

Machine Vision

Lecture Set – 05

Image Filtering

Huei-Yung Lin

Image Histogram

- A **histogram** lists the number of image pixels for each value

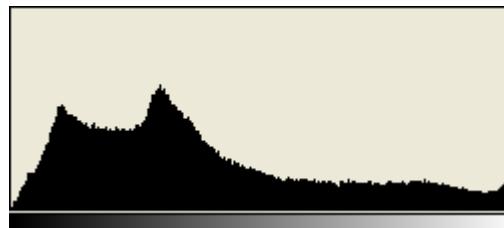
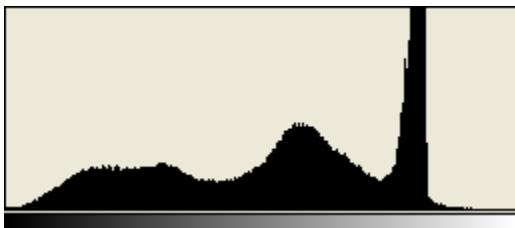


Image Histogram

- Operations based on image histograms are carried in two stages:
 - The accumulation of the histogram – give global knowledge of image
 - Its analysis is followed by an enhancement operation based on the **shape of histogram**
 - enable a suitable transformation

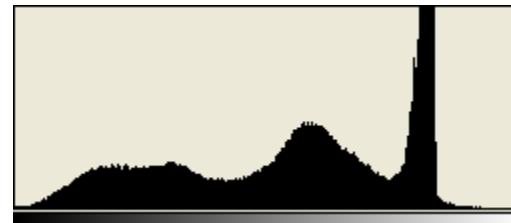
Histogram Processing

- Histogram of a digital image with gray levels in the range $[0, L - 1]$ is a discrete function

$$h(r_k) = n_k$$

where

- r_k : the k^{th} gray level
- n_k : the number of pixels in the image having gray level r_k
- $h(r_k)$: histogram of a digital image with gray levels r_k



Normalized Histogram

- Dividing each of histogram at gray level r_k by the total number of pixels in the image, n

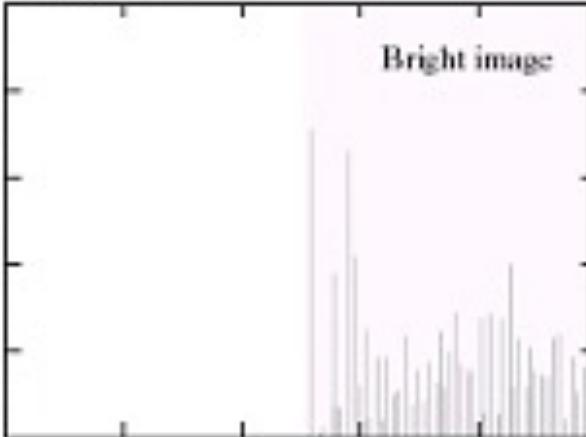
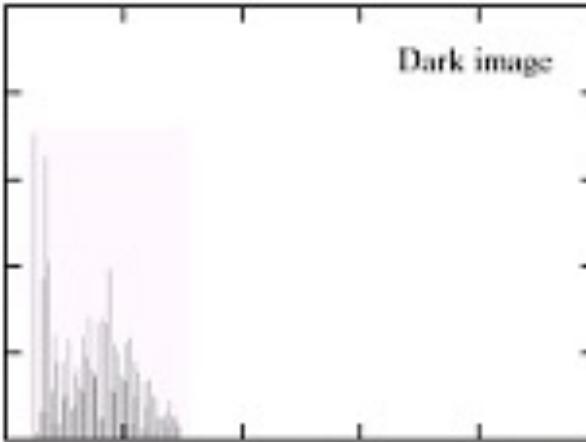
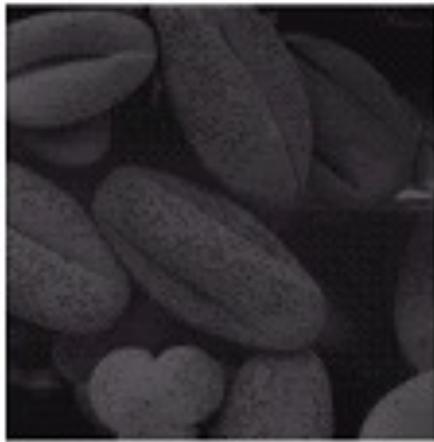
$$p(r_k) = n_k / n$$

