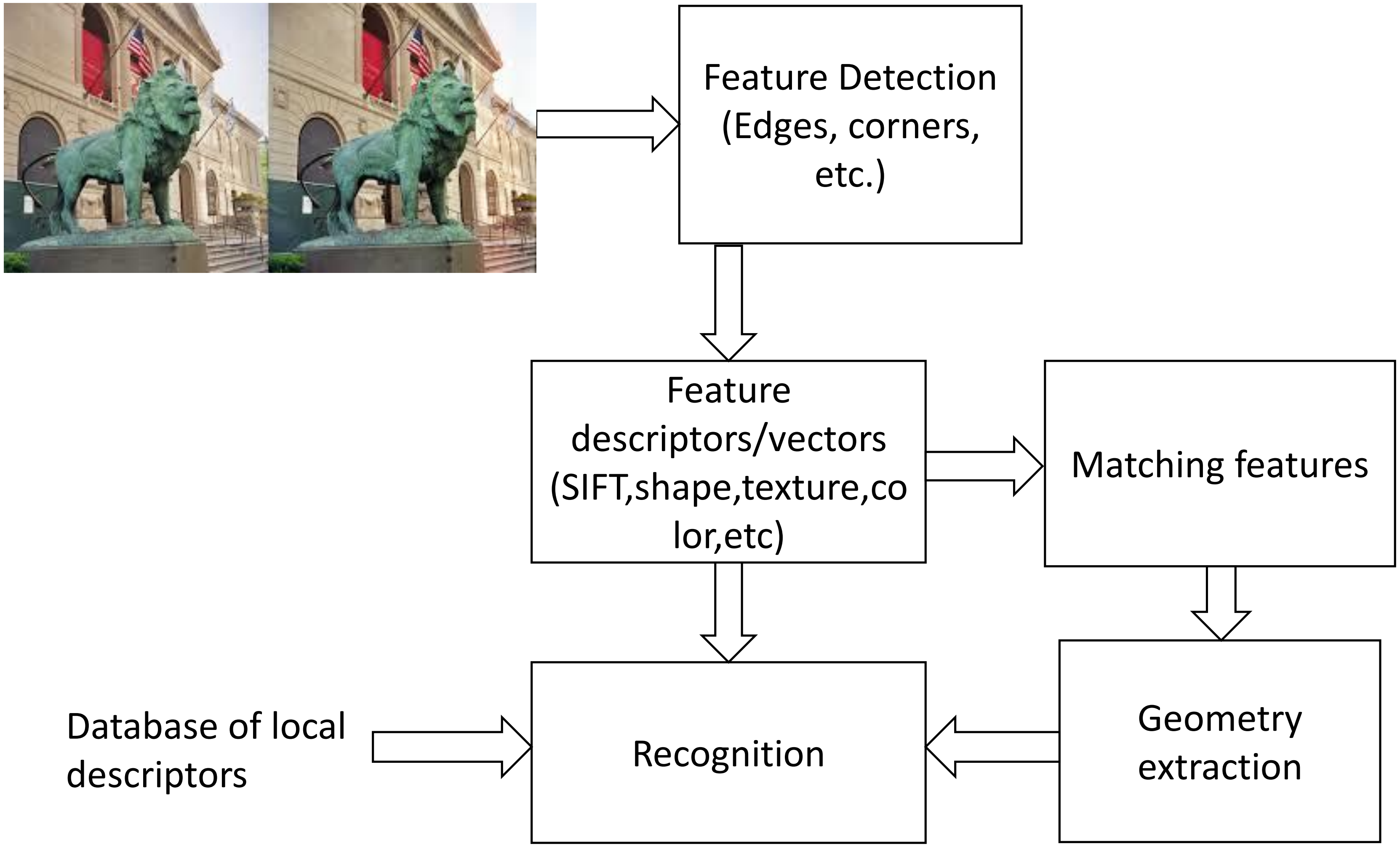
Low level image processing

Topics outline

- Image processing
    - Restoration
    - Enhancement
- Point operations
- Neighborhood operations/filtering

The big picture


Feature Detection (Edges, corners, etc.)
Feature descriptors/vectors (SIFT,shape,texture,color,etc)
Matching features
Geometry extraction
Recognition
Database of local descriptors

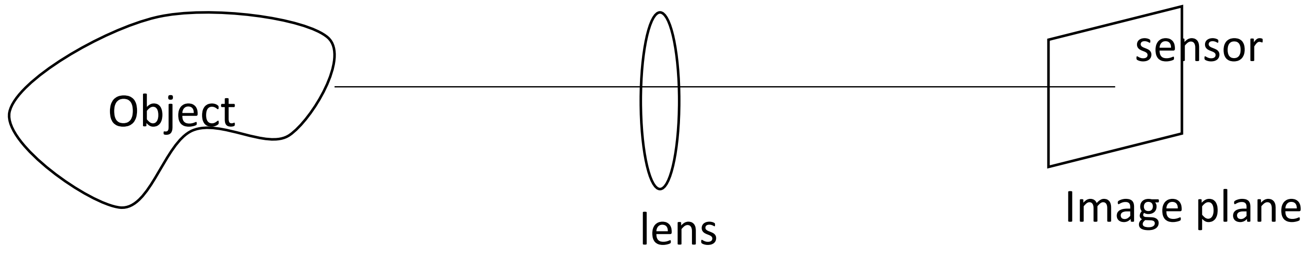
Enhancement vs. Restoration

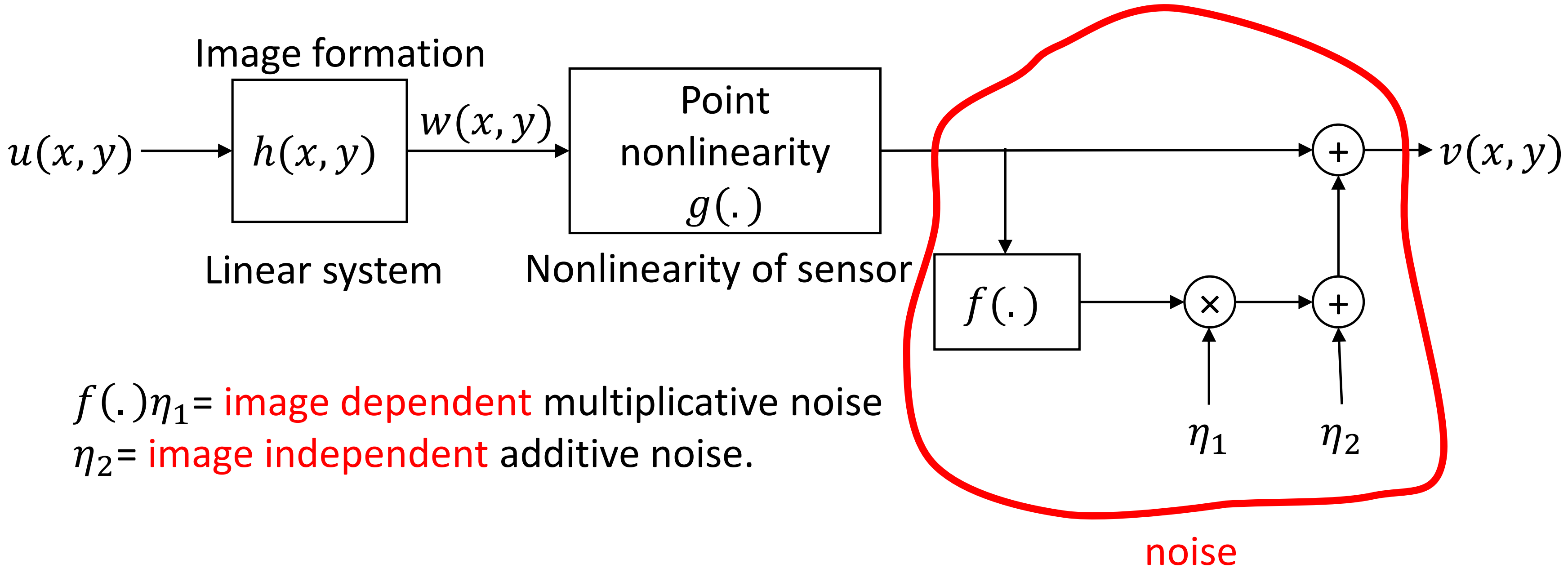
- Both enhancement and restoration aim to improve the quality of degraded images operations on them.
- Restoration: the model of degradation known, and this information is used to undo its effect and obtain the undegraded image. Assumes degradation is reversible.
- Enhancement: the model of degradation not known, and the improvements are done with the goal of obtaining a visually pleasing results or reducing noise for later processing.

Causes of degradation

- Noise in computer vision: Any entity in images, data, or intermediate result that is not interesting for the purposes of the main computation.
  - Noises can be additive or multiplicative.
  - Additive noise modeled as
    - $I_{result}(x, y) = I_{orig}(x, y) + n(x, y)$
  - Multiplicative noise modeled as
    - $I_{result}(x, y) = I_{orig}(x, y) * n(x, y)$

General Image observation model





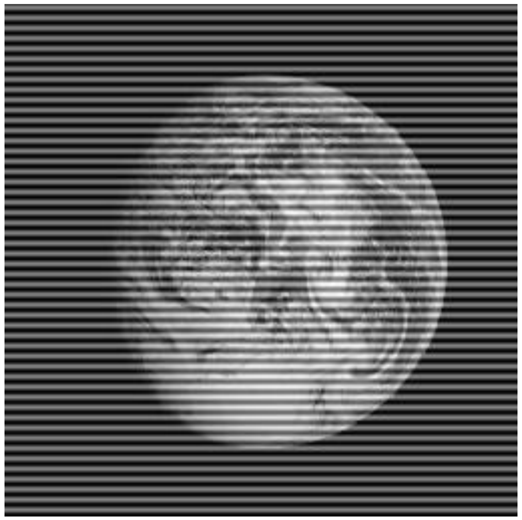
$f(.)\eta_1$= image dependent multiplicative noise
η_2= image independent additive noise.

Types of noise

- The degradation can be systematic or random
- In case degradation is random, the noise can come from a known probability distribution:
  - Gaussian
  - Uniform
  - Salt and pepper noise

Restoration

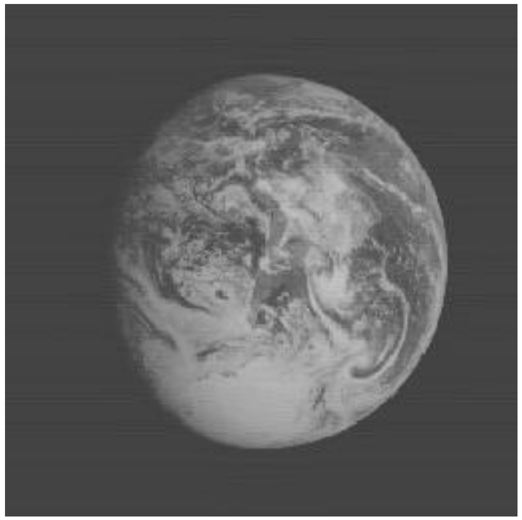
- We saw before how to remove a known (sinusoidal) noise from a degraded image.



Remove known sinusoidal noise

$\Rightarrow$

restoration



Before

After

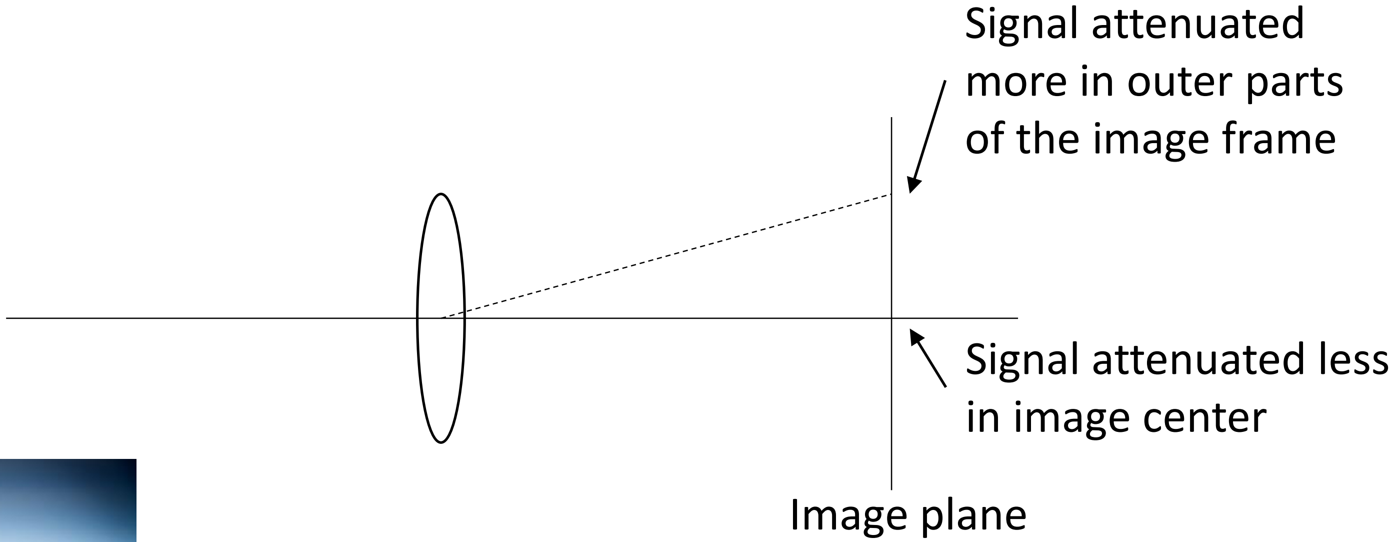
Restoration

We knew what the degradation was: a sinusoidal noise added to the image.

- We could undo the effect of the noise by
  - Identifying the frequency component, and
  - Filtering it out using frequency domain techniques.

Restoration

- <u>Another example</u>: Gray level correction (restoration) for uneven exposure (vignetting)


Signal attenuated more in outer parts of the image frame
Signal attenuated less in image center
Image plane

Outer areas of image darker due to vignetting.

Restoration (vignetting)

- Model the degradation as multiplicative process noise

$$\underbrace{g(x,y)}_{\text{actual image formed}} = \overbrace{e(x,y)}^{\text{attenuation}} \underbrace{f(x,y)}_{\text{input image}}$$

- If we know $e(x,y)$, then we can obtain the true image $f(x,y)$ from the data $g(x,y)$ by

$$f(x,y) = \frac{g(x,y)}{e(x,y)}$$

Restoration (vignetting)

- How do we obtain $e(x, y)$?
- Method: take the image of a uniform region.
- If $f_c(x, y) = c$, where c is a constant, then the image taken is $g_c(x, y)$.
- Now we can compute

$$e(x, y) = \frac{g_c(x, y)}{c}$$

- This can be considered as some sort of calibration process for image correction.

Restoration

- Now, given a degraded image $g(x, y)$ taken using the same imaging system, we can compute the original image $f(x, y)$ as follows:

$$f(x, y) = g(x, y)\left[\frac{c}{g_c(x, y)}\right]$$

Restoration

- NASA uses similar methods to perform color correction.

