Introduction to Digital Image Processing

— Spatial Filters

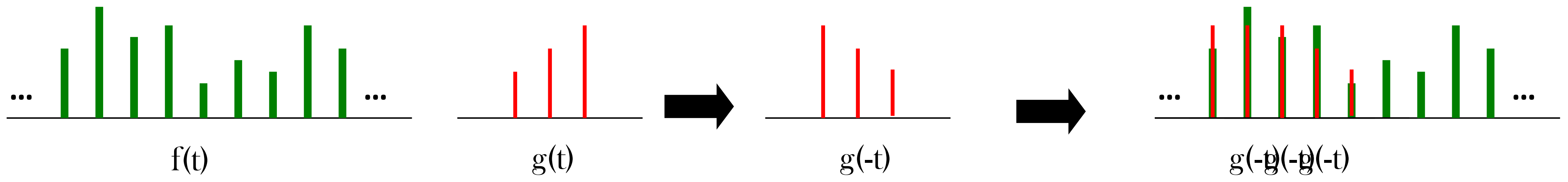
Xiaohui Yuan

Department of Computer Science and Engineering
University of North Texas
xiaohui.yuan@unt.edu

Spatial Filters

- Spatial filtering is a process that extracts features/patterns from an image by computing the response to a filter.
    - A filter is a matrix consisting of numbers that model a spatial pattern.
- It is used in many tasks such as
    - Feature extraction (points, lines, textures, etc.)
    - Image enhancement
    - Noise removal
- Filtering can be achieved in the spatial and frequency domains
    - Some filtering processes can be easily and effectively performed in the spatial or frequency domain or both.
- The spatial filtering relies on an operation called convolution

Convolution



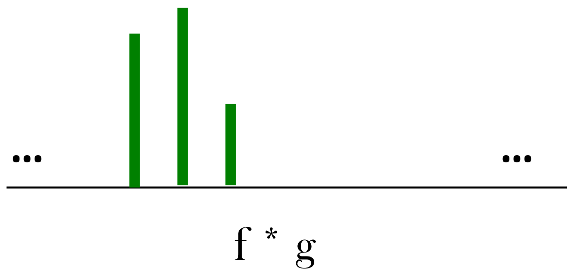
- The convolution of two functions *f* and *g* is defined as the integral of their product:
- Convolution is computed as follows

$$(f * g)(t) = \int f(\tau)\, g(t-\tau)\, d\tau$$

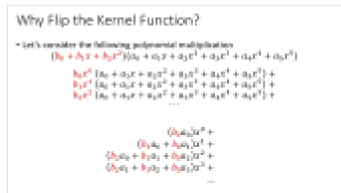
- The discrete version of the convolution is

$$(f * g)(m) = \sum_{n} f(n)\, g(m-n)$$

- Properties:
  - **Associativity** $(f * g) * h = f * (g * h)$
  - **Distributivity** $f * (g + h) = f * g + f * h$



In two-dimensional cases, function g(t) is **rotated/flipped** 180 degrees.

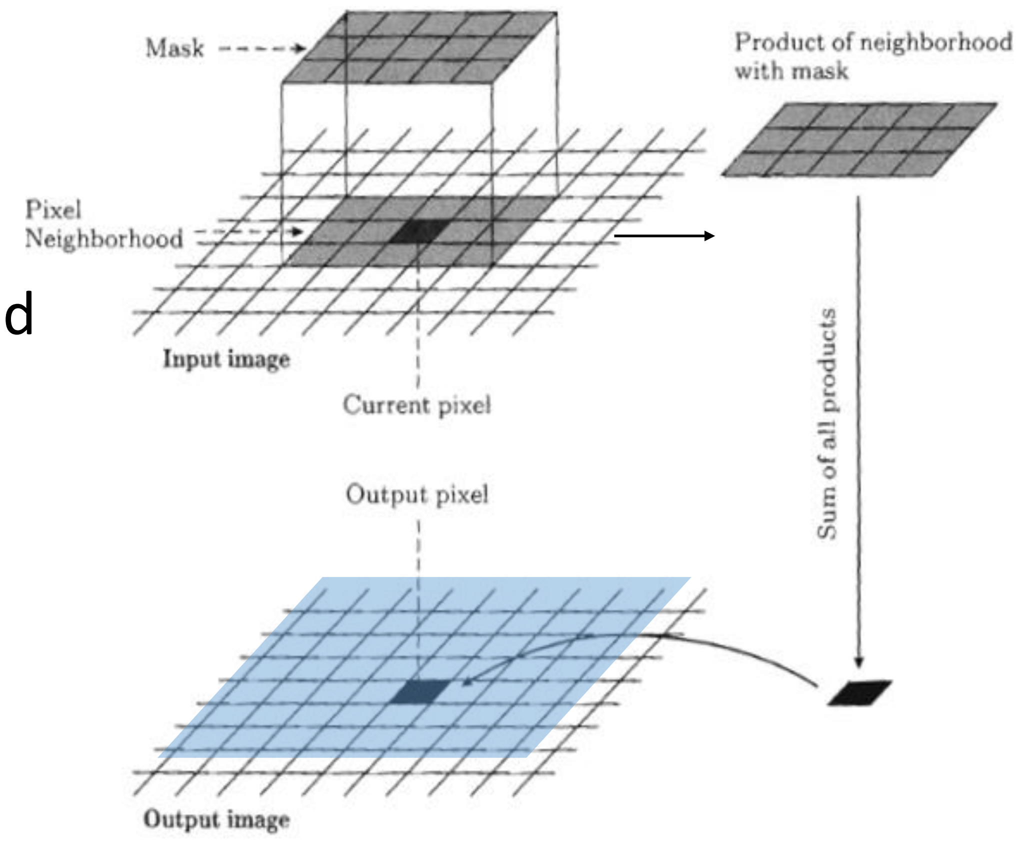

Why Flip the Kernel Function?

- Let's consider the following polynomial multiplication

[illegible]

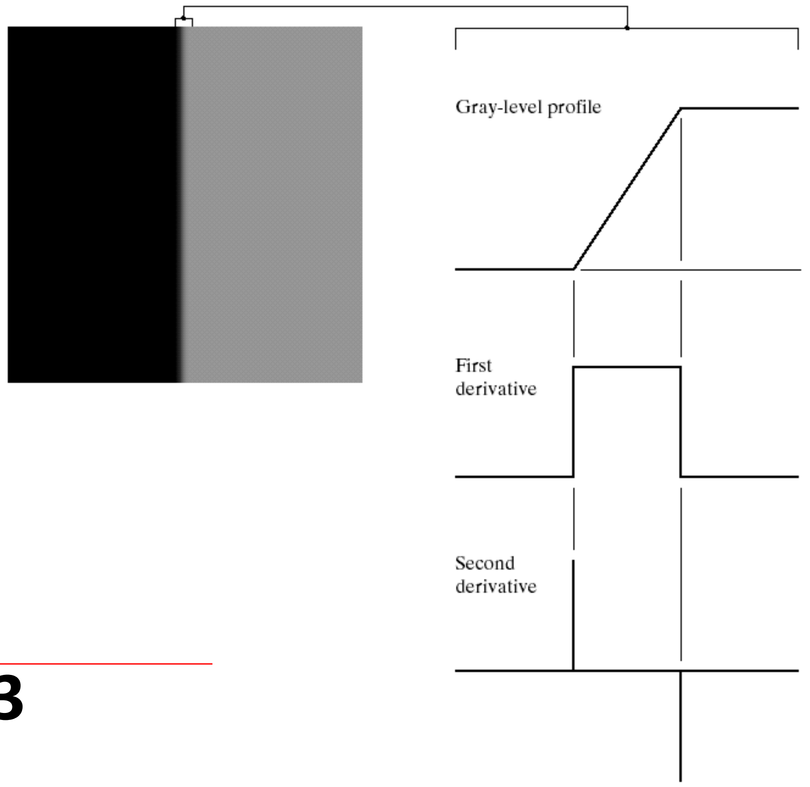
Spatial Filtering

- Spatial filtering is similar to convolution.
- It applies a matrix (a.k.a. filter, mask, kernel) to an image by computing the sum of element-wise products.
  - We slide a filter across the image (from left to right with a step size of one column at a time) and compute the dot product then move to the next row.
  - Repeat the computation until we complete the bottom row.
- The filters often do not flip partly because they are mostly symmetric
  - If not, we take that into consideration at the design phase