- For $k = 0, 1, \dots, L - 1$
 - $p(r_k)$ gives an estimate of the probability of occurrence of gray level r_k
 - The sum of all components of a normalized histogram is equal to 1

Histogram Processing

- Basic for numerous spatial domain processing techniques
- Used effectively for image enhancement
- Information inherent in histograms is also useful in **image compression and segmentation**

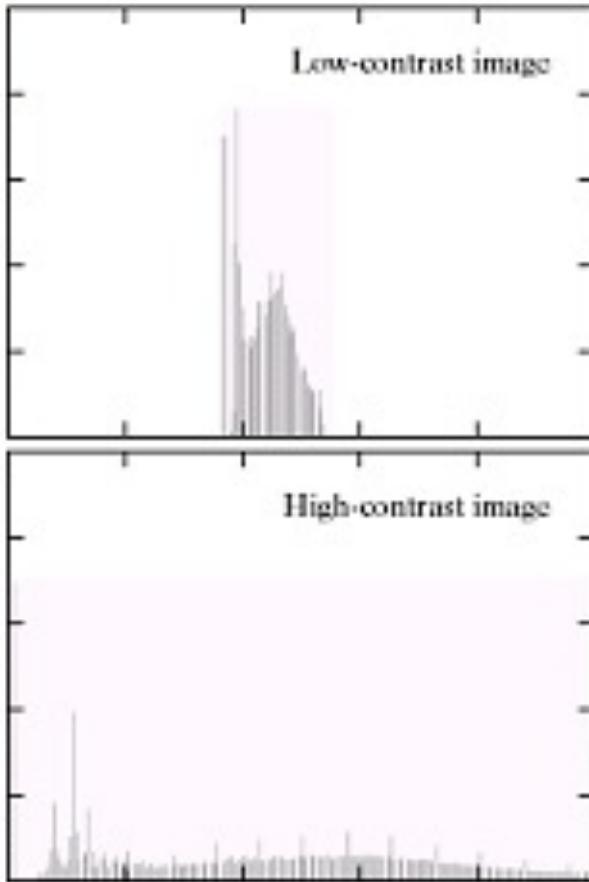
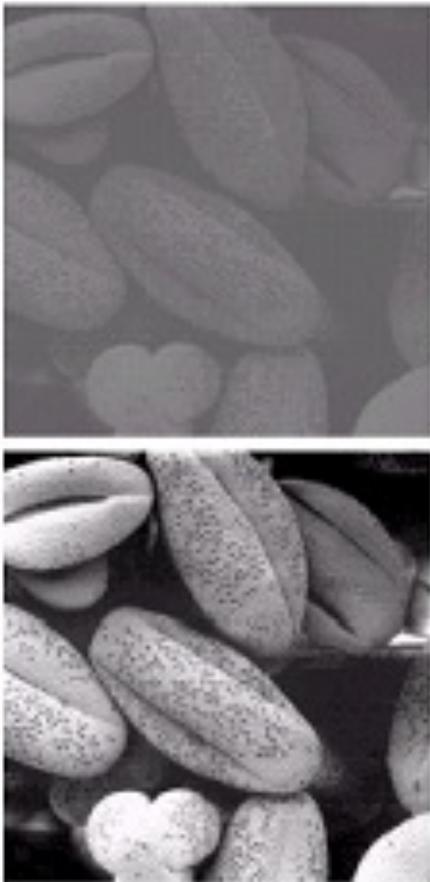
Example