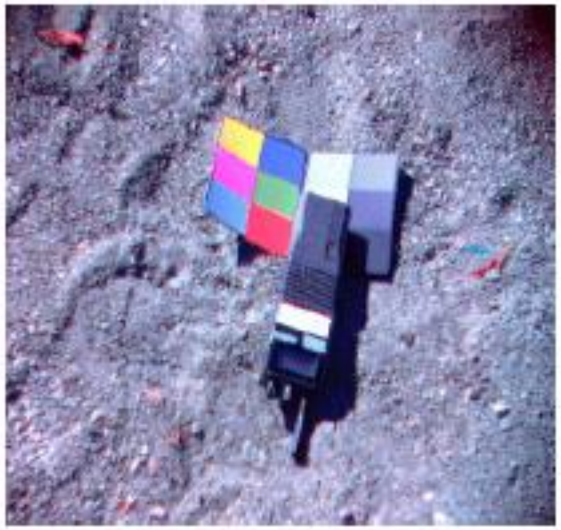
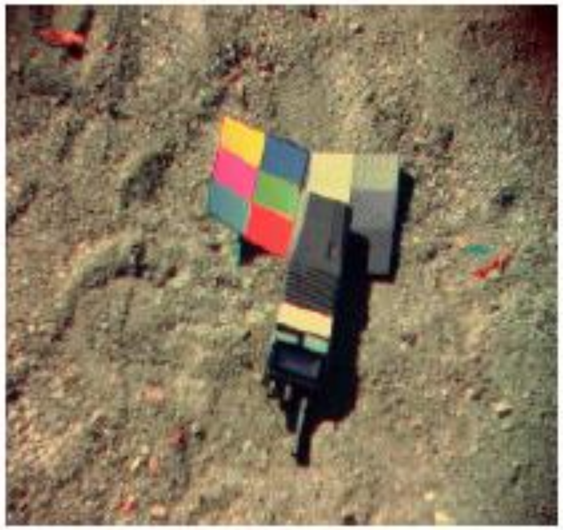


Image taken by NASA cameras


Corrected image

Restoration

- The color chart is the calibration item placed by NASA.
- The correction is done, by making sure that the RED, GREEN, and BLUE channels of the gray squares in the color panel have equal weights.

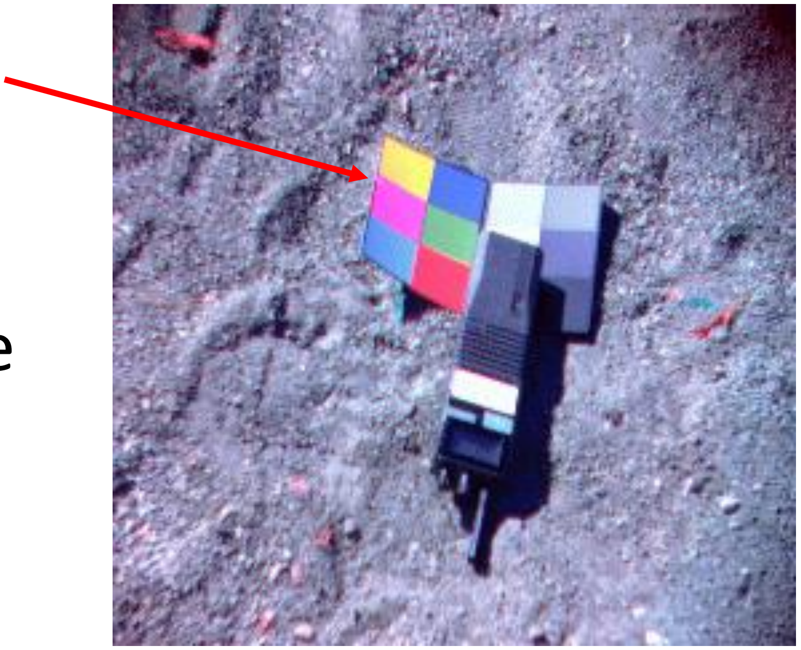


Image enhancement

- Goal: is to produce enhanced or cleaned up image for further processing and/or making the image more acceptable to human eye.
- Typically, the model of noise or degradation is not known.

Image enhancement methods

- Point operations
    - Contrast stretching
    - Gray level transforms
    - Histogram modifications
- Spatial filtering operations
    - Averaging
    - Median filtering
    - Unsharp masking
    - Low pass, band pass, high pass filtering
    - Zooming
- Pseudocoloring

Point operations

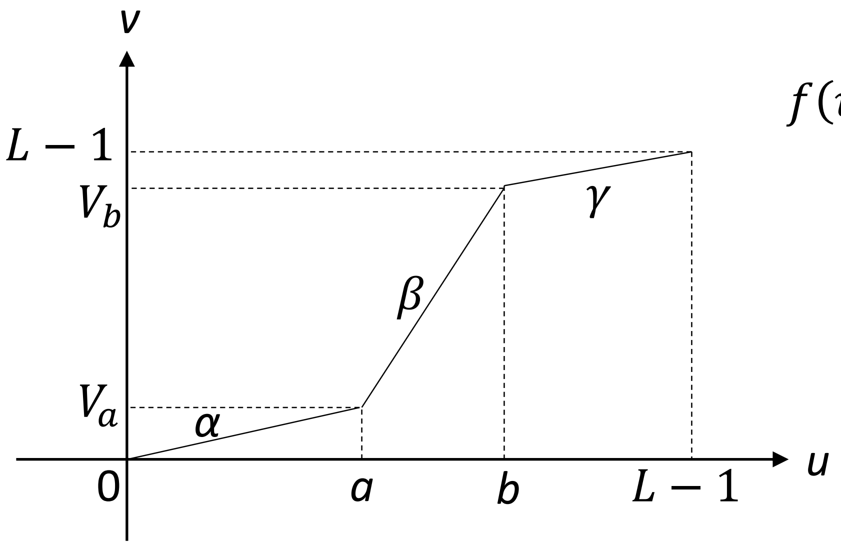
- Each pixel's gray value is modified independently of the values of its spatial neighbors.
- Input u: gray level $\in [0, L-1]$ where L is the number of gray levels.
- Output v: gray level $\in [0, L-1]$

$$v = f(u)$$

- Sometimes referred to as zero-memory methods, because there is no need to remember surrounding pixel values.

Point operations

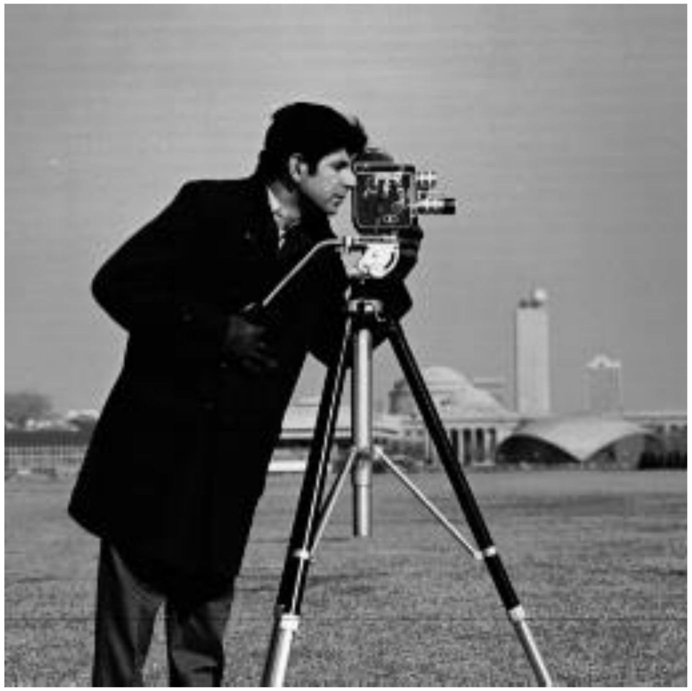
- Contrast stretching (piecewise linear gray range transformation):

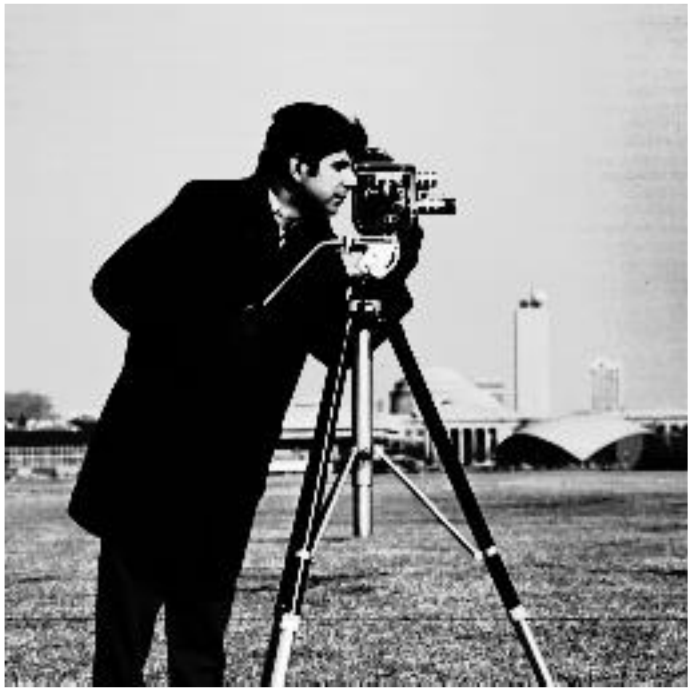


$$f(u) = \begin{cases} \alpha u & 0 \le u < a \\ \beta(u-a) + V_a & a \le u < b \\ \gamma(u-b) + V_b & b \le u < L \end{cases}$$

- Useful for images in which most gray levels fall into a narrow range of values.

Point operations

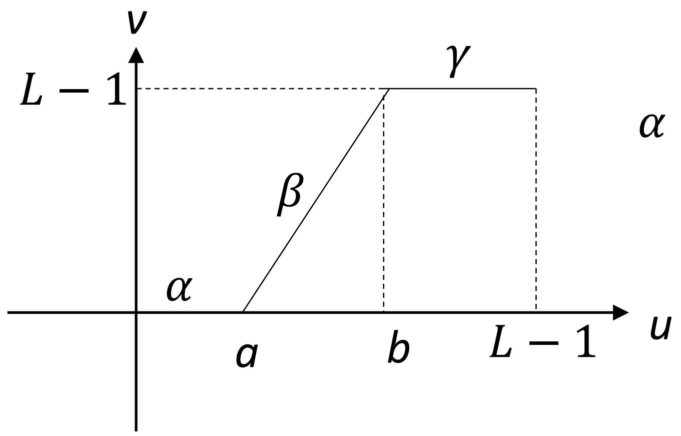

before


after

$$a = 84, V_a = 26$$
$$b = 155, V_b = 215$$

Point operations

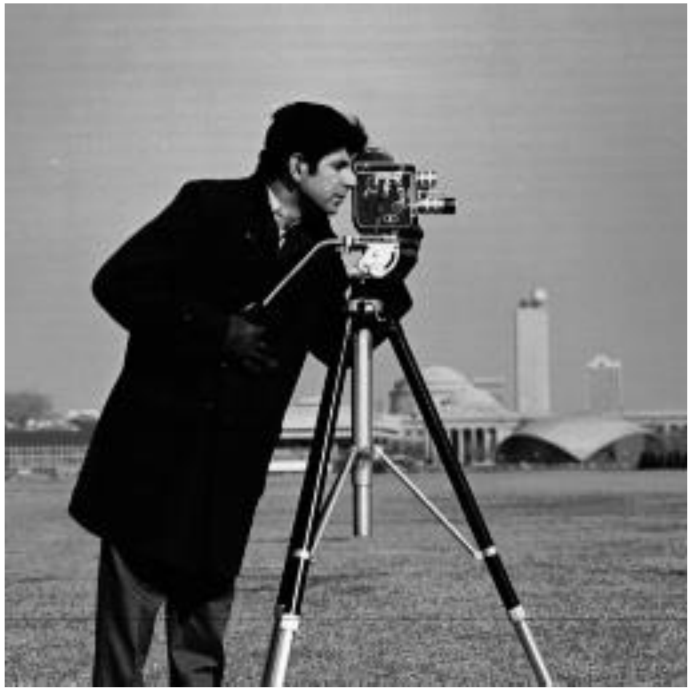
- Clipping (similar to contrast stretching)

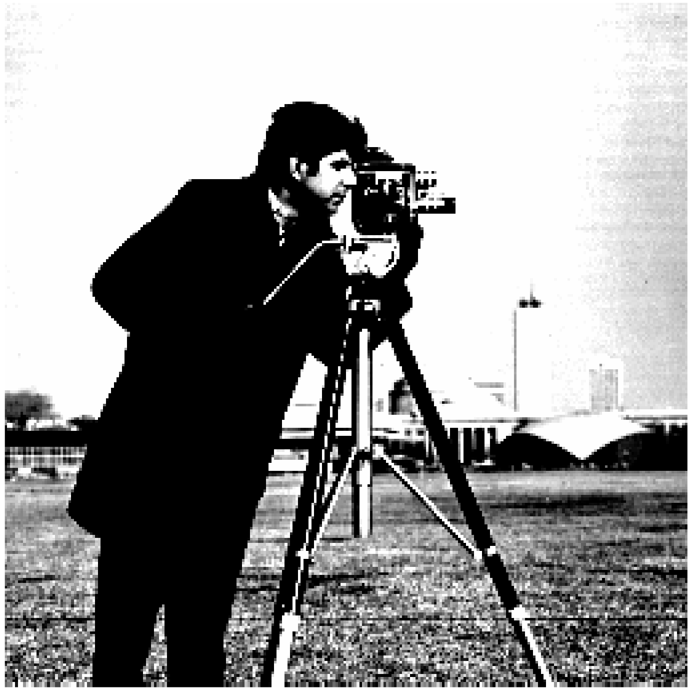


$\alpha = \gamma = 0$

- If signal lies in the range $[a, b]$, this is a way to stretch the contrast.