Filter for Edge Detection

- An edge is a set of pixels that lie on the boundary of two regions with different colors.
  - It describes the color discontinuity.
- An ideal edge has sharp changes in gray value.
  - However, a more commonly seen edge has a ramp-like profile
- .
- **Sobel edge detector** models the grayscale changes in a **3 by 3** neighborhood
- A larger value 2 (or -2) in the center column (or row) is used to emphasize the center point by giving a greater weight
- The coefficients in each filter sum to zero, such that a response of zero is produced in homogeneous regions



+1	+2	+1
0	0	0
-1	−2	−1

Extract Horizontal Lines

−1	0	+1
−2	0	+2
−1	0	+1

Extract Vertical Lines

Point Detectors

- What are the differences among these filters?

-1	-1	-1
-1	8	-1
-1	-1	-1

(a)

1	1	1
1	-8	1
1	1	1

(b)

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	24	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

(c)

1	1	1	1	1
1	-3	-3	-3	1
1	-3	8	-3	1
1	-3	-3	-3	1
1	1	1	1	1

(d)

Laplacian of Gaussian (LoG) Filter

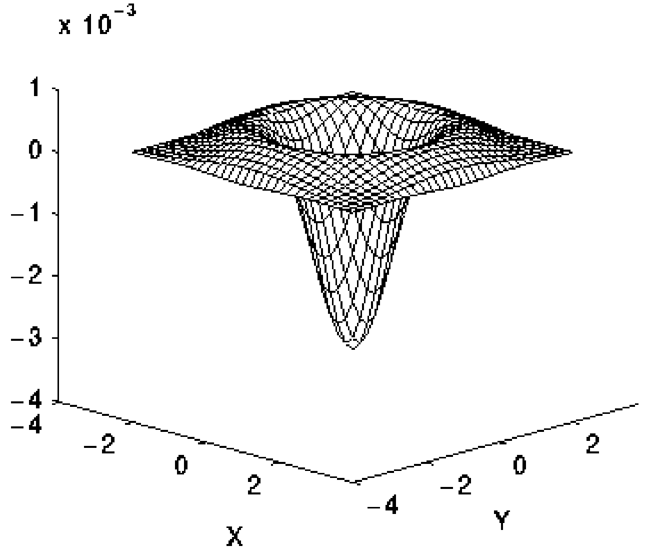
- **Laplacian filters** compute the second derivative of an image

$$L(x,y) = \nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

- To include smoothing, we combine the Laplacian and Gaussian functions
  - Laplacian of Gaussian, also known as "*Mexican Hat*" filter, is

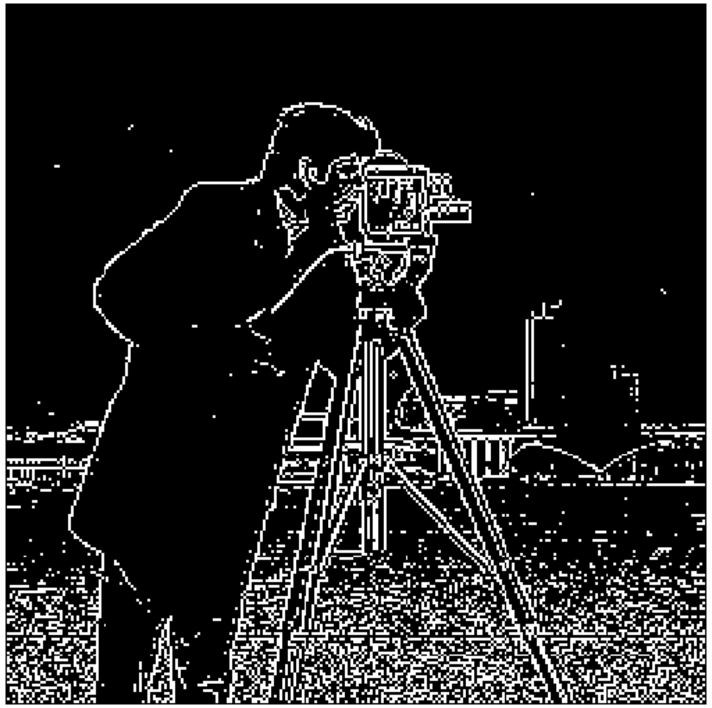
$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- In the vicinity of an intensity change, the LoG response is positive on the darker side of the edge, and negative on the lighter side.
- For a sharp edge between two regions of uniform but different intensities, the LoG response is:
  - Positive on one side of the edge.
  - Negative on the other side of the edge.
  - Zero at the edge pixel.
  - Zero at far distance from the edge.