- Dark image
 - Components of histogram are concentrated on the low side of the gray scale

- Bright image
 - Components of histogram are concentrated on the high side of the gray scale

Example



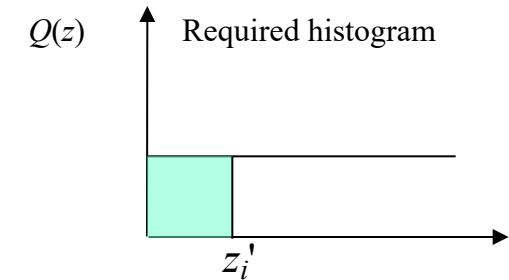
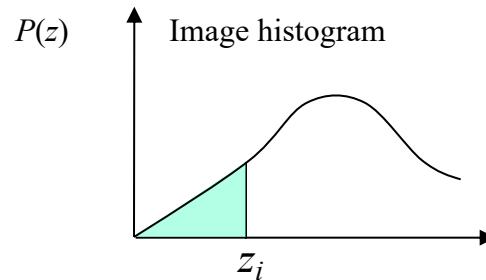
- Low-contrast image
 - histogram is narrow and centered toward the middle of the gray scale
- High-contrast image
 - histogram covers broad range of the gray scale and the distribution of pixels is not too far from uniform, with very few vertical lines being much higher than others

4/13/2023

- Midterm Exam next Thursday (4/20)
- No class next Tuesday (4/18)
- Homework #3 will be given later at 11:40.

Histogram Equalization

- Histogram equalization can be seen as an extension of the interactive contrast stretching
- To derive a transformation $T(z)$ such that the original histogram is transformed to an equi-probable histogram



$$\int_0^{z_i} P(z) dz = \int_0^{z_i'} Q(z) dz = MN \frac{z_i'}{z_{\max}}$$

$$z_i' = T(z_i) = \frac{z_{\max}}{MN} \int_0^{z_i} P(z) dz$$

$$z_i' = \frac{z_{\max}}{MN} \sum_{i=0}^{z_i} H(z)$$

Histogram Equalization

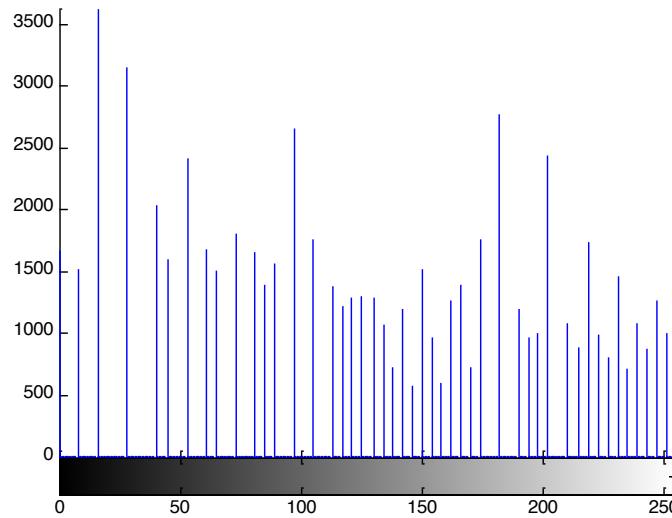
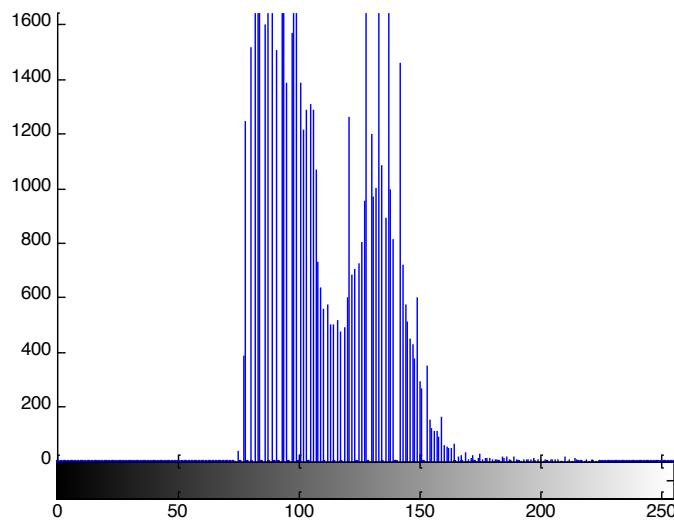