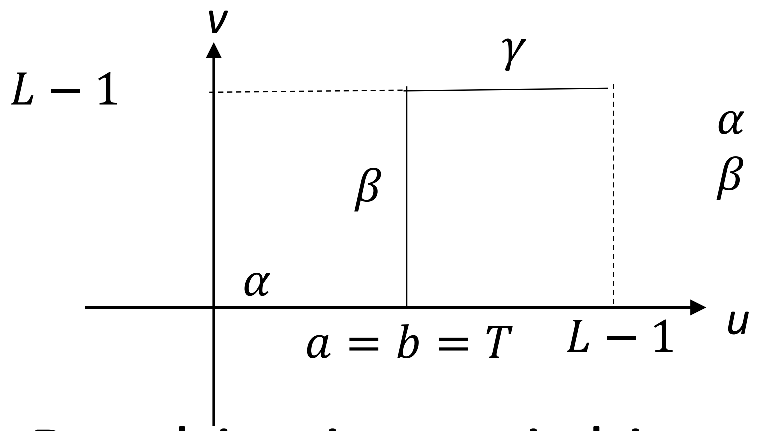
Point operations


before


after

Point operations

- Thresholding (clipping taken to limit)

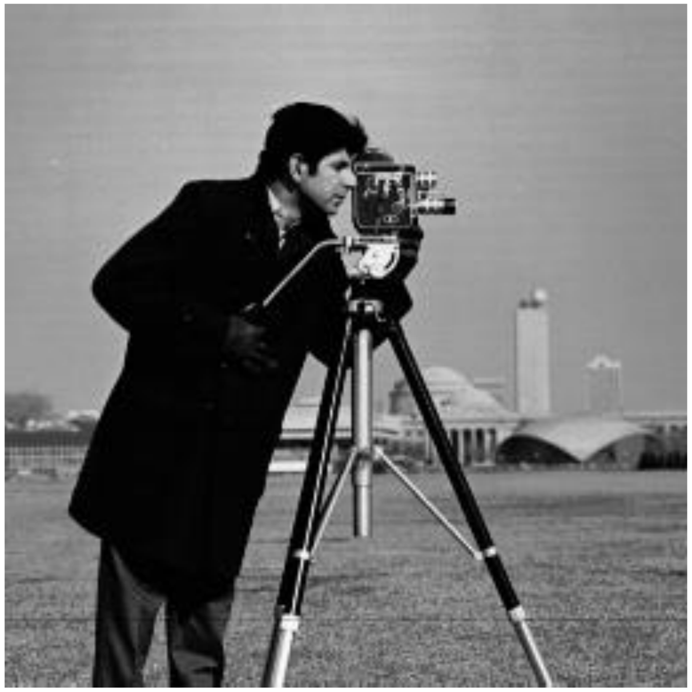


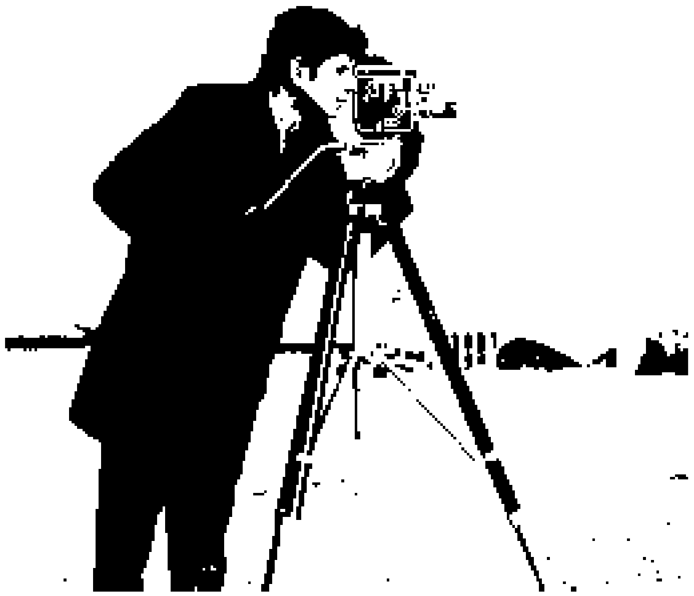
$\alpha = \gamma = 0$
$\beta = \infty$

- Resulting image is binary.

Point operations

- Thresholding example





before

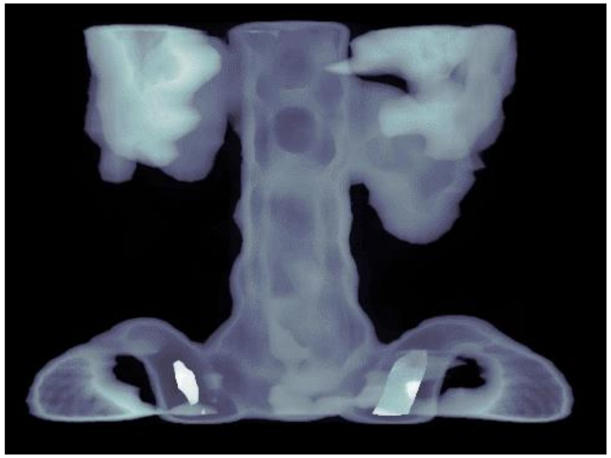
After
(T=74)

Point operations

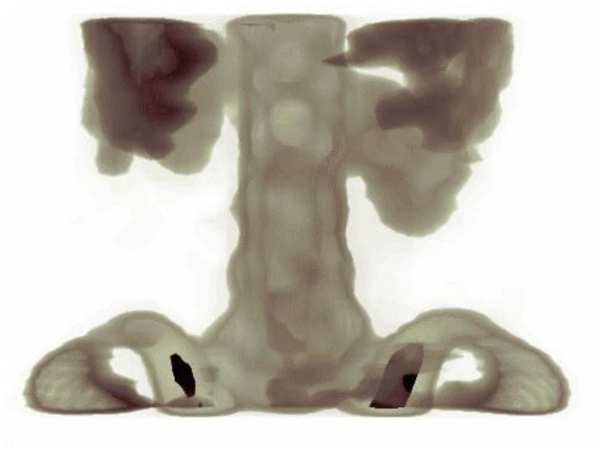
- Gray scale reversal

$$f(u) = (L-1) - u$$

- Example usage: displaying X-ray images which are normally negatives.



original



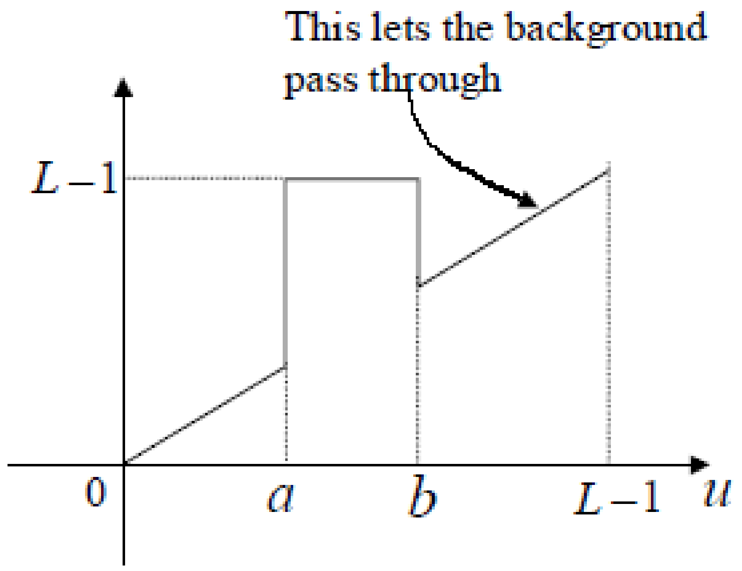
Gray scale reversed

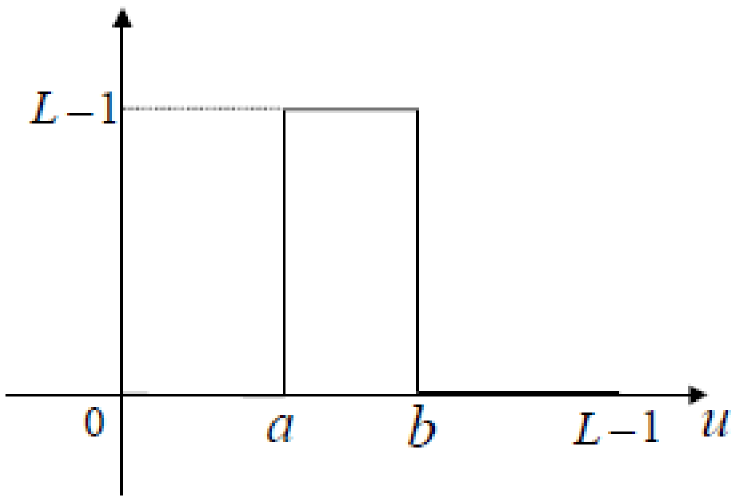
Point operations

Gray level window slicing

- When a certain image feature occurs at a certain gray level, window slicing can be used, to bring out the feature.
- Two ways to do it
    - With background.
    - Without background.

Point operations





With background

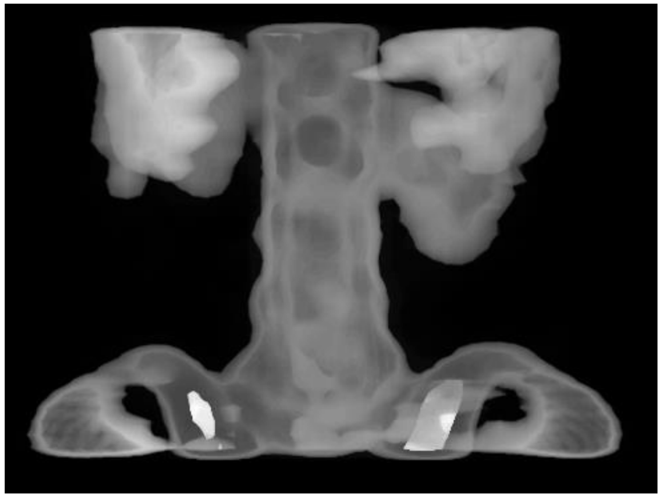
$$f(u) = \begin{cases} L-1 & a \le u \le b \\ u & \text{otherwise} \end{cases}$$

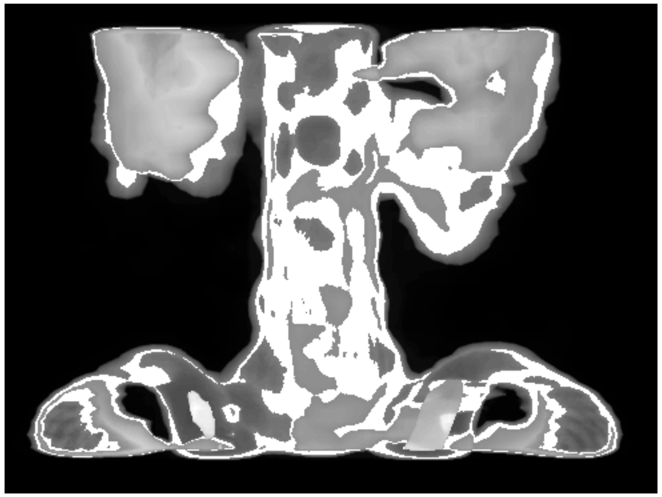
Without background

$$f(u) = \begin{cases} L-1 & a \le u \le b \\ 0 & \text{otherwise} \end{cases}$$

Window slicing example

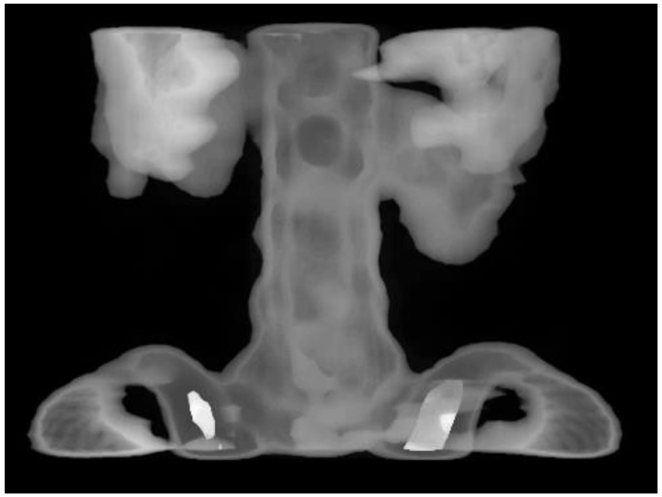
- With background. Parameters: *a*=111, *b*=131

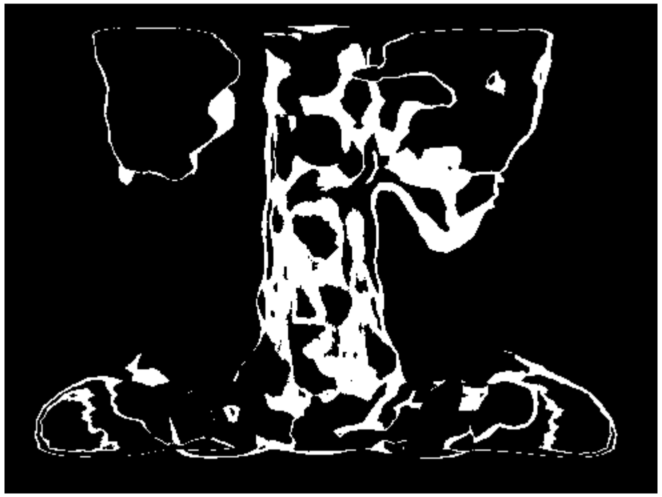

original


After bone sections highlighted

Window slicing example

- Without background. Parameters: *a*=111, *b*=131


original


After bone sections highlighted; background suppressed

Point operations

Bit extraction

- If each gray level u is represented by B bits, then $f(u)$ extracts the *nth* most significant bit
- Example: can be used in determining the number of useful bits.

Range compression

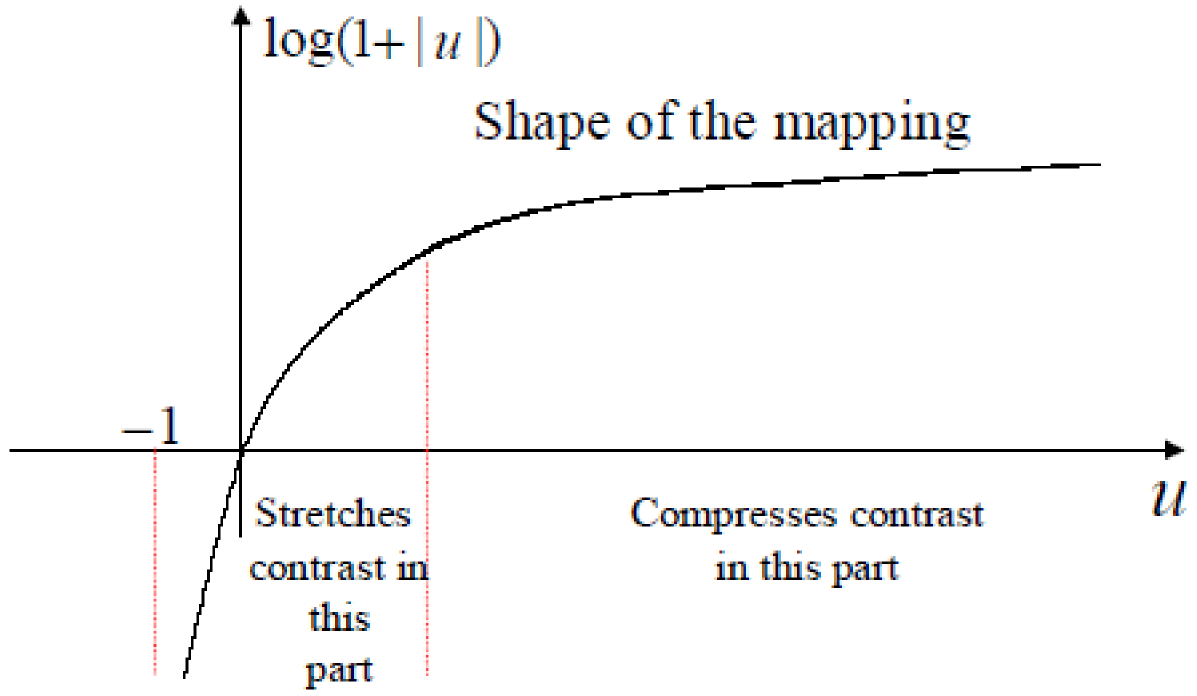
- If the dynamic range of an image is too large, its range can be compressed.

$$f(u) = c \log_{10}(1 + |u|)$$

- The logarithmic transform also tends to bring out details in darker regions.