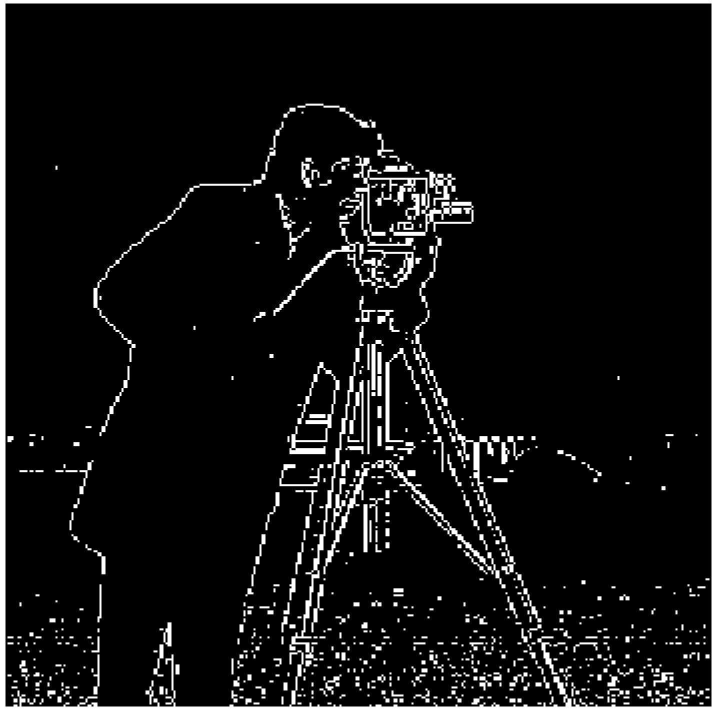


$\sigma = 1.4$

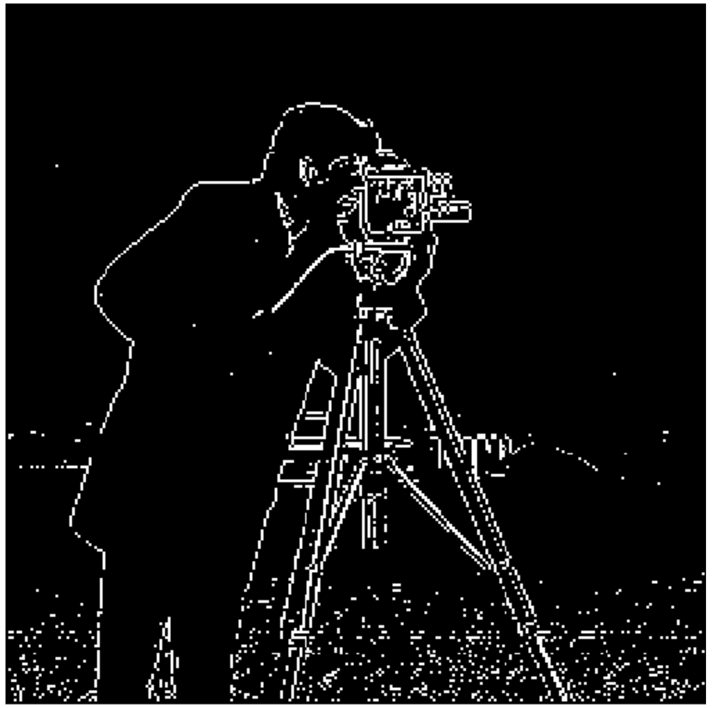
0	0	3	2	2	2	3	0	0
0	2	3	5	5	5	3	2	0
3	3	5	3	0	3	5	3	3
2	5	3	-12	-23	-12	3	5	2
2	5	0	-23	-40	-23	0	5	2
2	5	3	-12	-23	-12	3	5	2
3	3	5	3	0	3	5	3	3
0	2	3	5	5	5	3	2	0
0	0	3	2	2	2	3	0	0



$3x3, \sigma^2 = 0.2$



$7x7, \sigma^2 = 0.2$



$13x13, \sigma^2 = 0.15$

Canny Edge Detector

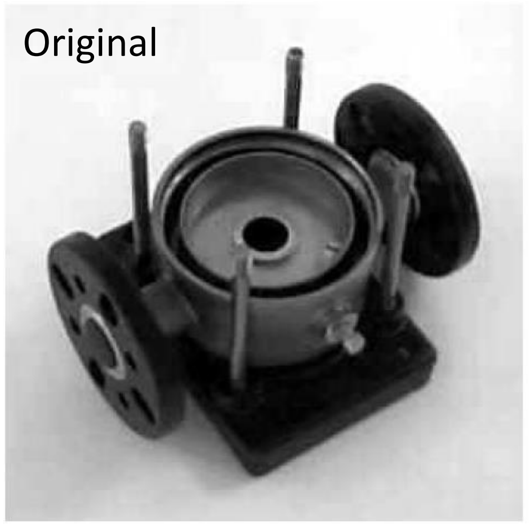
- Canny edge detector aims at finding the <u>strongest</u> edges as well as the <u>connected</u> weak ones.
- The algorithm consists of five steps:
  1. **Smoothing**: Blur the image to suppress noise.
  2. **Finding gradients**: The edges should be marked where the gradients of the image have large magnitudes.
  3. **Non-maximum suppression**: Only local maxima should be marked as potential edges.
  4. **Double thresholding**: Potential edges are determined by two different thresholds.
  5. **Edge tracking**: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

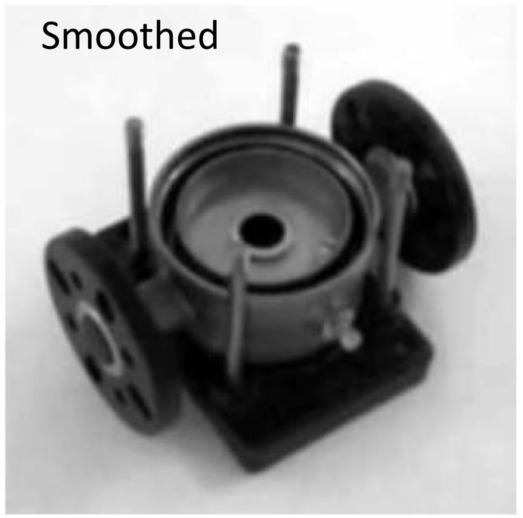
Steps 1 Smooth the image

- To prevent that noise is mistaken for edges, the image is first smoothed by a Gaussian filter.
- The Gaussian filter has a standard deviation of 1.4.

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Why 1/159?





Steps 2 Find Gradients

- The gradients at each pixel in the smoothed image are calculated using the Sobel operators.
- The gradient magnitudes (a.k.a. edge strengths) is computed using Euclidean or Manhattan distance

G_x and G_y are the gradients in the x- and y-directions, respectively.

Euclidean Distance: $G = \sqrt{G_x^2 + G_y^2}$

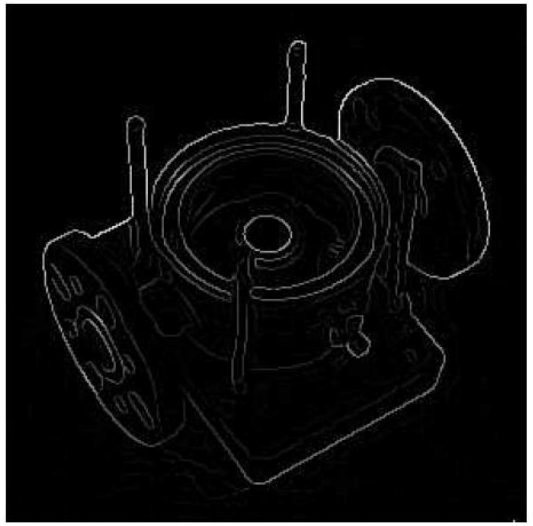
Manhattan Distance: $G = |\mathrm{G_x}| + |\mathrm{G_y}|$

- The gradient map clearly shows the strength of edges but does not tell the direction.
- The direction of the edges is computed as follows:

$$\theta = \arctan(\frac{|G_y|}{|G_x|})$$

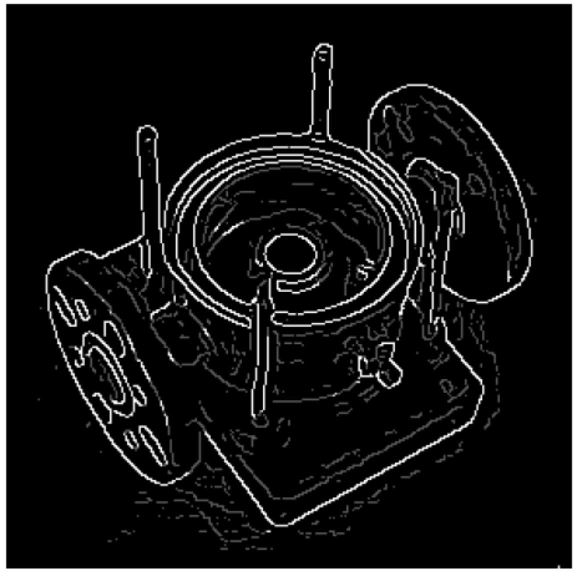
Step 3 Non-maximum Suppression

- This step is to convert the “blurred” edges in the image of the gradient magnitudes to “sharp” edges by preserving only the local maxima.
  1. Round the gradient direction to the nearest 45° angle.
  2. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions. For instance. if the gradient direction is north (90°), compare with the pixels to the north and south.
  3. If the edge strength of the current pixel is the largest; preserve its value; otherwise, suppress it to zero.
- The example shows a gradient map with almost all pixels having gradient directions pointing north.
  - They are therefore compared with the pixels above and below.
  - The pixels that turn out to be maximal in this comparison keep their value, and the rest are suppressed to zero.