Image Noise

- Noise in computer vision:
 - Any entity, in images, data or intermediate results, that is not interesting for the purpose of the main computation

- Image noise in different cases
 - For image processing algorithms:
 - Spurious fluctuation of pixel values introduced by the image acquisition system
 - For algorithms with numerical input:
 - Inaccuracies of input, computer's precision, round-off error
 - For grouping algorithms:
 - Contours which do not belong to any meaningful object

Image Noise

- Images are corrupted by “noise” mostly during acquisition
 - Signal processing techniques can be used to model and remove this noise
 - Different types of noise are countered by different techniques, depending on the noise’s nature and characteristics
- **Image Restoration:** removal of noise
 - Salt & pepper noise
 - Gaussian noise
 - Impulsive noise

Acquisition Noise

- The image values are not those expected, they are corrupted during image acquisition
- Two images are never **exactly** the same!
- Noise is characterized by their statistic behavior
 - Mean & standard deviation of n images
- Pixel values are not completely independent of each other
 - Some cross-talking occurs between adjacent photo-sensors
 - Auto-covariance (image processing)

Additive Noise & Noise Amount

- In the additive noise model the signal is corrupted with random fluctuations

$$\hat{I}(i, j) = I(i, j) + n(i, j)$$

- The amount of noise is measured by means of σ_n , the standard deviation of $n(i, j)$
- Signal-to-noise ratio (SNR): $SNR = \frac{\sigma_s}{\sigma_n}$
 - σ_s : s.d. of the signal
- In decibel: $SNR_{dB} = 10 \log_{10} \frac{\sigma_s}{\sigma_n}$
- Quantization noise
- Multiplicative noise: $\hat{I}(i, j) = n(i, j) \cdot I(i, j)$

Gaussian Noise

- Without any information, noise is often modeled as **zero mean Gaussian distribution**
- This model predicts that noise values are distributed symmetrically around zero, and pixel values $\hat{I}(i, j)$ around their true values $I(i, j)$



Impulsive Noise

- Also known as spot or peak noise
- Usually introduced by the acquisition process
- Alters random pixels, making their values very different from the true value and neighboring pixels
- Appears as a sprinkle of dark and light spots
- It can also be caused by
 - Transmission errors
 - Faulty elements in the CCD array
 - External noise corrupting the A/D conversion
- Salt and pepper noise

Salt & Pepper Noise

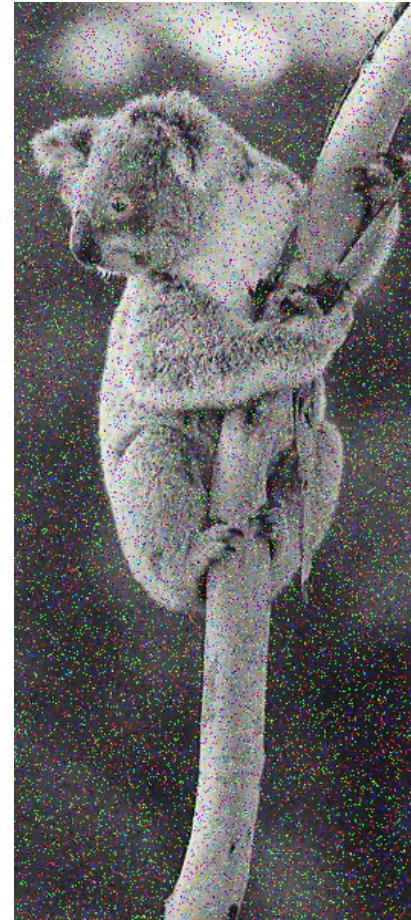
- Salt and pepper noise can model errors introduced by the acquisition process
- It is also used to simulate impulse noise in **synthetic images**
- x and y are random variables in the range $(0,1)$ and l is a constant

$$\hat{I}(m,n) = \begin{cases} I(m,n) & x < l \\ i_{\min} + y(i_{\max} - i_{\min}) & x \geq l \end{cases}$$