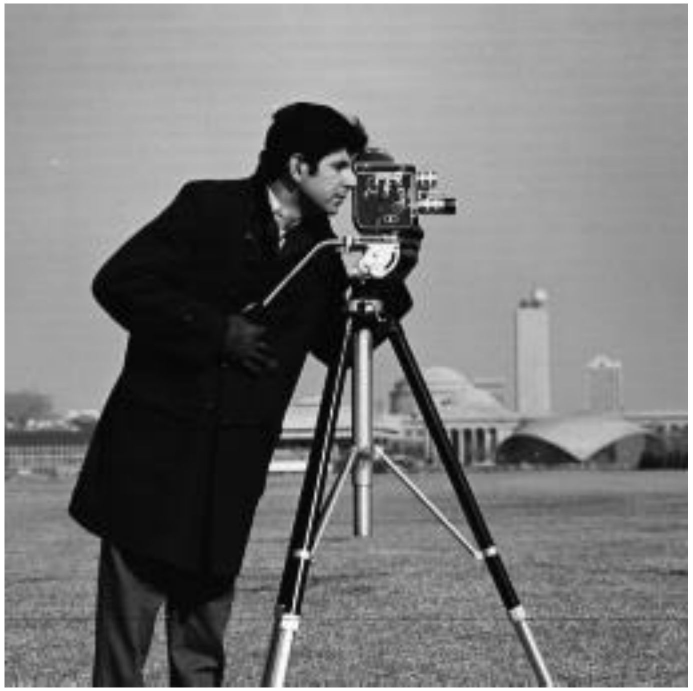
Log transform

- Nonlinear contrast stretching

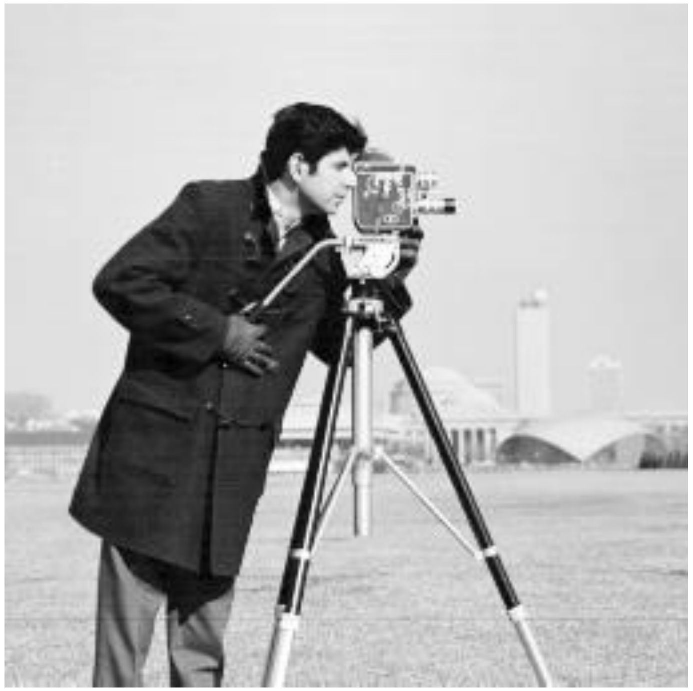


$\log(1+|u|)$
Shape of the mapping
-1
u
Stretches contrast in this part
Compresses contrast in this part

Log transform example



original



Log transform applied:
More details visible in dark regions of the image (e.g., coat)

Point operations based on histogram

Definitions:

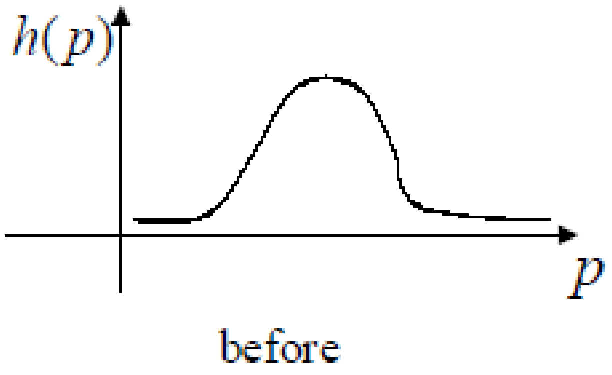
- The relative frequency of occurrence of gray levels in an image is $h(u)$.
- If $h(u)$ is normalized such that

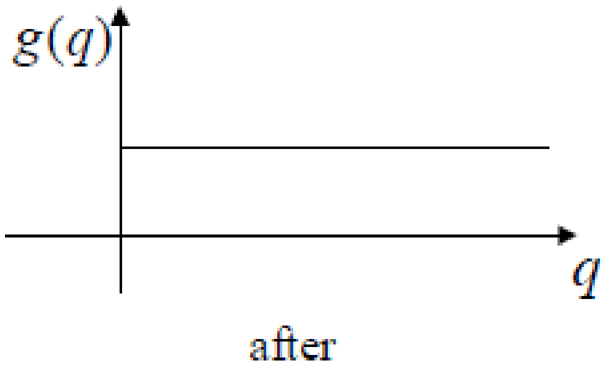
$$\int_0^L h(u)du = \text{image area}$$

then $h(u)$ is called the histogram.

Histogram operations

- A common technique for increasing contrast in an image is histogram equalization (or histogram flattening).
- The goal is to redistribute gray levels among pixels so that the resulting histogram is flat.



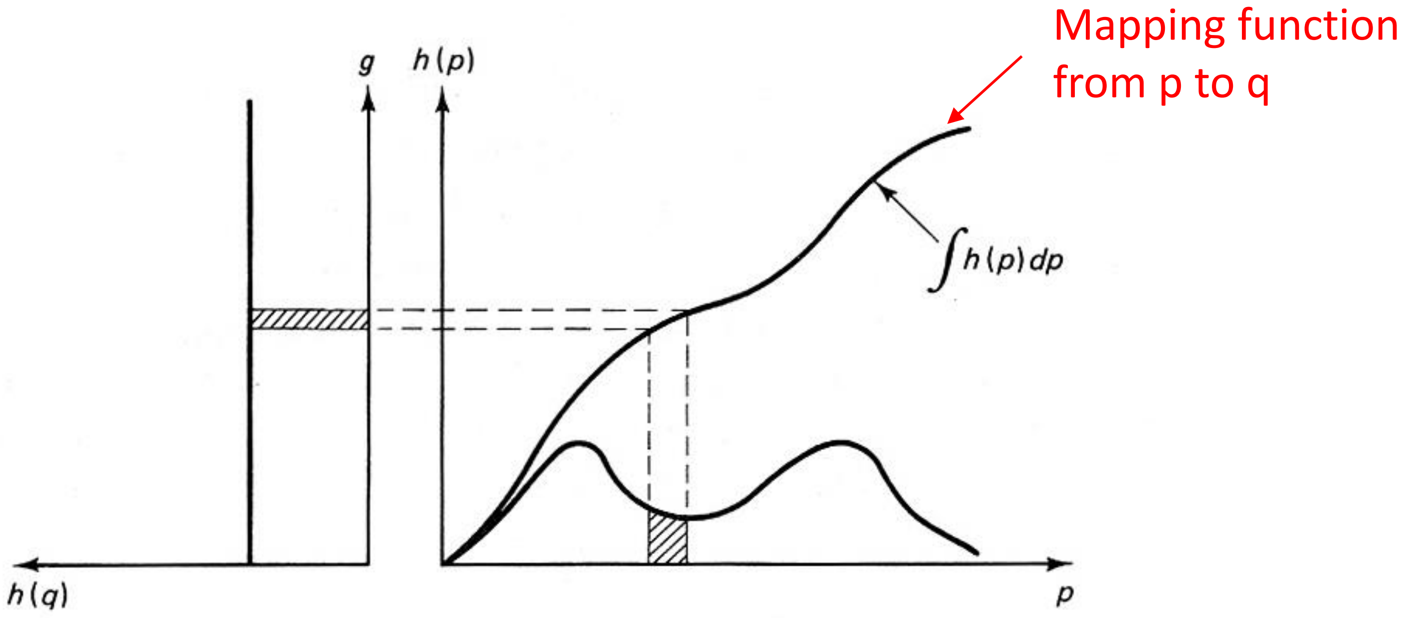


Histogram equalization

- The result of this redistribution is that the gray level range is expanded near the peak of the histogram (where most pixels are) and compressed near the tails.
- Since gray levels are stretched for most pixels, more detail is visible.
- How is this transformation done?

Histogram equalization

- Define a mapping *E*: *P*→ *Q*, where *P* is the set of gray levels for the input image, and *Q* is the new set of transformed gray levels.



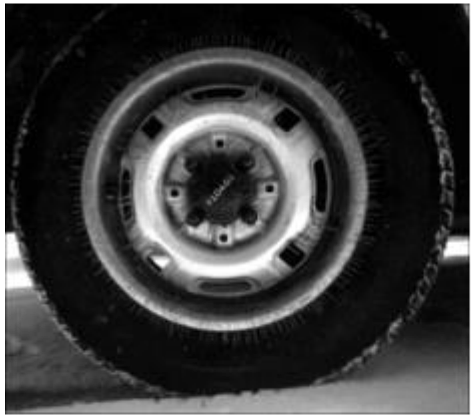
Histogram equalization

- With discrete values and a range of pixel values [0,255], we can use:

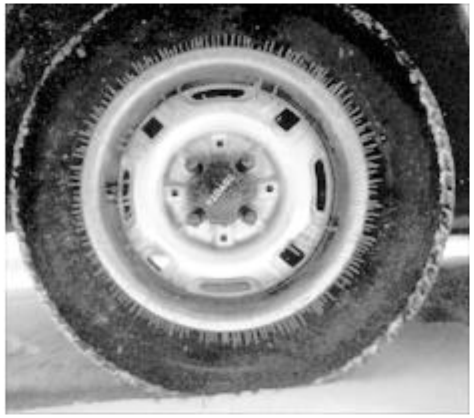
$$c(I) = \frac{1}{N}\sum_{i=0}^{I} h(i) = c(I-1) + \frac{1}{N}h(I)$$

- Where N is the total number of pixels (or image area)
- $c(I)$ is the cumulative distribution of (area under) the histogram up to the value I.

Histogram equalization example



Before



After

More details visible in the dark tire region

Magnification (zooming)

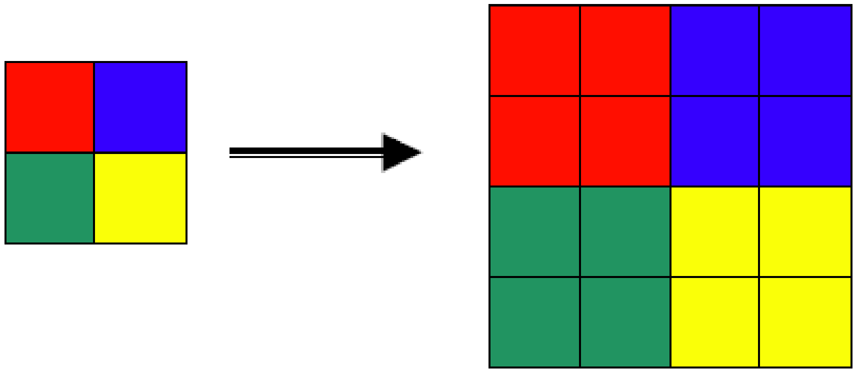
- How and why do we scale images?
  - We may want to do this for display purposes, or
  - We may want to do it for multiscale image processing without losing quality or information
- We can zoom in (i.e., make the image larger), or
- We can zoom out (i.e., make the image smaller).

Zooming in

- Method 1: Replication
    - For doubling each dimension:
    - Let f be the resulting image and I be the input image:
- Each pixel in a scan line is repeated once

$$f(i, 2j) = I(i, j)$$
$$f(i, 2j+1) = I(i, j)$$

- Each scan line is repeated once.



Zooming in

- Method 2: Linear interpolation
  - The intensities between two pixels are linearly interpolated in a row

$$f_1(i, 2j) = I(i, j)$$
$$f_1(i, 2j+1) = \tfrac{1}{2}[I(i,j) + I(i, j+1)]$$

  - Then each scan line is interpolated

$$f(2i, j) = f_1(i, j)$$
$$f(2i+1, j) = \tfrac{1}{2}[f_1(i,j) + f_1(i+1, j)]$$

10	50
20	60

$\Rightarrow$

10	30	50	?
15	35	55	?
20	40	60	?
?	?	?	?

2x zoom by linear interpolation

Reds first

Greens next

Zooming out

- Method 1: Sampling
  - Every other pixel is sampled from the rows and columns
- Method 2: Averaging
  - 2x2 windows are averaged and the result is assigned to the appropriate pixel in the smaller resulting image.

Zooming out by sampling

10	31	40	30
10	25	30	32
15	30	27	40
20	35	35	50

$\Longrightarrow$

10	40
15	27

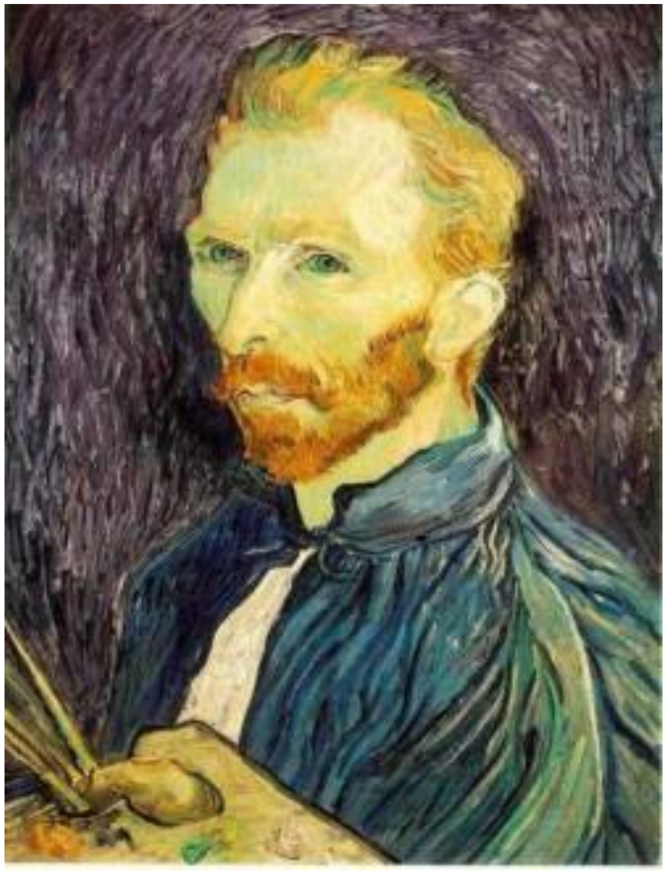
Zooming out by averaging

$$\begin{array}{|c|c|c|c|}\hline 10 & 31 & 40 & 30 \\ \hline 10 & 25 & 30 & 32 \\ \hline 15 & 30 & 27 & 40 \\ \hline 20 & 35 & 35 & 50 \\ \hline \end{array} \Rightarrow \begin{array}{|c|c|}\hline 19 & 33 \\ \hline 25 & 38 \\ \hline \end{array}$$