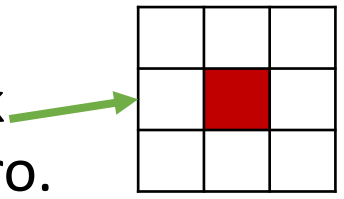
Step 4 Double Thresholding

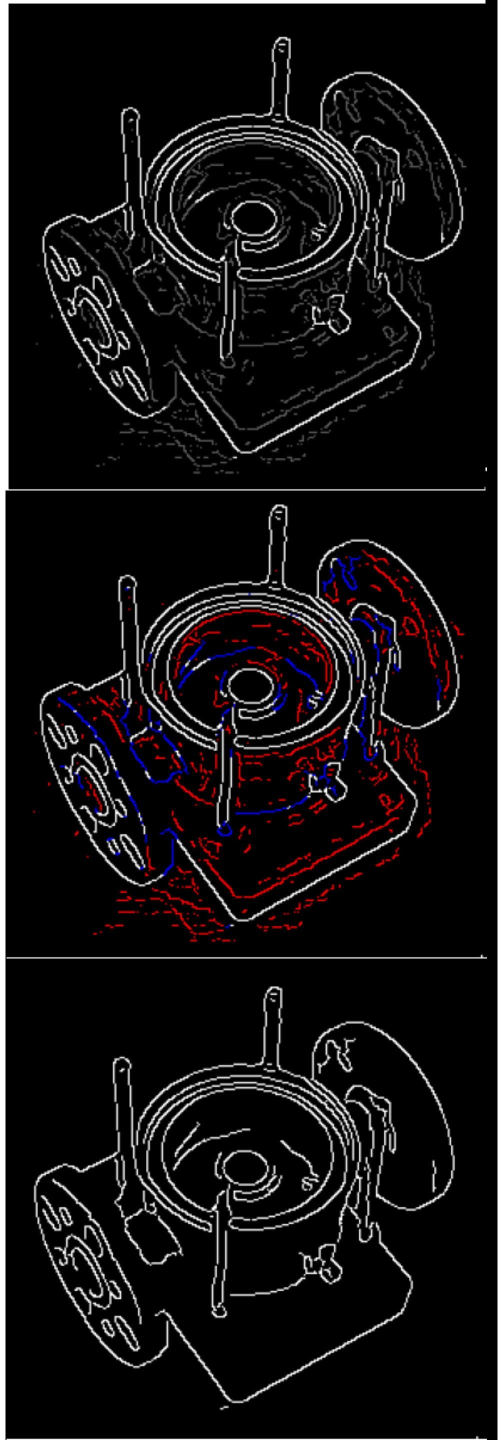
- Many of the edges are probably true edges, but some may be caused by noise or color variations due to rough surfaces.
- A simple way to differentiate edge and noise is to use a threshold so that only edges stronger than a certain value would be preserved.
- Double thresholding is used and for each pixel
  - stronger than the high threshold are marked as **strong**;
  - weaker than the low threshold are **suppressed**; and
  - between the two thresholds are marked as **weak**.
- The edge map is achieved by applying a double thresholding with thresholds of 20 and 80



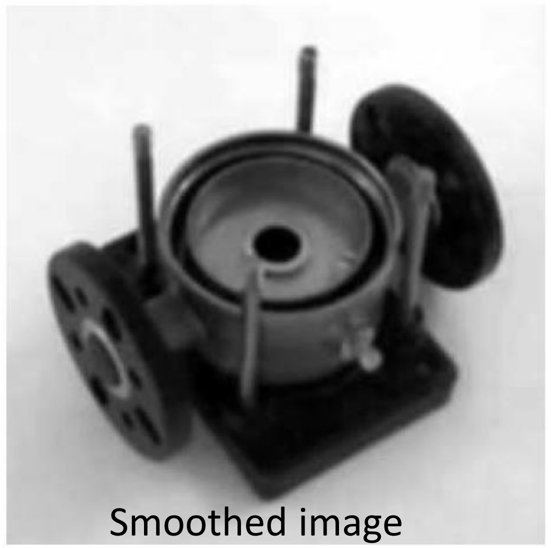
Step 5 Edge Tracking

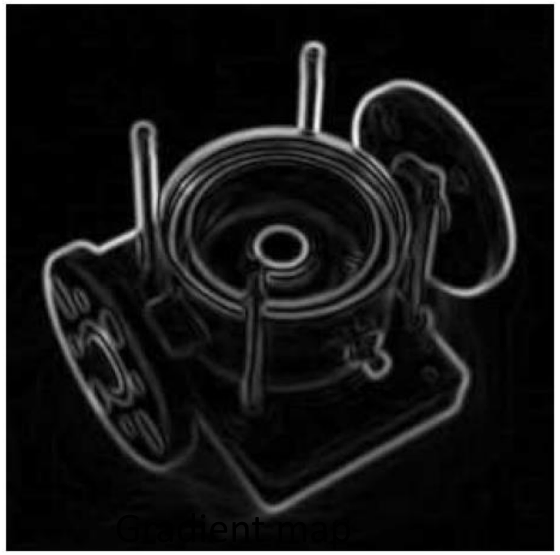
- **Strong edges** are included in the final edge image.
- **Weak edges** are included if and only if they are connected to strong edges.
  - The rationale is that noise and other small variations are unlikely to result in a strong edge (with proper adjustment of the threshold levels).
  - The weak edges can either be due to true edges or noise/color variations.
  - The latter is probably distributed independently of the edges.
  - Weak edges due to true edges are much more likely to be connected to strong edges.
- If there exists at least **one (strong) edge pixel** in the **8-connected neighborhood** of a weak edge pixel, the weak edge pixel is preserved; otherwise, it is suppressed to zero.