- i_{\max} and i_{\min} indicate how severe the noise is

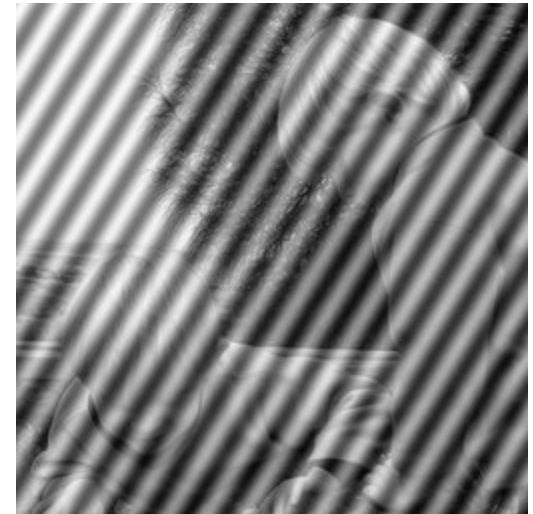
Example

■ Salt & Pepper Noise



Example

- Speckle noise (multiplicative noise)
- Periodic noise



Noise Filtering

- Noise Filtering
 - Given an image I corrupted by noise n , attenuate n as much as possible without altering I significantly
- Image derivatives are very sensitive to noise and should be avoided if possible
- Linear filtering:
 - Convolving the image with a constant matrix (called mask or kernel)

Image Filtering

■ Local neighborhoods

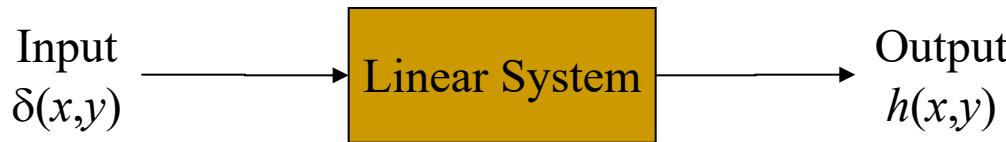
- Hard to tell anything from a single pixel
 - Example: you see a reddish pixel
 - Is this the object's color? Illumination? Noise?
- The next step in order of complexity is to look at local neighborhood of a pixel

■ Linear Filters

- Given an image $I(x,y)$ generate a new image $O(x,y)$:
 - For each pixel (x,y) , $O(x,y)$ is a linear combination of pixels in the neighborhood of $I(x,y)$
- This algorithm is “linear in input intensity” and “shift invariant”
- **Discrete convolution**, pattern of weights is called “kernel”

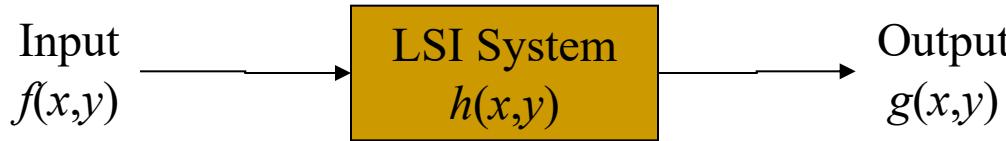
Linear Systems

- Many image processing operations can be modeled as a linear system:
 - $h(x,y)$ is the response to the unit impulse $\delta(x,y)$



■ Space invariant system:

- A system whose response remains the same irrespective of the position of the input pulse
- A LSI (linear space invariant) system can be completely described by its impulse response $h(x,y)$