Other interpolation and sampling methods

- filtering and sampling
  - Gaussian filter one possibility

Subsampling with Gaussian pre-filtering

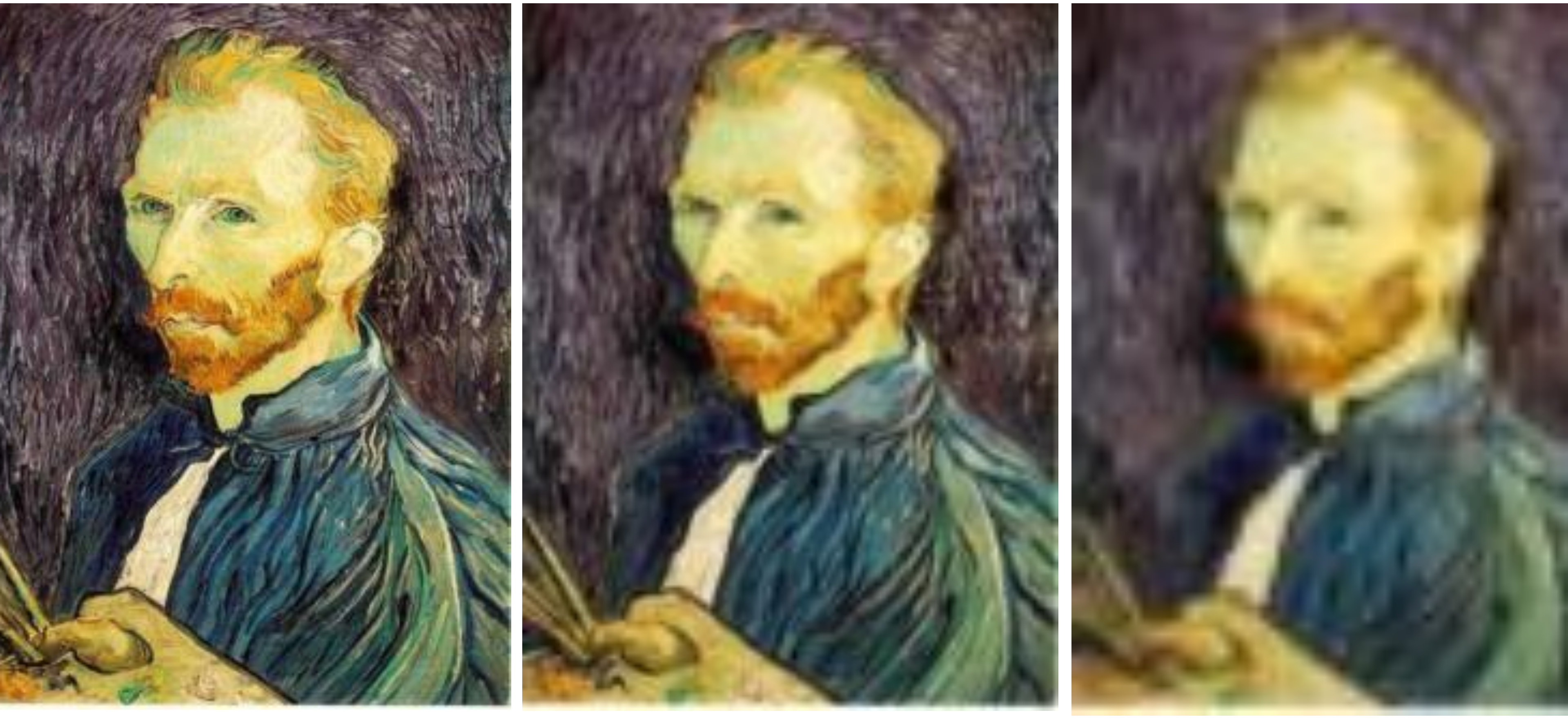

Gaussian 1/2

G 1/4

G 1/8

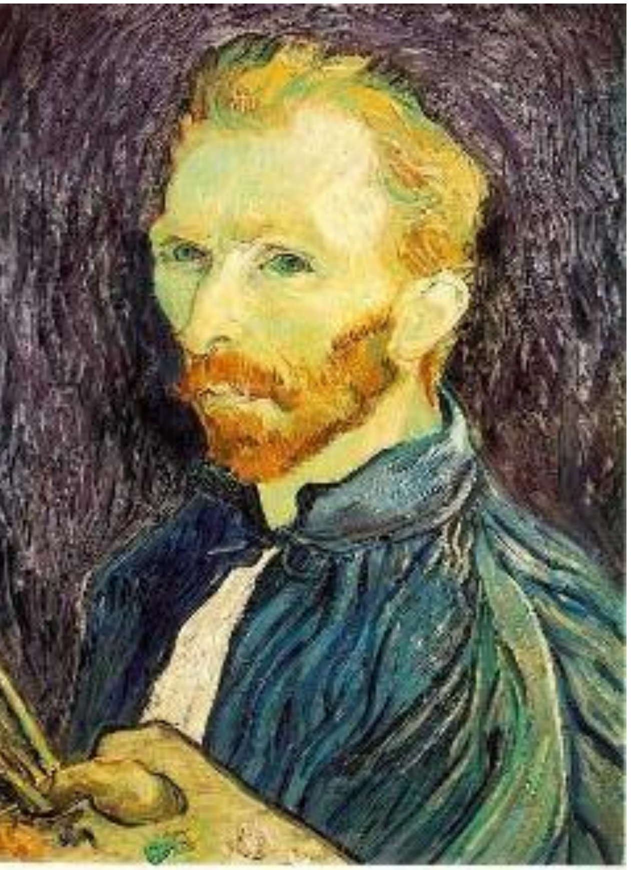
Solution: filter the image, *then* subsample

Subsampling with Gaussian pre-filtering

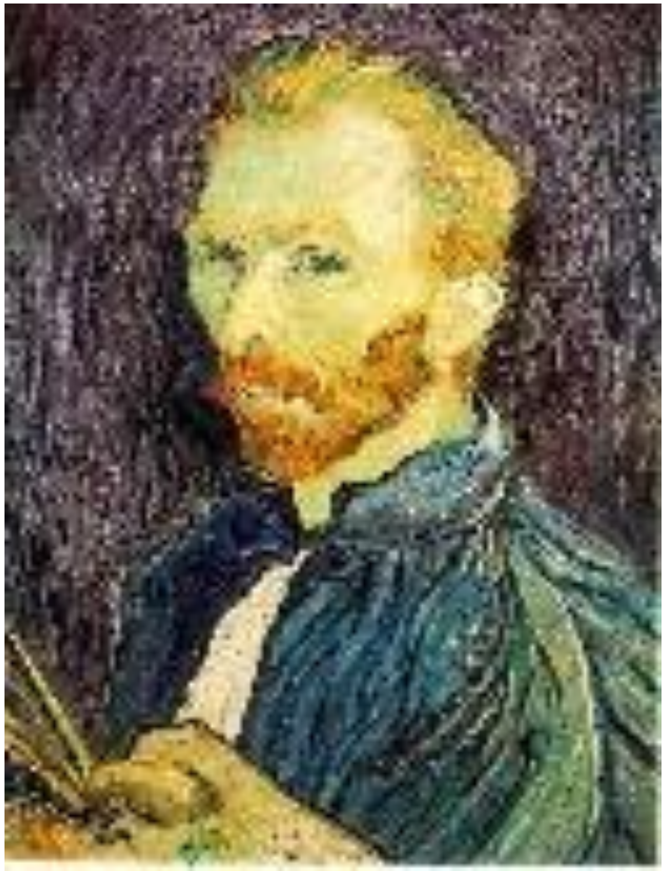

Gaussian 1/2 G 1/4 G 1/8

Solution: filter the image, *then* subsample

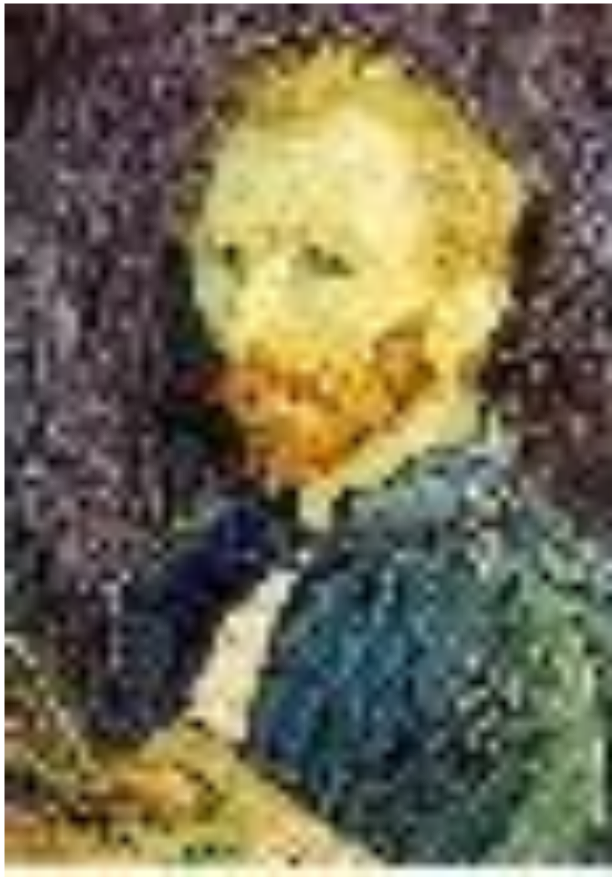
Compare with...



1/2

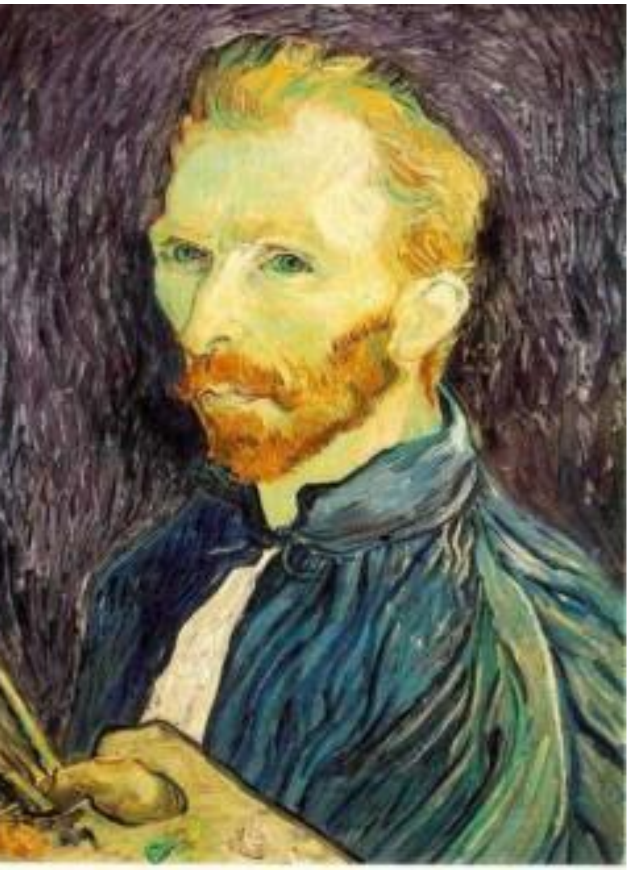


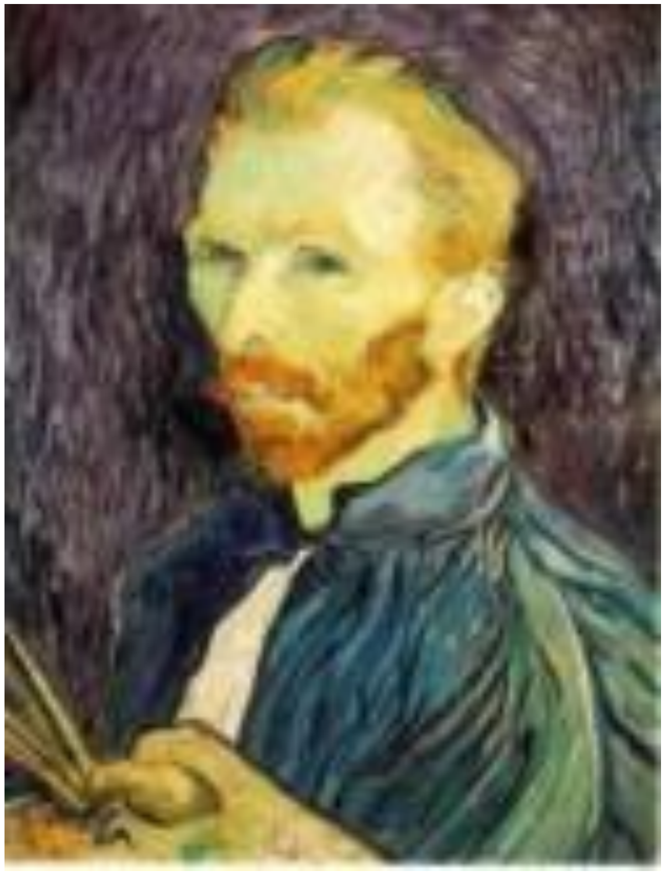
1/4 (2x zoom)

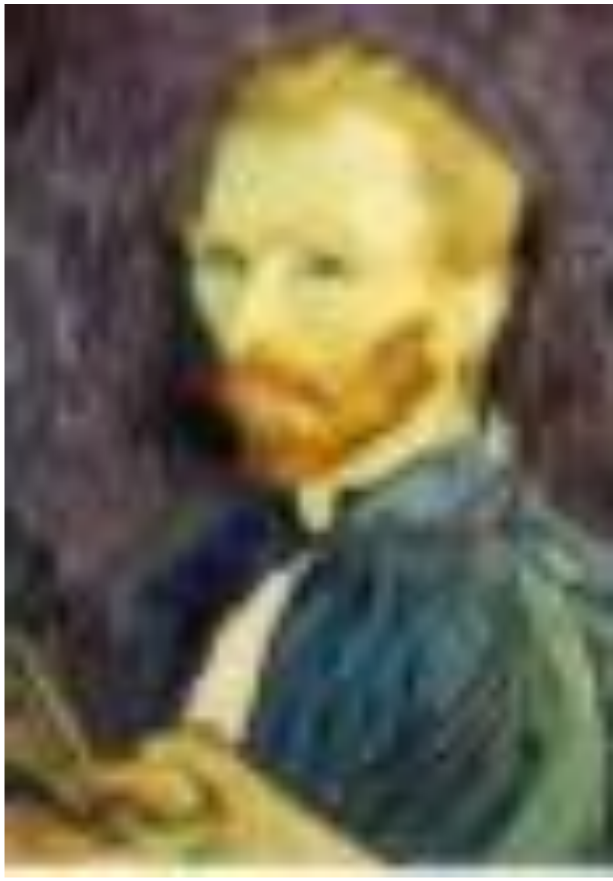


1/8 (4x zoom)