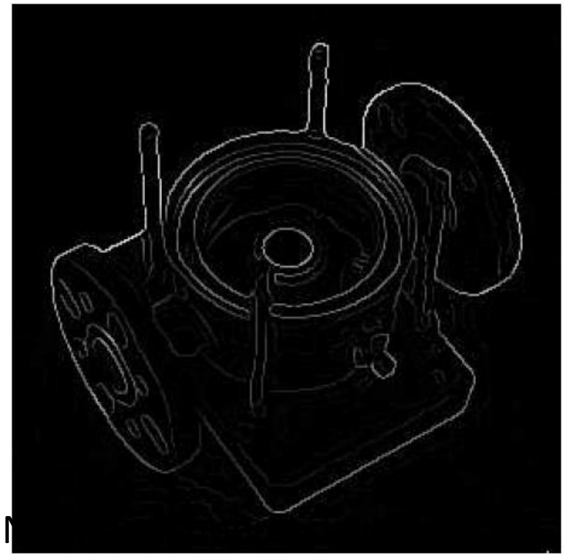


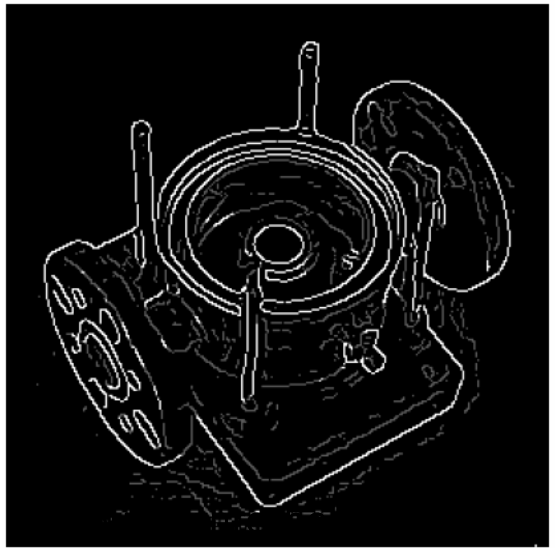


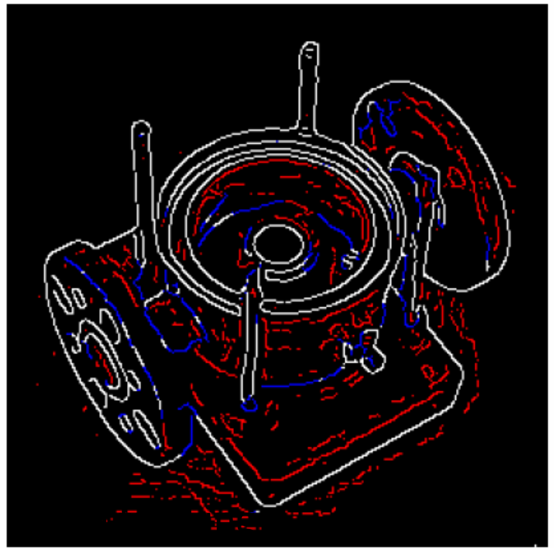
Canny Edge Detection Results


Smoothed image

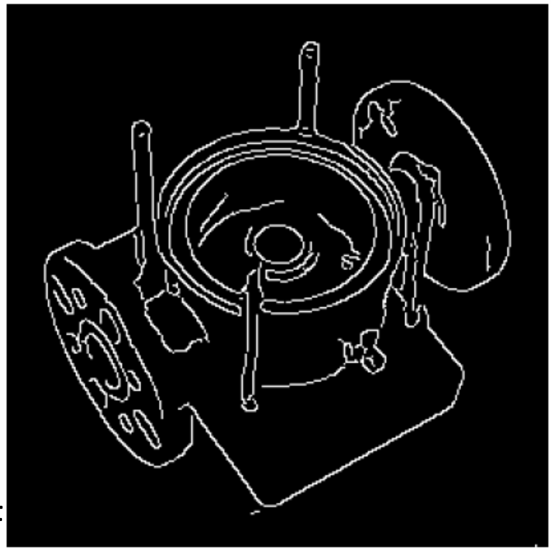

Gradient map


N



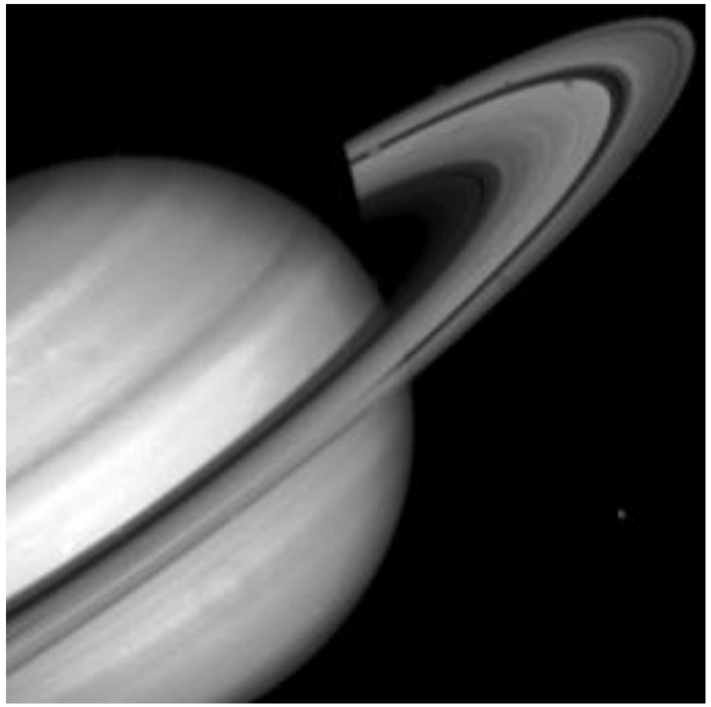


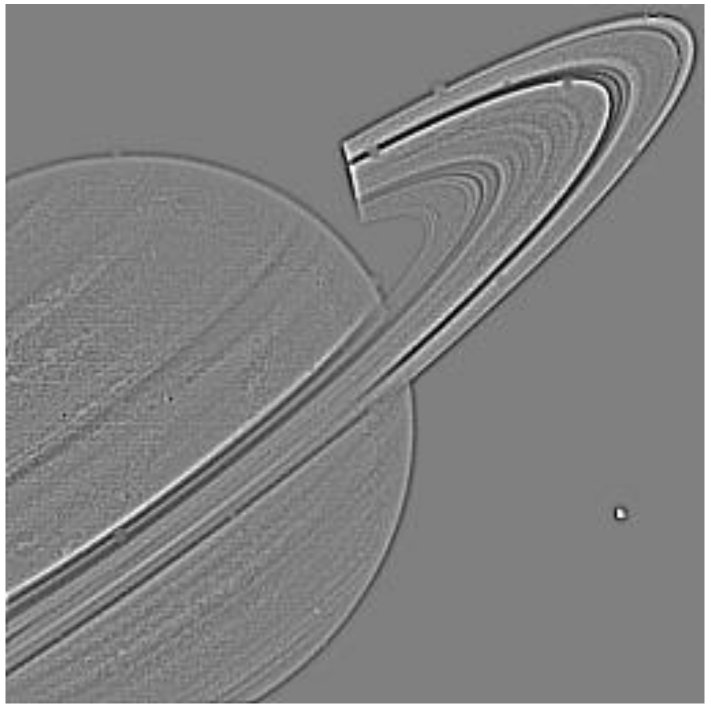
F



Edge Enhancement

- Spatial filtering can be used to make edges sharper in an image.
  - This process is called edge enhancement.


Input image


Edge map

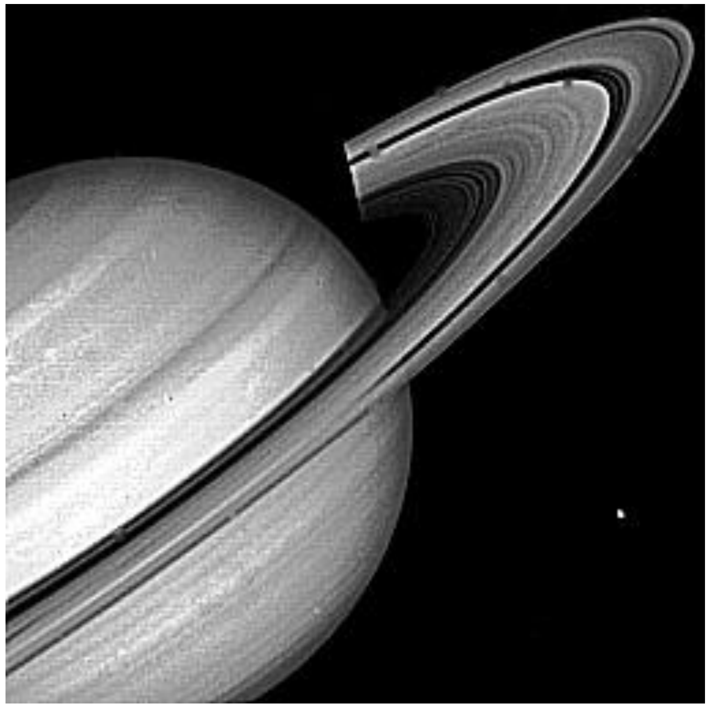
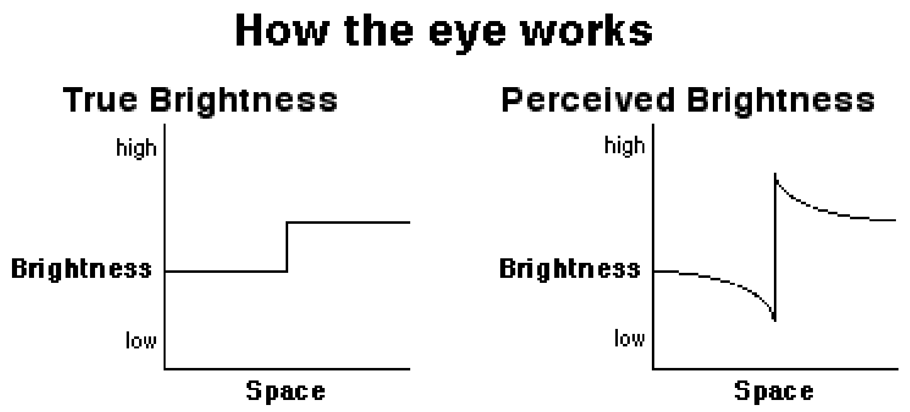
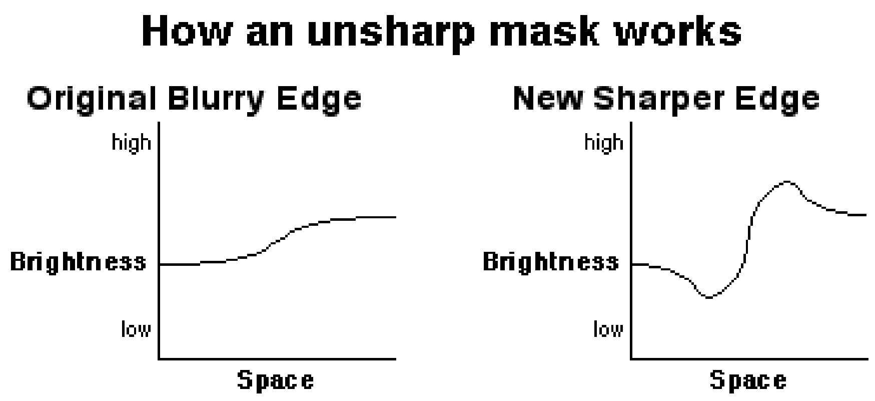
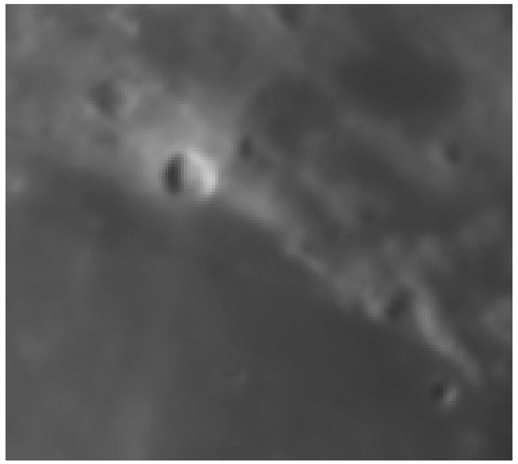