LSI System

- LSI systems satisfy the linearity property

$$a \cdot f_1(x, y) + b \cdot f_2(x, y) \rightarrow a \cdot g_1(x, y) + b \cdot g_2(x, y)$$

- The output is the convolution of input with the impulse response

$$\begin{aligned}g(x, y) &= f(x, y) * h(x, y) \\&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x', y') h(x - x', y - y') dx' dy'\end{aligned}$$

- For the discrete case

$$\begin{aligned}g[i, j] &= f[i, j] * h[i, j] \\&= \sum_{k=1}^n \sum_{l=1}^m f[k, l] h[i - k, j - l]\end{aligned}$$

Convolution

- If f and g are images, convolution becomes the computation of weighted sums of the image pixels
- The impulse response, $h[i, j]$, is referred to as a convolution mask or convolution kernel
- The value $g[i, j]$ of the new image is calculated by

$$g[i, j] = f[i, j] * h[i, j] = \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} f[h, k]h[i - h, j - k]$$

Convolution

- Convolution is a linear and spatial invariant operation
- Convolution in the spatial or time domain corresponds to multiplication in the frequency domain

4/25/2023

- HW #4 will be given this Thursday.

Computation

- Convolution is a fairly **expensive** operation requiring a large number of computations on typical images
- Many computer architecture provide specialized instructions for these kinds of operations
- Two convolution kernels commonly used for noise reduction are
 - Mean Kernel (average smoothing)
 - Gaussian Kernel
- Some Facts
 - $X * (aY + bZ) = a(X * Y) + b(X * Z)$
 - $X * (Y * Z) = (X * Y) * Z$

Linear Filters

■ Linear filters

- Implemented using the weighted sum of the pixels in successive window
- Usually spatial invariant and implemented using a convolution mask
- Can also be spatially variant (different filter weights used for different parts of the image)

■ Nonlinear filters

- Not a weighted sum of pixels
- Can be spatially invariant or spatially variant
- **Median filter**, for instance

Linear Smoothing Filter

- “Good” linear smoothing filters?
 - Should be chosen so that the filter has a single peak (main lobe)
 - Symmetric in the vertical and horizontal directions
 - Remove high-frequency components
 - Will lose sharp detail in the image
 - Step changes will be blurred into gradual changes

Spatial Filtering

- Use filter (mask/kernel/template/window)
- The values in a filter subimage are referred to as coefficients, rather than pixel
- Simply move the filter mask from point to point in an image
- At each point (x,y) , the response of the filter at that point is calculated using a predefined relationship

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$$R = w_1 z_1 + w_2 z_2 + \dots + w_{mn} z_{mn} = \sum_{i=1}^{mn} w_i z_i$$