Subsampling with Gaussian pre-filtering


Gaussian 1/2

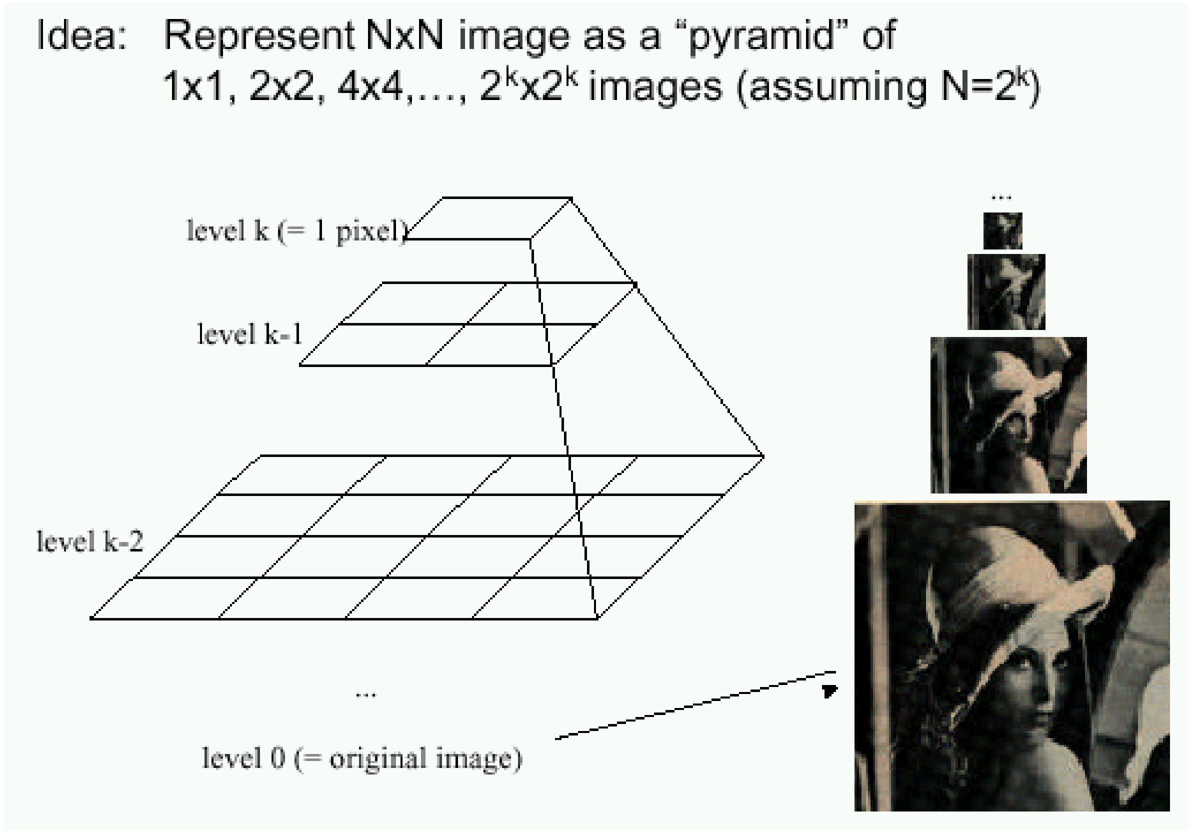

G 1/4


G 1/8

Solution: filter the image, *then* subsample

- How can we speed this up? Precompute at many scales

Sometimes we want many resolutions


Idea: Represent NxN image as a "pyramid" of
1x1, 2x2, 4x4,..., 2^kx2^k images (assuming N=2^k)
level k (= 1 pixel)
level k-1
level k-2
...
level 0 (= original image)
...

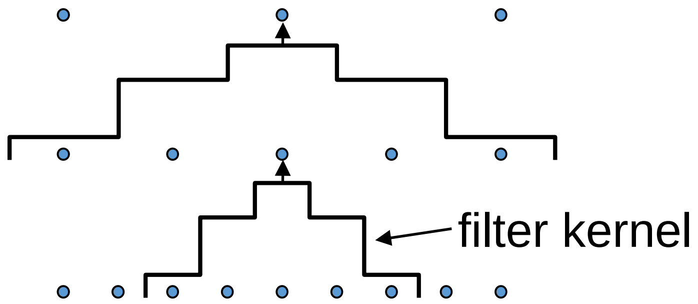
Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

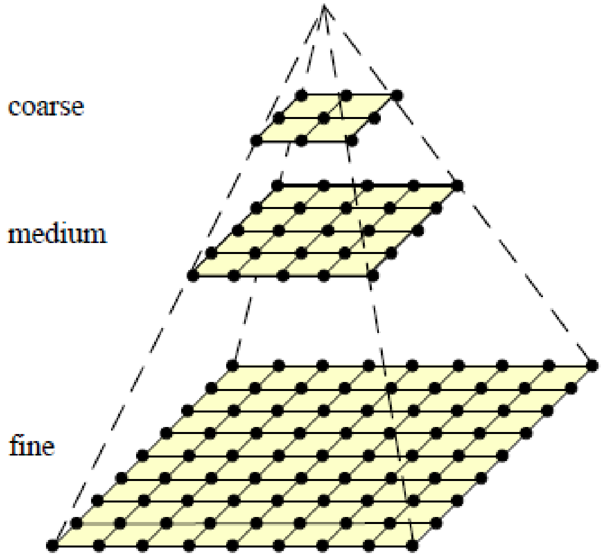
- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

Gaussian Pyramids have all sorts of applications in computer vision

- We'll talk about these later in the course

Gaussian pyramid construction





Repeat

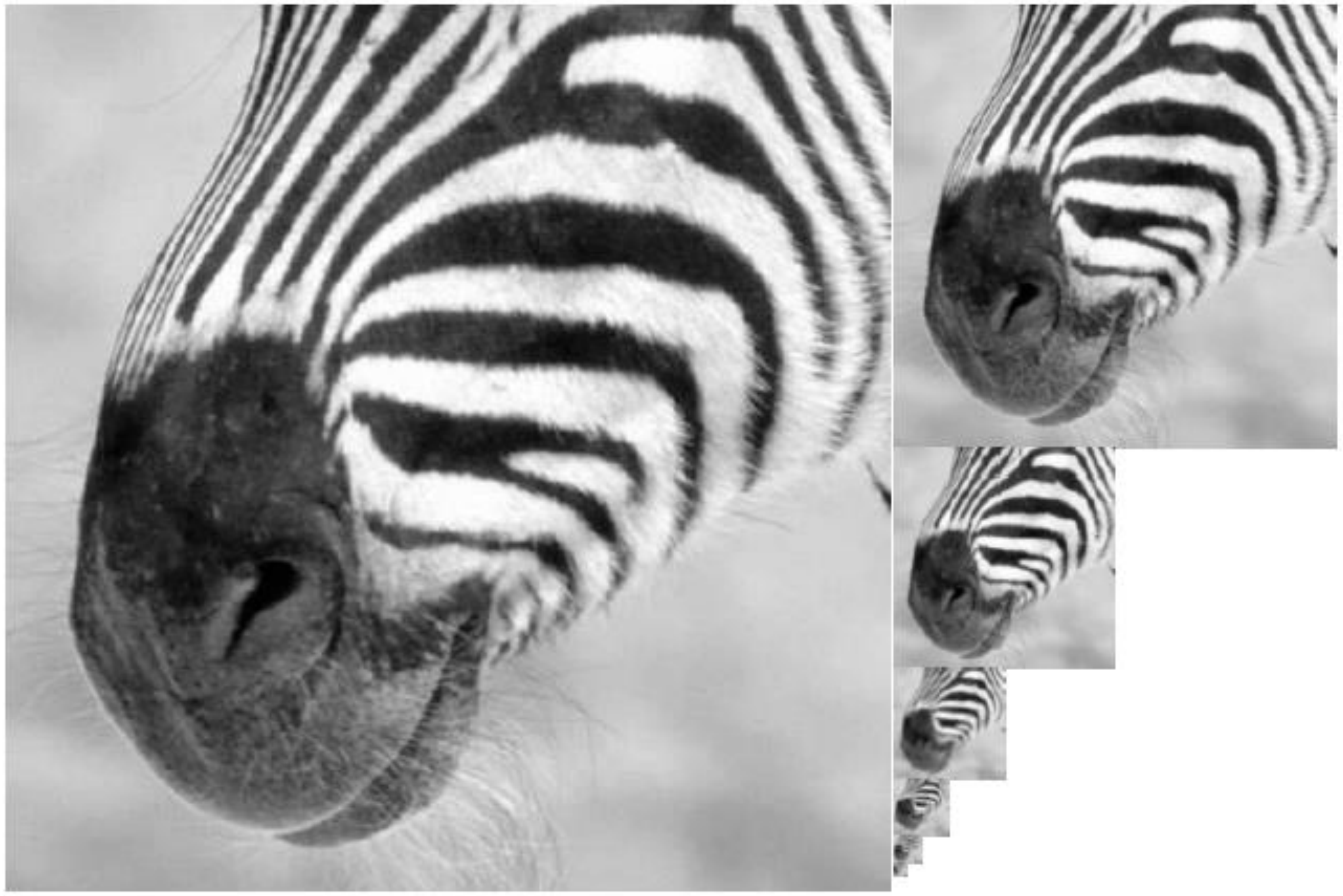
- Filter
- Subsample

Until minimum resolution reached

- can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!

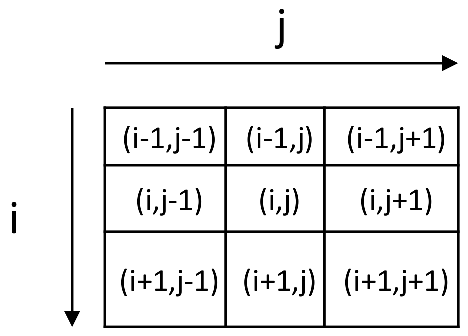
Gaussian pyramid example



From book by Forsythe and Ponce.

Spatial (neighborhood) operations

- Weighted combinations of pixels in small neighborhoods
- Simplest enhancement operation is replacing a pixel value with the average of its neighbors. (4-neighbors, 8-neighbors or other neighborhoods)
- Let $I(i,j)$ be the original image and $f(i,j)$ be the computed image.



$N(i,j)$: neighbors of pixel (i,j)

This is convolution

$$f(i,j) = \sum_{(k,l)\in N(i,j)} \sum w_{k,l} I(i-k, j-l)$$

$w_{k,l}$: are weights.

Spatial (neighborhood) operations

- All equal weights can be accomplished by

$$f(i,j) = \frac{1}{|N(i,j)|} \sum_{(k,l)\in N(i,j)} \sum I(i-k, j-l)$$

- Weights all equal to $1/|N(i,j)|$.
- For a 3×3 neighborhood, $|N(i,j)| = 9$.

$$f(i,j) = \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} I(i-k, j-l)$$

- This can be viewed as a 2D mask of weights on the neighborhood $N(i,j)$.
- How is this mask used?

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Masks and convolution

- In image processing, various convolution operations are often represented by masks with which the image is convolved.
- This is accomplished by the following steps
  1. Place mask on pixel (i, j) for which result is to be computed
  2. Compute point wise multiplication of all pixels of the image and mask that overlap
  3. Sum up the products
  4. Replace pixel (i, j) value with the computed sum of products
  5. Slide mask over to the next pixel and repeat operation

Example: box filter

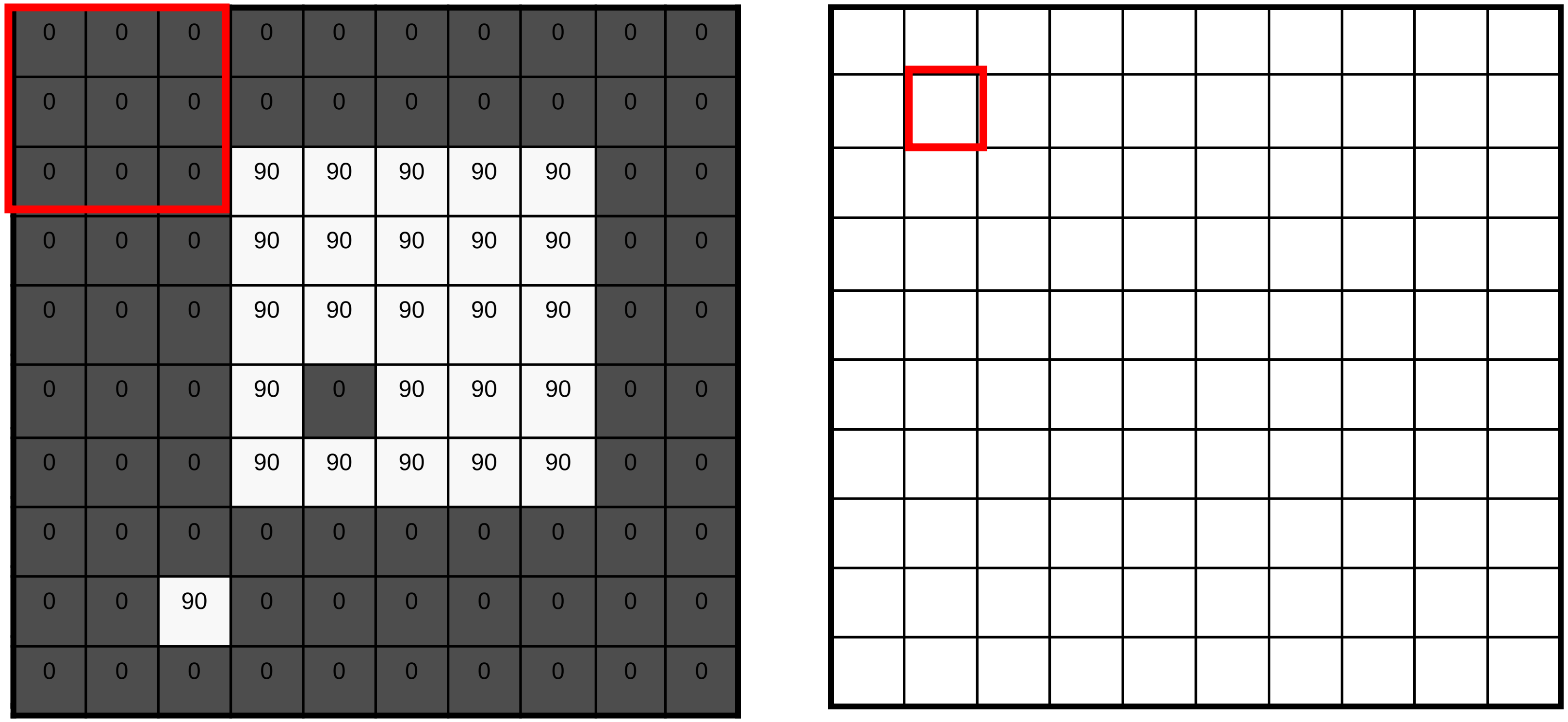
$$g[\cdot,\cdot] \qquad \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Slide credit: David Lowe (UBC)

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$ $\qquad\qquad$ $h[.,.]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20						

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot\,,\cdot\,]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
				?					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Image filtering

$$g[\cdot,\cdot] \quad \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$f[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[.,.]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k, n+l]$$

Credit: S. Seitz

Box Filter

What does it do?