Edge enhanced image

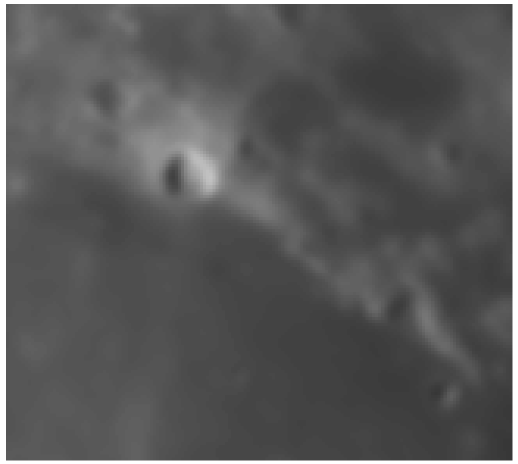
Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

How Our Eye Perceive Edges






Raw Image


Smoothed

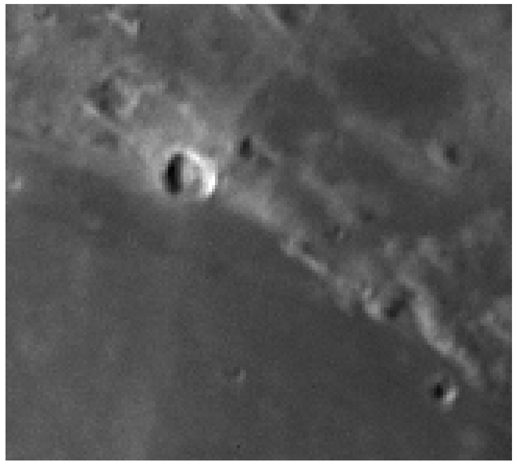

Sharpened

Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Unsharp Mask

- The idea of the unsharp mask filter is to subtract an unsharp version of the image from the original:

$$f_s(x,y) = f(x,y) - \bar{f}(x,y)$$

- An unsharp mask cannot create additional detail
  - It enhances the appearance by increasing small-scale acutance
- A more general version of the unsharp mask is called high-boost filtering
  - It subtracts a blurred version from a scaled image

$$f_s(x,y) = kf(x,y) - \bar{f}(x,y)$$

How do we achieve an unsharp mask via convolution?

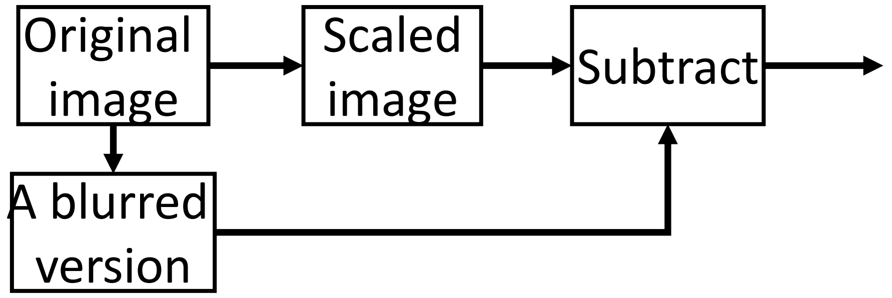


Image Noise

- Noise types
  - Independent of spatial location
    - Gaussian noise
    - Salt and pepper noise
  - Spatially dependent
    - Periodic noise
- To model image noise, we adopt the following assumption
  - Image noise is independent of the underlying image (i.e., clean image)
  - The noisy image can be represented by an additive model

    $$I(x,y)=f(o(x,y))+n(x,y)$$

    where $I(x,y)$ is the acquired image, $\mathrm{o}(x,y)$ is the object to be imaged, $n(x,y)$ is the additive noise, and function $f(\cdot)$ is the transformation function.

Gaussian Noise

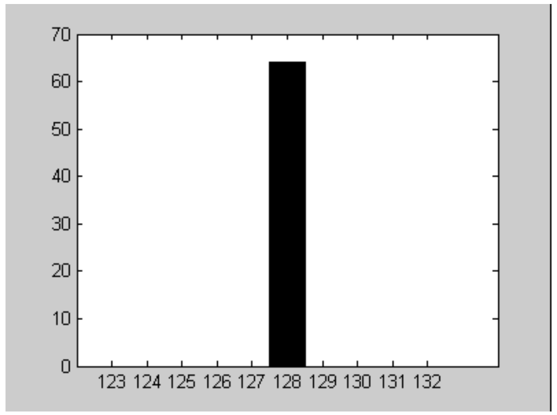
- Each pixel in an image is disturbed by a Gaussian random variable with zero mean and variance σ^2

X: noise-free image, Y: noisy image

$$Y(i,j) = X(i,j) + N(i,j),$$
$$N(i,j) \sim N(0,\sigma^2), \quad 1 \le i \le H, \quad 1 \le j \le W$$

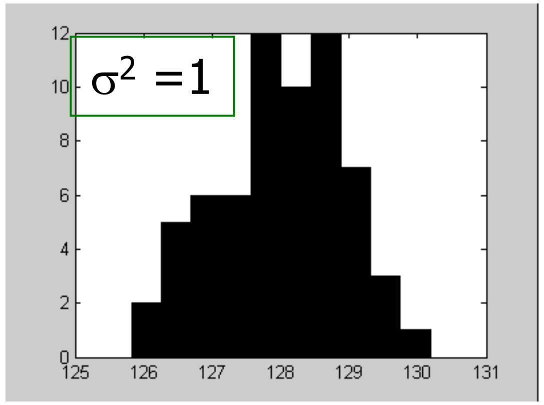
- Every pixel in the image is contaminated with some amount of noise.

