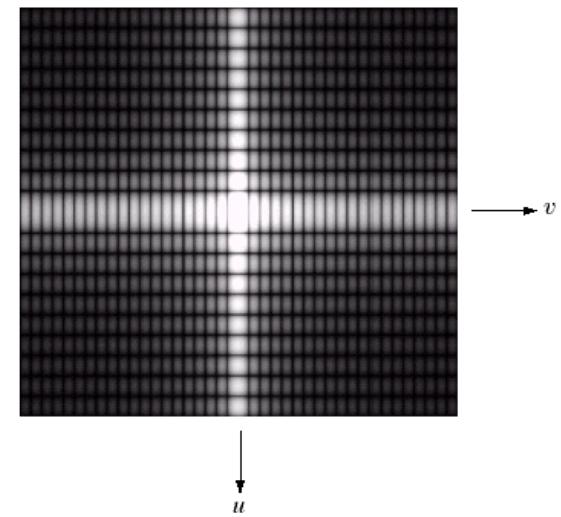
for $u = 0, 1, 2 \dots M-1$ and $v = 0, 1, 2 \dots N-1$.



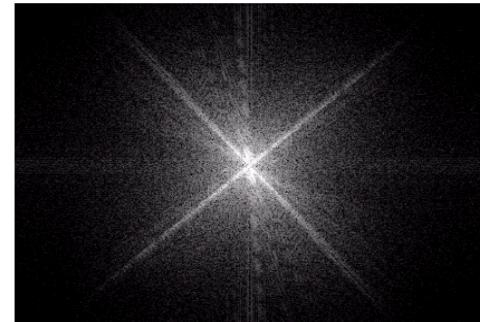
- The DFT of a 2D image can be visualised by showing the spectrum of the images component frequencies