3×3 Smoothing Linear Filters

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

box filter

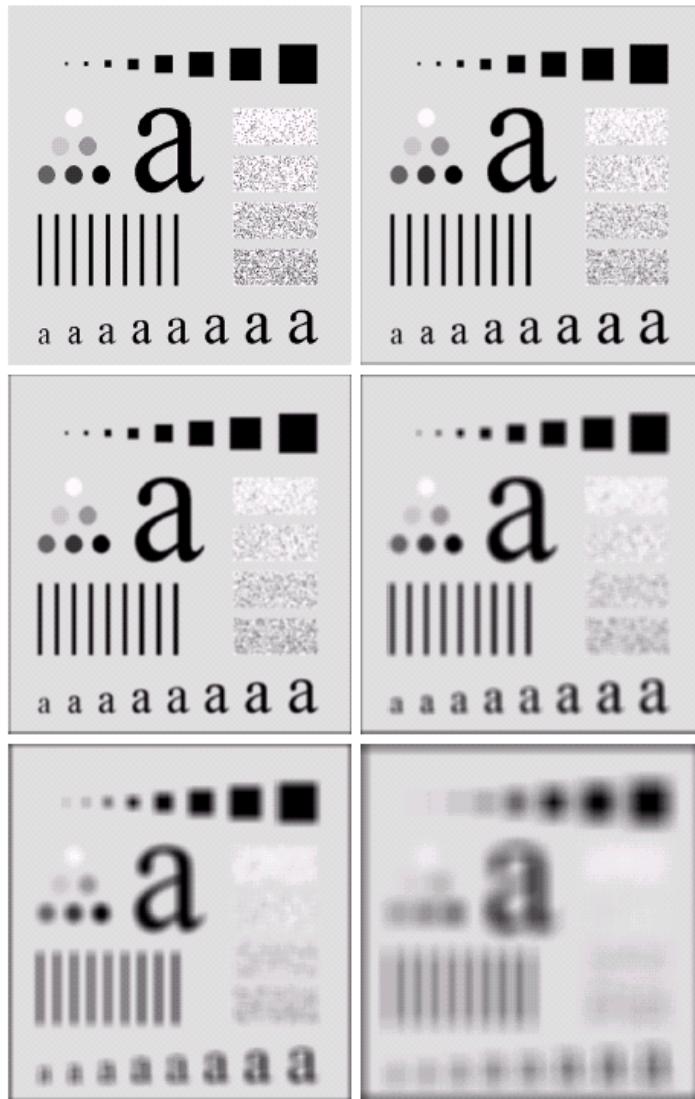
$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

weighted average

The center is the most important and other pixels are **inversely** weighted as a function of their distance from the center of the mask

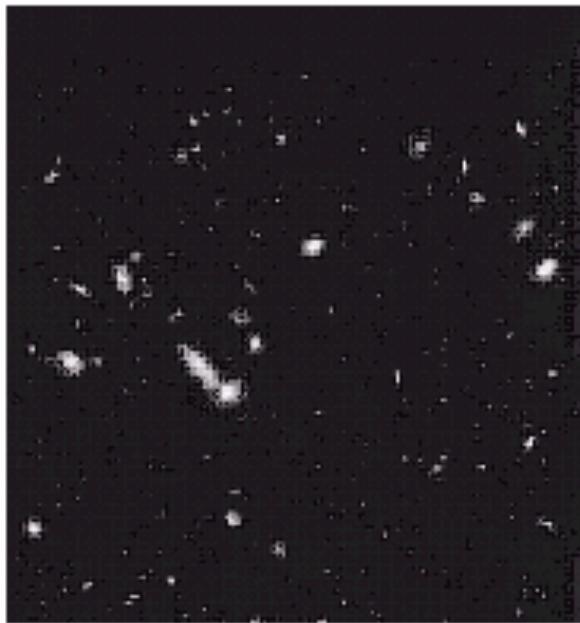


Example

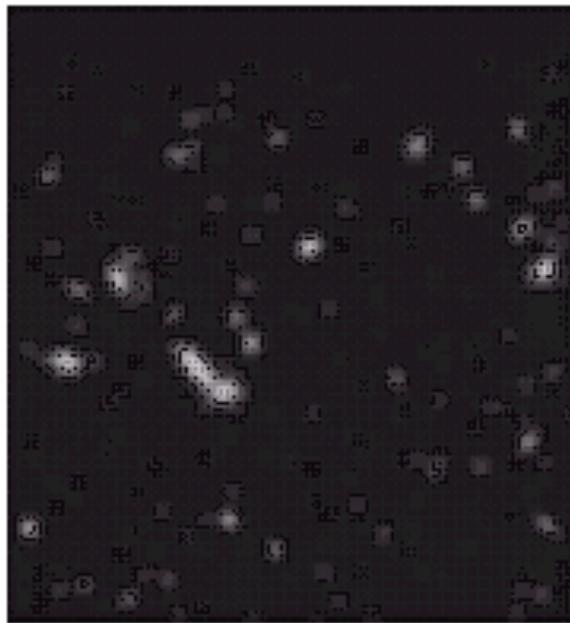


- a). original image 500×500 pixel
- b). - f). results of smoothing with square averaging filter masks of size $n = 3, 5, 9, 15$ and 35 , respectively.
- Note:
 - Big mask is used to eliminate small objects from an image
 - The size of the mask establishes the relative size of the objects that will be blended with the background