128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128
128	128	128	128	128	128	128	128



X

128	128	129	127	129	126	126	128
126	128	128	129	129	128	128	127
128	128	128	129	129	127	127	128
128	129	127	126	129	129	129	128
127	127	128	127	129	127	129	128
129	130	127	129	127	129	130	128
129	128	129	128	128	128	129	129
128	128	130	129	128	127	127	126



Y

Average Filters

- **Homogeneous average filters**

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\qquad \frac{1}{16}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

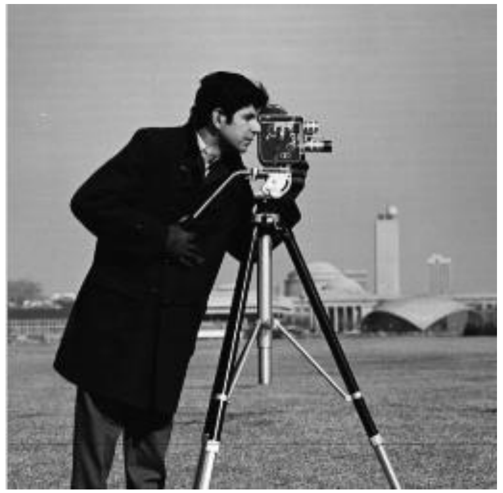
- **Gaussian average filters** are weighted average filters
- The value of each element follows a 2D isotropic Gaussian function

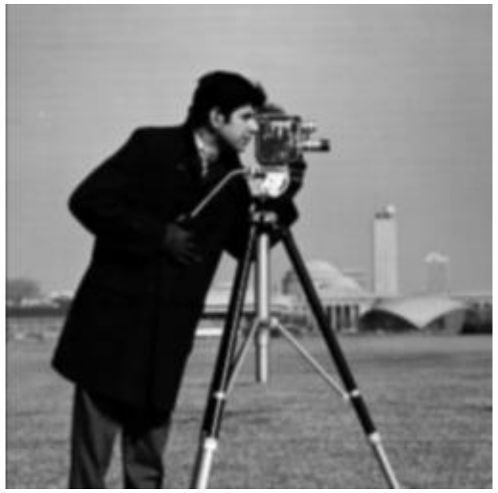
$$\frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\qquad \frac{1}{273}\begin{bmatrix}1&4&7&4&1\\4&16&26&16&4\\7&26&41&26&7\\4&16&26&16&4\\1&4&7&4&1\end{bmatrix}$$

Gaussian Filters

- Gaussian filters are mathematically very well-behaved
  - The Fourier transform of a Gaussian filter is also Gaussian
- They are rotationally symmetric
  - Good starting points for many edge detection algorithms
- They are separable and can be easily implemented in multi-dimensional applications
- They are often used in constructing other filters, e.g., Gabor filters

Original image





Filter Result (7 x 7)

Salt and Pepper Noise

- Each pixel in an image has the probability of p/2 (0≤ p ≤ 1) being contaminated to be either white (salt) or black (pepper).

$$Y(i,j) = \begin{cases} 255 & \text{with probability of p/2} \\ 0 & \text{with probability of p/2} \\ X(i,j) & \text{with probability of 1-p} \end{cases}$$

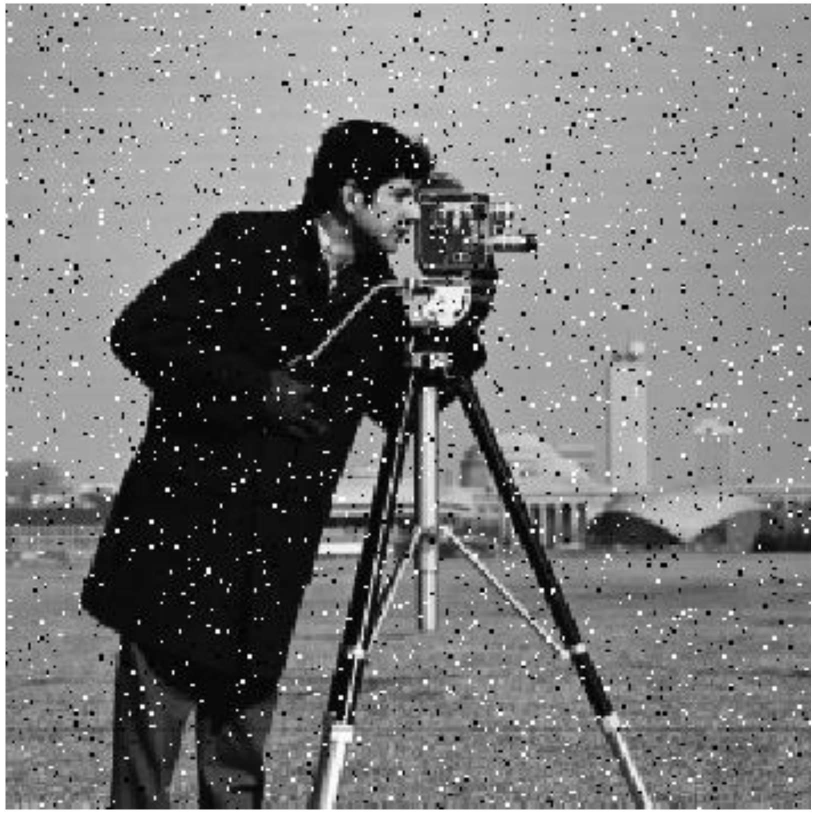
$$1 \le i \le H, 1 \le j \le W$$

noisy pixels

clean pixels

X: noise-free image, Y: noisy image

- In some applications, noisy pixels are not simply black or white, which makes the noise removal problem more difficult.



Median Filter

- Problem with Averaging Filter
  - Blurring edges and details in an image
  - Not effective for impulse noise (Salt-and-pepper)
- Median filter
  - Taking the median value instead of the average
  - Algorithm: sort all the pixels in increasing order and take the middle value
    - The window shape does not need to be a square
    - Special shapes can preserve line structures
- Order-statistics filter
  - Instead of taking the mean, rank all pixel values in the window, and take the n-th order value.
  - For example, max or min

225	225	225	226	226	226	226	226
225	225	255	226	226	226	225	226
226	226	225	226	**0**	226	226	255
255	226	225	**0**	226	226	226	226
225	255	**0**	225	226	226	226	255
255	225	224	226	226	0	225	226
226	225	225	226	255	226	226	228
226	226	225	226	226	226	226	226

Y

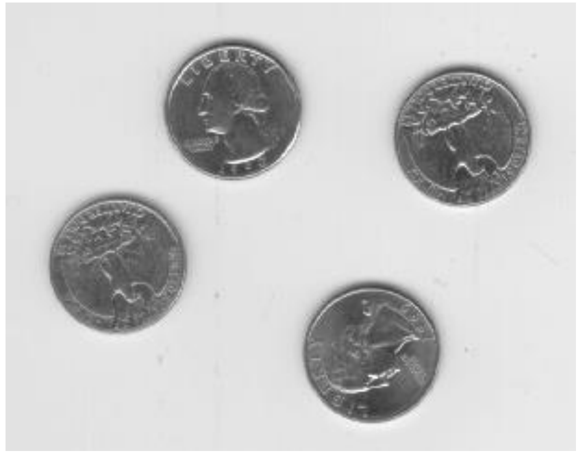
Sorted: [0, 0, 0, 225, **225**, 225, 226, 226, 226]