DFT



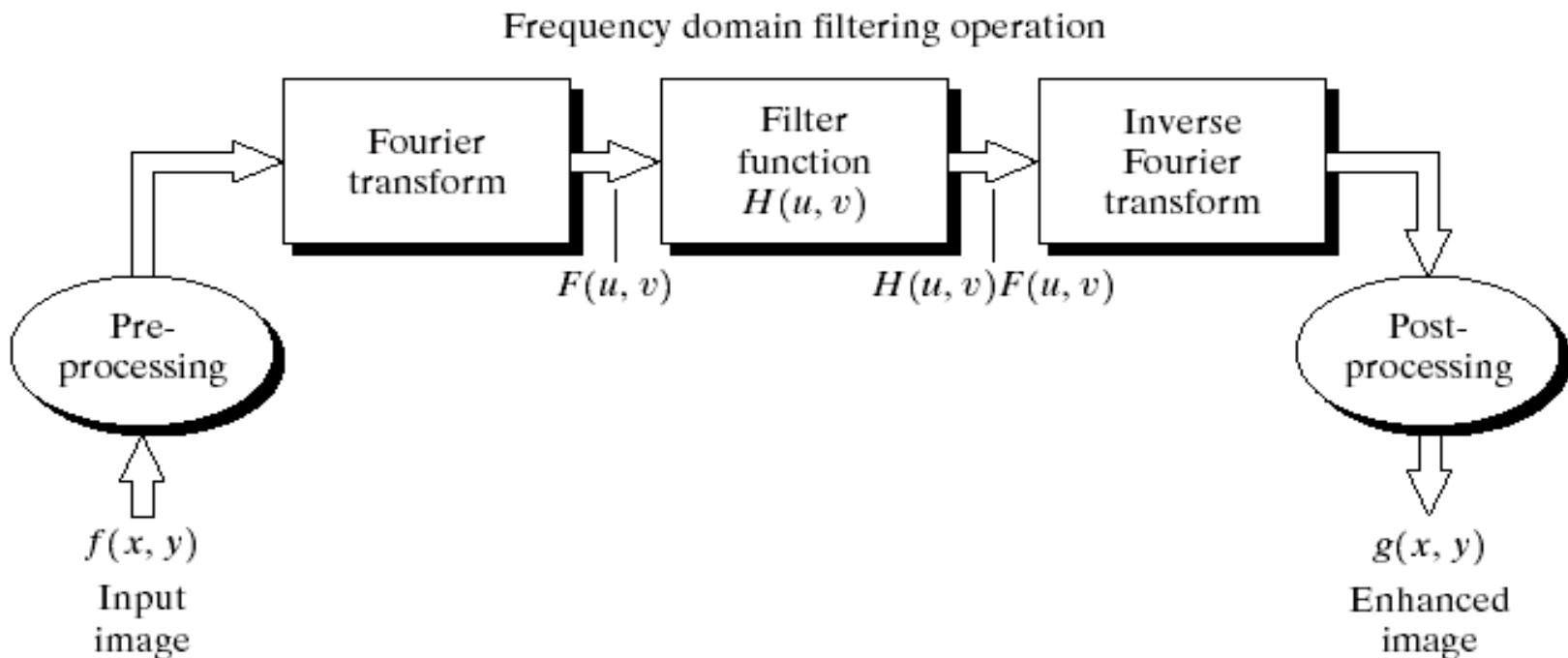
DFT

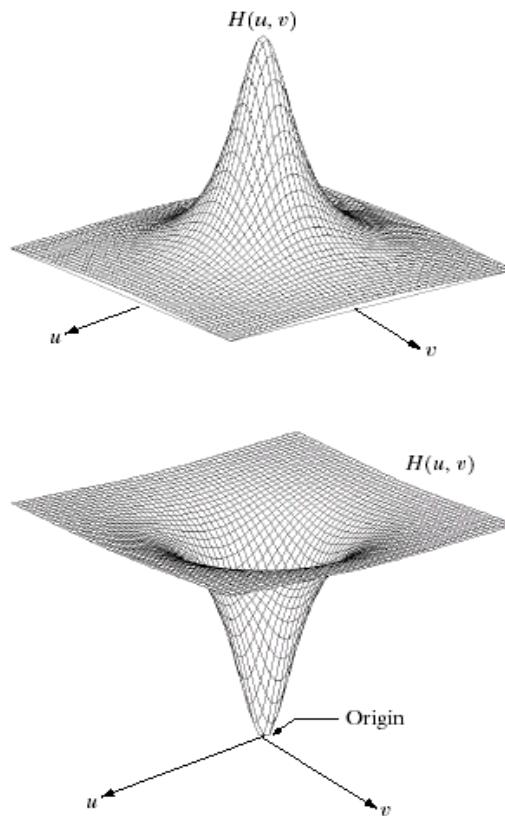
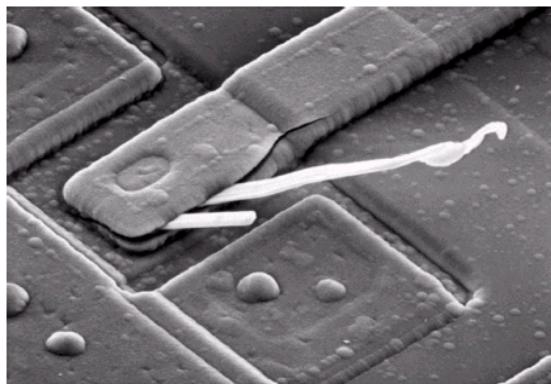


Fourier spectrum of the image

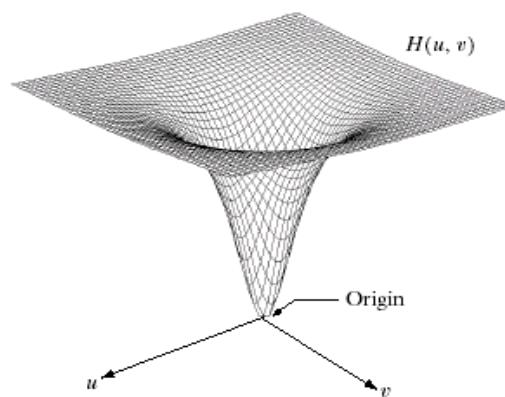
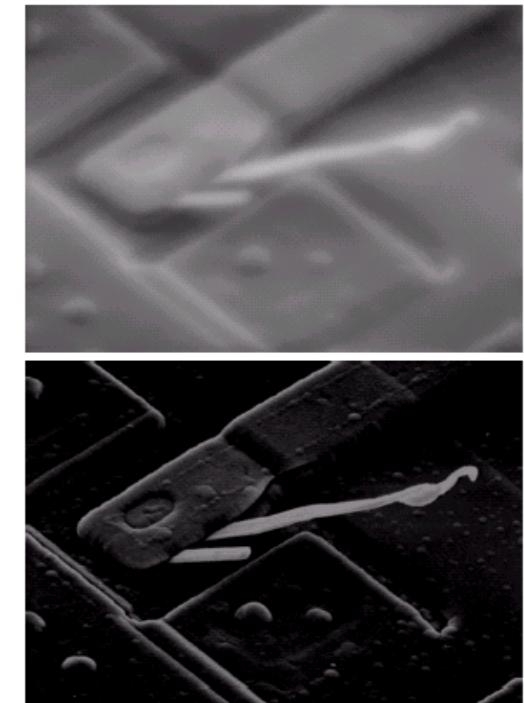
To filter an image in the frequency domain:

1. Compute $F(u, v)$ the DFT of the image
2. Multiply $F(u, v)$ by a filter function $H(u, v)$
3. Compute the inverse DFT of the result

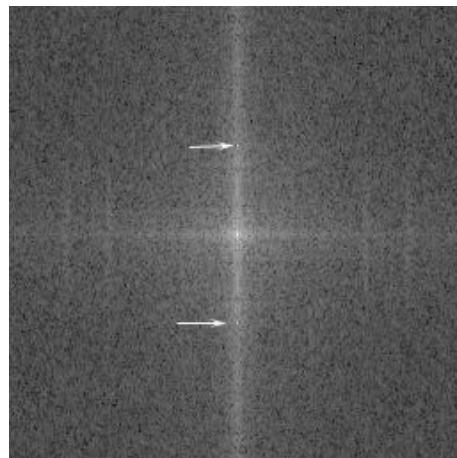
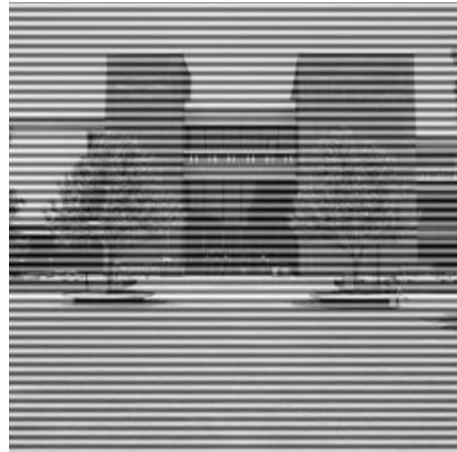




Low Pass Filter



High Pass Filter



2D Fourier transform



After filtering