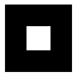
- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

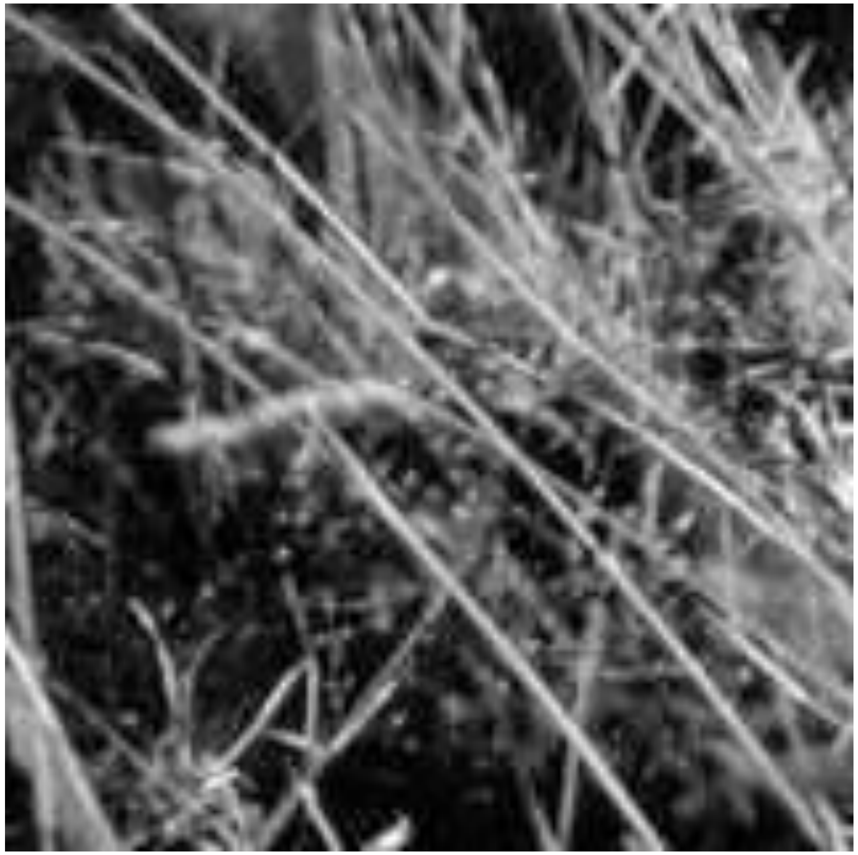
$$g[\cdot\,,\cdot\,]$$

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

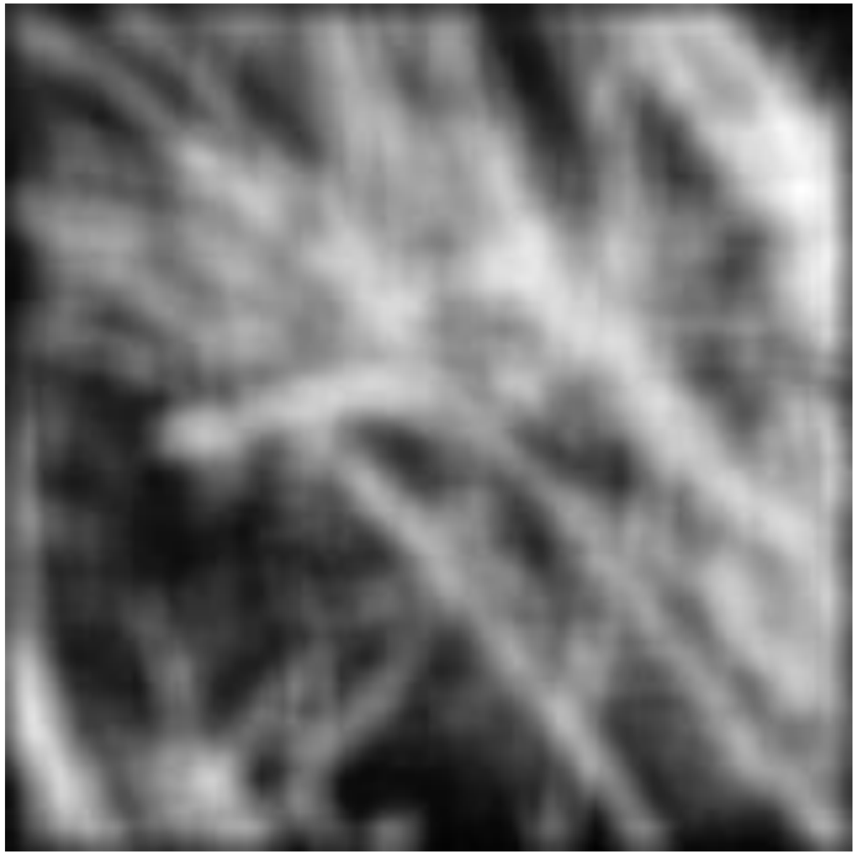
Slide credit: David Lowe (UBC)

Smoothing with box filter





Before



After

Masks and convolution

- Now we can design filters in the spatial domain by designing the proper masks.
  - we have a general way to compute many spatial filtering operations.
- Example: other weighted averaging filters

0	1/8	0
1/8	1/4	1/8
0	1/8	0

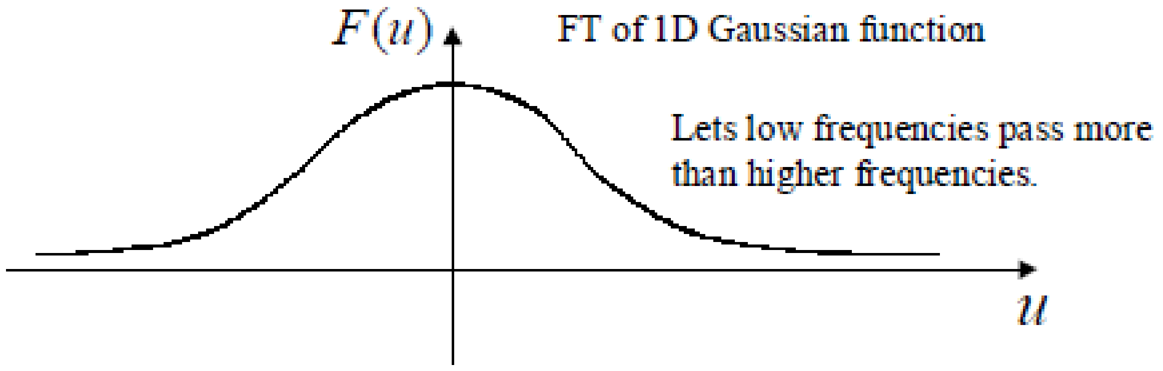
New weights ➔ gives more weight
to pixels closer to the central pixel (1/4).

Spatial averaging filters

- The weights can come from a Gaussian function.

$$w(x,y) = e^{-a(x^2+y^2)}$$

- Recall that the FT of this function is itself a Gaussian. Therefore, this filter is a low-pass filter.



Gaussian separability

- A function $g(a,b)$ is said to be separable if there are two functions g_1 and g_2 of one variable each such that
$$g(a,b) = g_1(a)g_2(b)$$
- The 2D Gaussian with unit area
$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$
is separable to
$$g_1(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}$$
such that
$$g(x,y) = g_1(x)g_1(y)$$

2D Gaussian filter

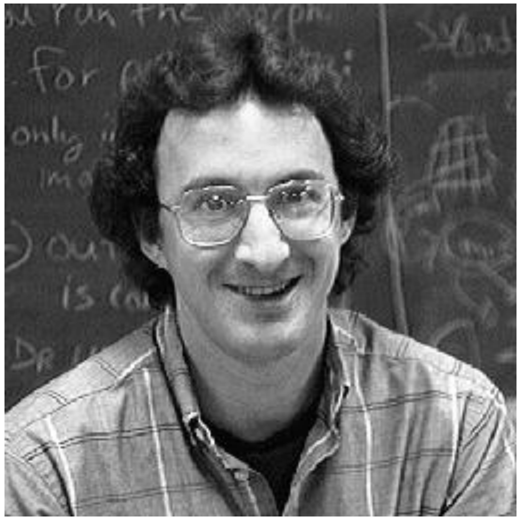
- Convolution of $f(x,y)$ with a 2D Gaussian filter $g(x,y)$ is given by:

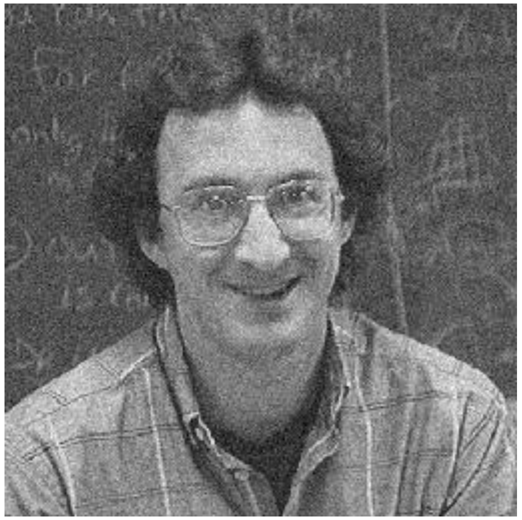
$$f * g = \iint e^{-a(\xi^2+\eta^2)} f(x-\xi, y-\eta)\,d\xi\,d\eta$$

$$= \int e^{-a\xi^2}\left[\int e^{-a\eta^2} f(x-\xi, y-\eta)\,d\eta\right] d\xi$$

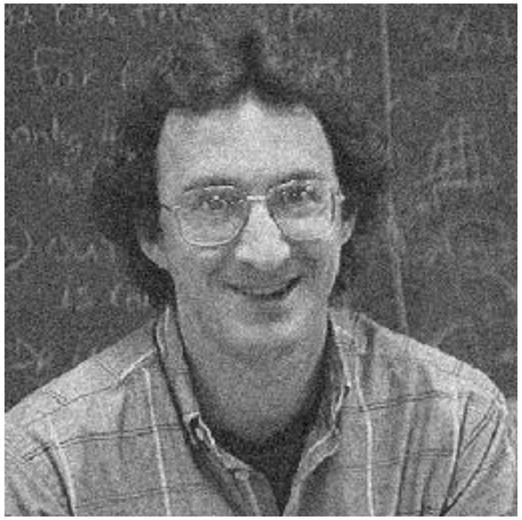
- The 2D Gaussian function is separable. That is, convolving a 2D function with a 2D Gaussian can be accomplished with 2 1D convolutions in sequence, one in the y direction, one in the x direction.
- Computationally more efficient!

Spatial averaging examples


Original image


Gaussian noise added

Spatial averaging examples



Input image with Gaussian noise added

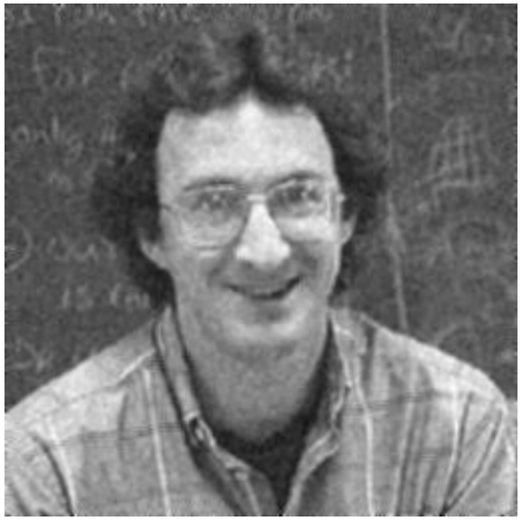


Image filtered with 3x3 box averaging

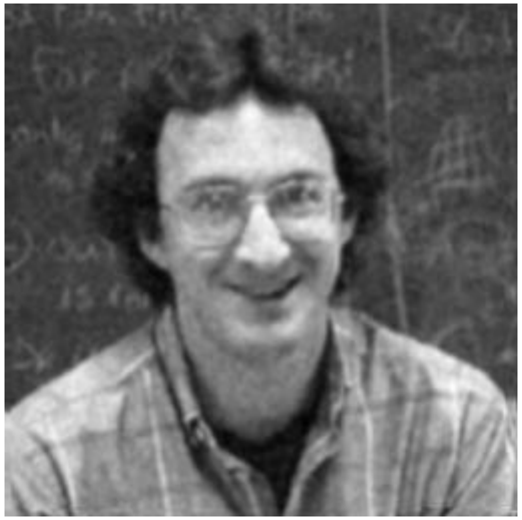


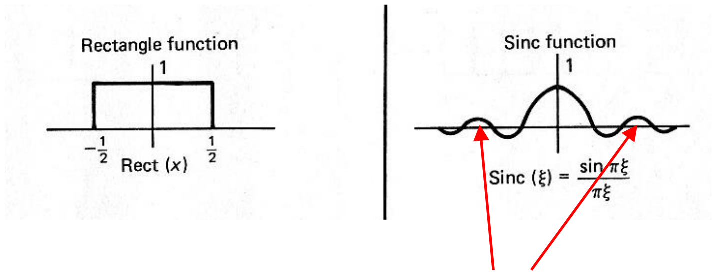
Image filtered with Gaussian function

Why is Gaussian filtered image smoother than box filtered image?

Answer

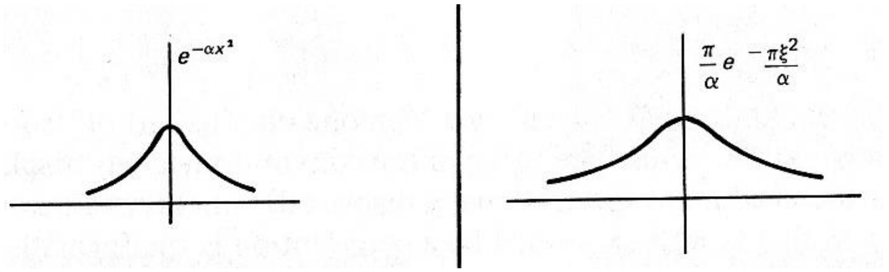
- The answer is in the frequency domain characteristics of the two filters.
- Let's look at 1D versions of box filter and Gaussian filter.

Box filter: Rect(x) function and its FT



Sinc function lets more high frequency components pass through

Gaussian function and its FT



Gaussian kills high frequency components

Smoothing filters

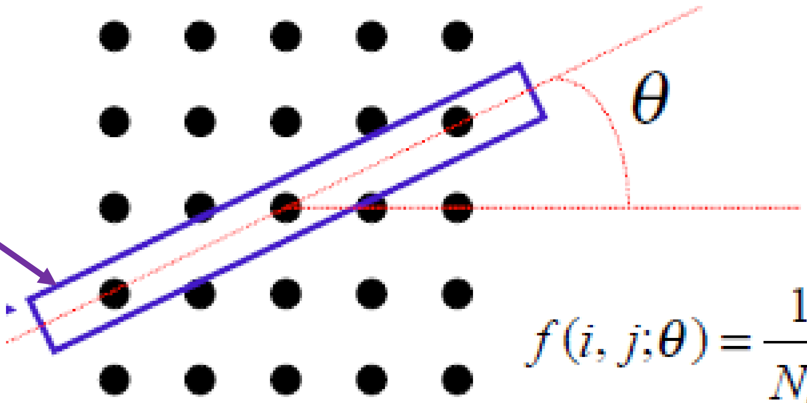
Problem:

- They smooth out and eliminate noise, but they also blur edges.
- Solution:
  - Directional smoothing
  - Median filtering (nonlinear)

Directional smoothing

- Spatial averages $f(i,j;\theta)$ are computed for various θ.

W_θ: averaging window in θ direction



$$f(i,j;\theta) = \frac{1}{N_\theta} \sum\sum_{(k,\ell)\in W_\theta} I(i-k, n-\ell)$$

Directional smoothing

- Then for each pixel (i, j) a direction θ^* is found such that

$$|I(i, j) - f(i, j; \theta^*)| \text{ is minimum}$$

- Then for pixel (i, j)

$$f(i, j) = f(i, j; \theta^*)$$

- The idea is not to smooth over an edge with high contrast.