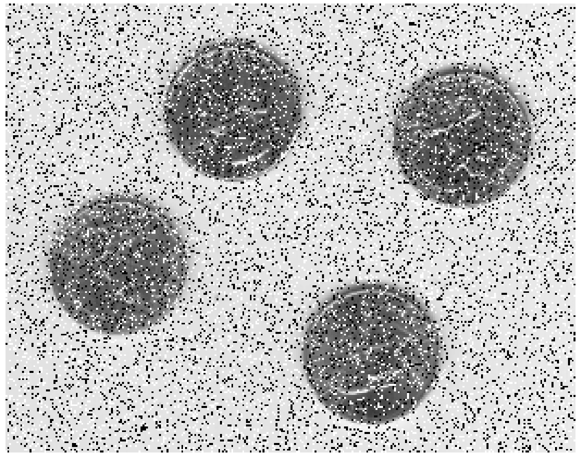
0	225	225	226	226	226	226	226
225	225	255	226	226	226	226	226
225	226	226	226	0	226	226	226
255	226	225	**225**	226	226	226	226
225	225	0	225	226	226	226	226
255	225	225	226	226	226	226	226
225	225	225	226	255	226	226	226
226	226	226	226	226	226	226	226

$\hat{X}$

Noise Removal Examples

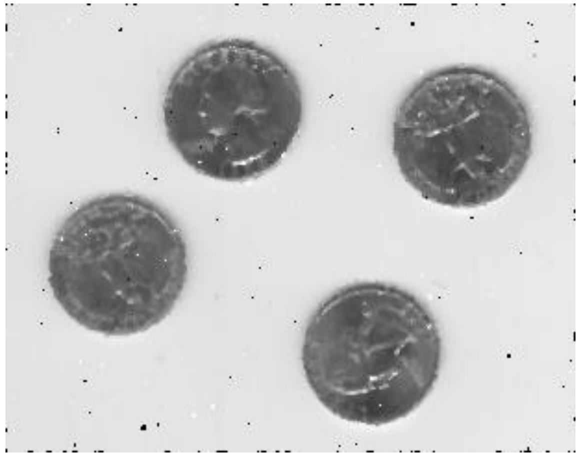
clean

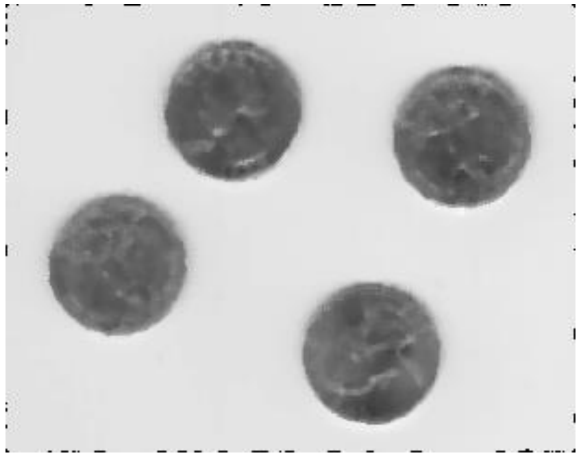




noisy (p=0.2)

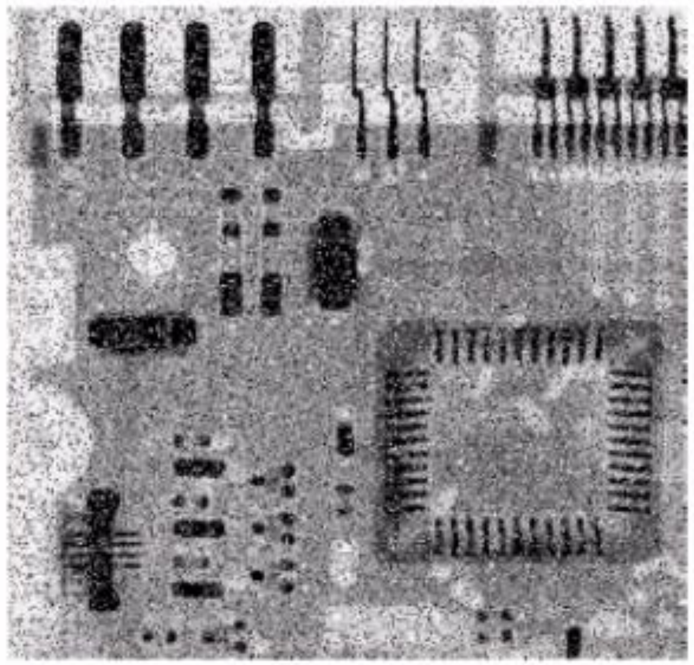
3-by-3 window



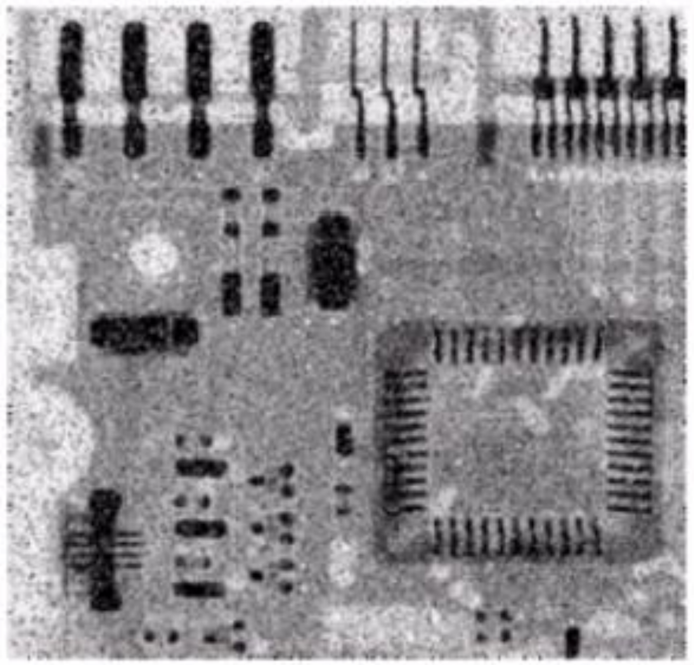


5-by-5 window

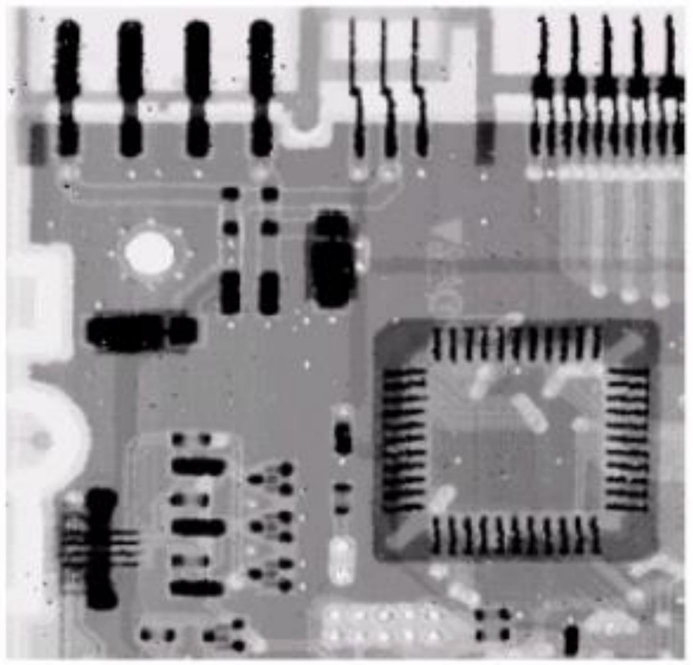
Comparison of the Results of Average and Median Filters



Original image



Noise reduction with a 3 by 3 average filter



Noise reduction with a 3 by 3 median filter

- What is good about median operation?
  - Salt and pepper noise appears as black and white dots, taking median effectively suppresses this type of noise
- What is bad about median operation?
  - Noticeable edge distortion

Periodic Noise

- Cause: electrical or electromechanical interference during image acquisition.
- Characteristics
  - Spatially dependent
  - Easy to observe the distorted components in the frequency domain
- Denoising method
  - Suppressing noise components in the **frequency** domain