Median filtering

- Input pixel is replaced by the median value in a window.
  $$f(i,j) = \text{Median}(I(i-k, j-l), \text{ for}(k,l) \in W\}$$

<u>Example (1D median using window size 3):</u>

Original	0	0	6	0	0	1	2	1	0	1	2	0	0	4	4	4	0
Average		2	2	2	1/3	1	4/3	1	2/3	1	1	2/3	4/3	8/3	4	8/3	
Median		0	0	0	0	1	1	1	1	1	1	0	0	4	4	4	

Single line lost

Step edge preserved

Median filtering

Corners are clipped

0	0	0	0
1	1	0	0
1	1	0	0

Before

0	0	0	0
1	0	0	0
1	1	0	0

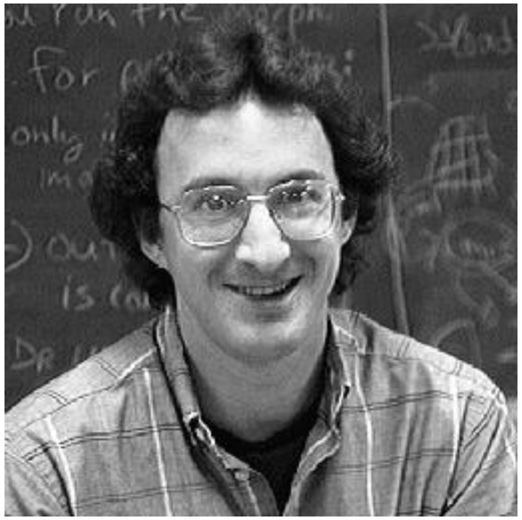
After

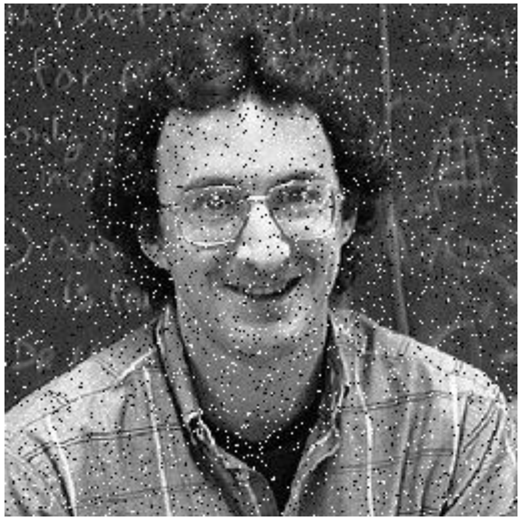
Median filtering

Properties:

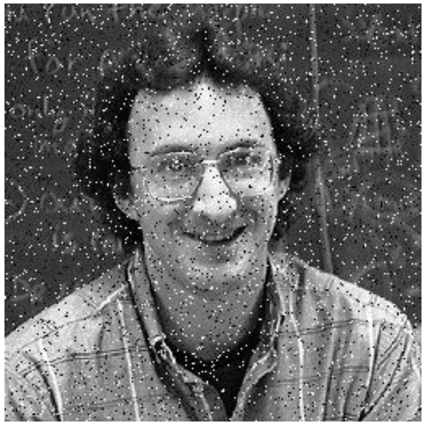
- Nonlinear filter
- Useful for removing isolated pixels or lines (of course, we may not want to lose lines, in which case this would be considered bad).
- Preserves step edges (does not smooth them)
- Clips corners
- Performs well on binary noise (salt-and-pepper noise) but poorly on Gaussian noise.
- Computationally costly (needs sorting of values)
  - Practical implementation: separable median filter: i.e., two 1D median filters in *x* and *y* directions.

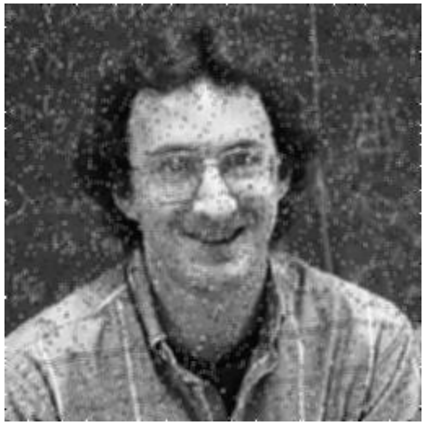
Median filter example

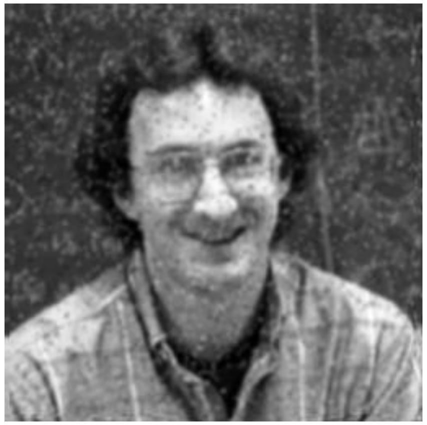

Original image

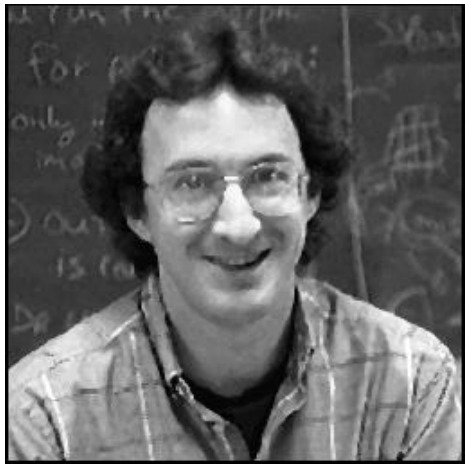

Salt-and-pepper noise added

Median filter example


Salt-and-pepper noise added


Box average 3x3 window


Gaussian filtered


Median filtered 3x3 window.

Other spatial filtering operations

- One can model certain image degradations with physical phenomena and try to invert the process.
- Example: Model image blurring with heat diffusion process. Then to sharpen the image try to recover the image before heat diffusion by reversing the diffusion process.
- The resulting filter is called unsharp masking.
  - Very common in most image processing tools.

Heat diffusion

- The heat equation is given by

$$\frac{\partial g}{\partial t} = k\nabla^2 g \text{ where } k > 0$$

- Where $g(x, y, t)$ is the temperature as a function of the spatial variables $x, y, \text{and } t$.
- So we are modeling the image function as the temperature on a surface which is changing with time.
- Diffusion equation says the rate at which the temperature changes at a given point ($\partial g/\partial t$) is proportional to the difference of temperature between this point and its neighboring points, the Laplacian ($\nabla^2 g$).

Modeling blur with diffusion

- At $t = 0$, $g(x, y, 0)$ is the unblurred image that we are trying to recover.
- At $t = \tau > 0$, we have $g(x, y, \tau) = I(x, y)$ which is the blurred image.
- As a side note, the solution to the heat diffusion equation is a Gaussian.
- To sharpen the image we try to recover $g(x, y, 0)$ from the blurred image $I(x, y) = g(x, y, \tau)$, knowing the model of the blurring process.

Sharpening using diffusion equation

- Expand $g(x, y, \tau)$ around $t = \tau$ using Taylor series:

$$g(x, y, t) = g(x, y, \tau) + (t - \tau)\frac{\partial g(x, y, \tau)}{\partial t} + \cdots$$

- Evaluating this around $t = 0$ and keeping only the first order term, we have an approximation:

$$\begin{aligned} g(x, y, 0) &= g(x, y, \tau) + (0 - \tau)\frac{\partial g(x, y, \tau)}{\partial t} \\ &= g(x, y, \tau) - \tau\frac{\partial g(x, y, \tau)}{\partial t} \end{aligned}$$

Sharpening using diffusion equation

- Substitute $f(x, y)$ for $g(x, y, 0)$ and $k\nabla^2 g$ for $\frac{\partial g}{\partial t}$, we get

$$f \cong g - k\tau\nabla^2 g$$

- Use discrete approximations for the derivatives by using differences on discrete image grid:

∂f / ∂x	Partial derivative in x direction	Δ_x	Discrete difference
∂f / ∂y	Partial derivative in y direction	Δ_y	Discrete difference

- Δ_x and Δ_y can be approximated on the discrete grid in various ways.

Discrete approximations to derivative operations

- Non symmetric ways:

$$\Delta_x f(i,j) = f(i,j) - f(i,j-1)$$
$$\Delta_y f(i,j) = f(i,j) - f(i-1,j)$$

- There could be symmetric versions, too:

$$\Delta_x f(i,j) = f(i,j+1) - f(i,j-1)$$
$$\Delta_y f(i,j) = f(i+1,j) - f(i-1,j)$$

Discrete approximations to derivative operations

- Discrete versions of the second order difference is given by:
- $\Delta_x^2 = \Delta_x f(i, j+1) - \Delta_x f(i, j)$
  $= [f(i, j+1) - f(i, j)] - [f(i, j) - f(i, j-1)]$
  $= f(i, j+1) + f(i, j-1) - 2f(i, j)$
- $\Delta_y^2 = \Delta_y f(i+1, j) - \Delta_y f(i, j)$
  $= [f(i+1, j) - f(i, j)] - [f(i, j) - f(i-1, j)]$
  $= f(i+1, j) + f(i-1, j) - 2f(i, j)$

Discrete approximations to derivative operations

- Now define the Laplacian:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- By the following discrete approximation
- $\nabla^2 f(i,j) = \Delta_x^2 f(i,j) + \Delta_y^2 f(i,j)$

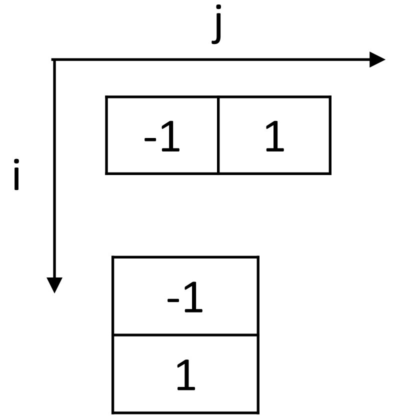
$$\begin{aligned} &= [f(i,j+1) + f(i,j-1) - 2f(i,j)] \\ &+[f(i+1,j) + f(i-1,j) - 2f(i,j)] \\ &= f(i+1,j) + f(i-1,j) + f(i,j+1) \\ &+ f(i,j-1) - 4f(i,j) \end{aligned}$$

Discrete approximations of derivative operations

- For pixel (i, j) the discrete approximations can be represented by masks:

$$\Delta_x f(i,j) = f(i,j) - f(i,j-1)$$

$$\Delta_y f(i,j) = f(i,j) - f(i-1,j)$$



Discrete approximations of derivative operations

- Finally, for
- $\nabla^2 f(i,j) = f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1) - 4f(i,j)$

0	1	0
1	-4	1
0	1	0

How are masks used?

- Instead of writing the expressions explicitly for the operations we are computing, we can write them in terms of weight (mask values).
- Let mask values be $m(i,j)$. Then

$$\nabla^2 f(i,j) = f(i+1,j) + f(i-1,j) + f(i,j+1) + f(i,j-1) - 4f(i,j)$$

$$= \underbrace{\sum_{k=-1}^{1}\sum_{l=-1}^{1} m(k,l) f(i-k, j-l)}_{\text{Bu this is once again convolution!}}$$

Back to unsharp masking using diffusion equation

- We want to sharpen the image using the formula
$$f \cong g - k\tau\nabla^2 g$$
- Assume for simplicity that $k = 1$ and $\tau = 1$. Then the discrete approximation for f is:
$$f(i,j) = g(i,j) - \nabla^2 g(i,j)$$

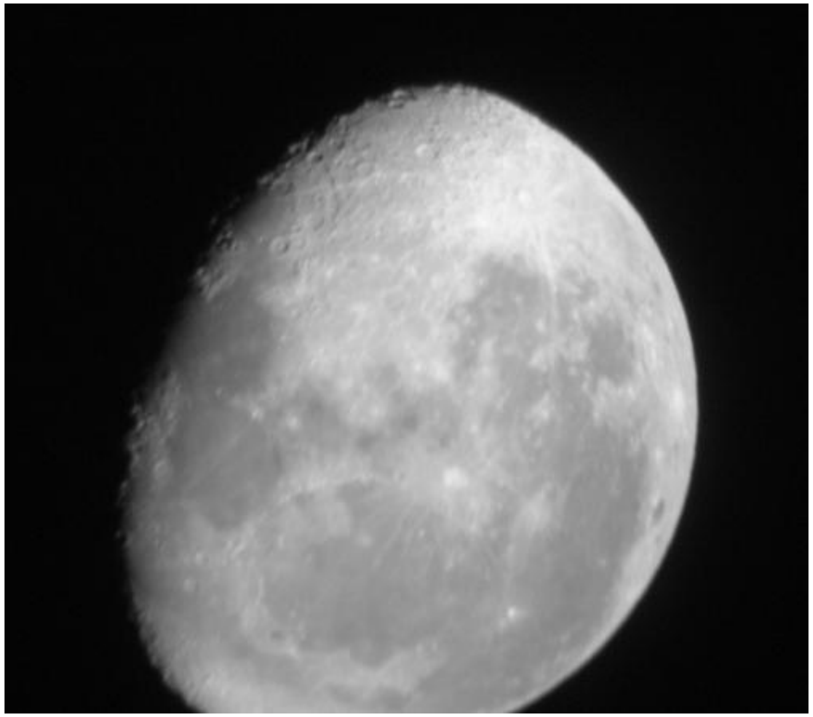
Back to unsharp masking using diffusion equation

- In terms of convolution masks, this is given by:

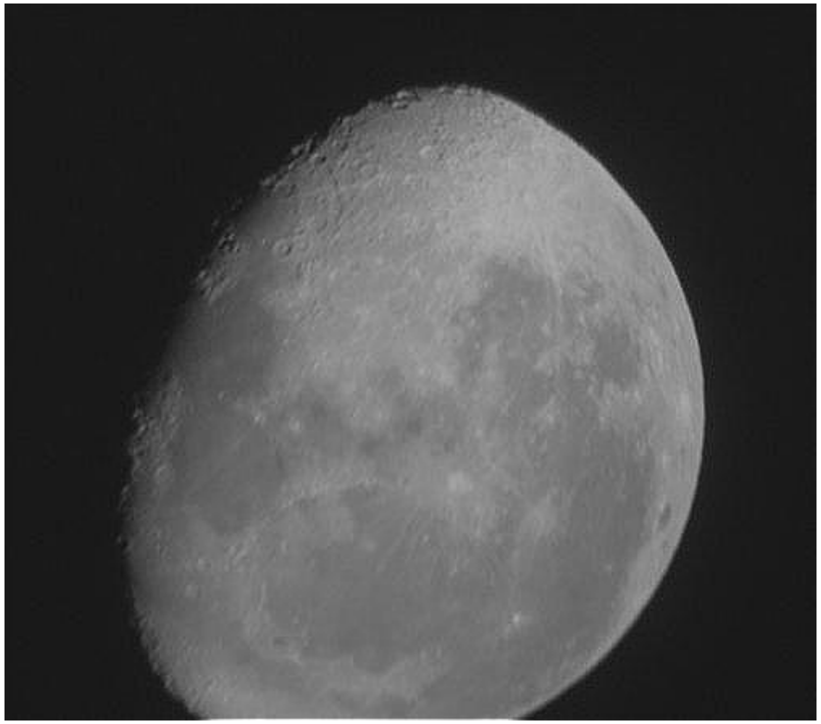
$$\underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{g} - \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{\nabla^2 g} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- So, to sharpen a blurred image, we convolve it with a 3x3 mask whose weights are given above.

Example unsharp masking



Original moon image



Unsharp mask applied

Summary

- Image restoration
  - Degradation model must be known.
  - The degradation is reversed using this known model.
- Image enhancement
  - Degradation model is not known.
  - Goal is to (1) clean up noise or (2) make image visually more pleasing
- Methods:
  - Spatial filtering methods
  - Frequency domain methods
- Point operations
  - Contrast stretching
  - Histogram modification and histogram equalization
- Spatial neighborhood operations
  - Averaging filters
  - Median filters (nonlinear)
  - Unsharp masking