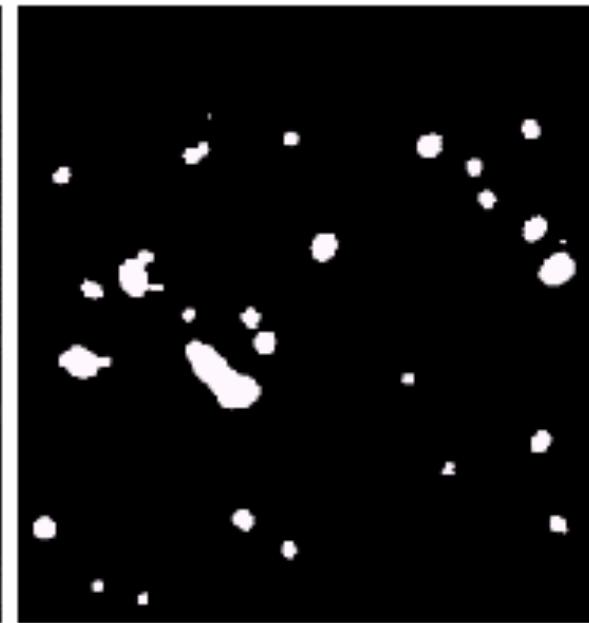
Example - Application



original image



result after smoothing with
15x15 averaging mask



result of thresholding

- We can see that the result after smoothing and thresholding, the remains are the largest and brightest objects in the image

Mean Filter

- Smoothing by averaging

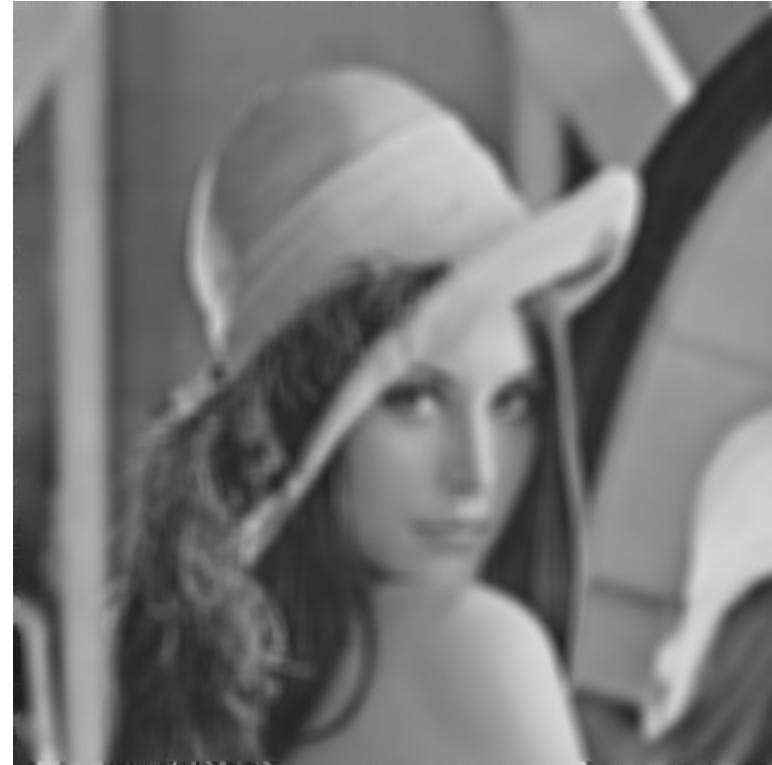
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

- Entries must add up to one (Why?)

Example



Mean 5×5



Mean 11×11

Example



Frequency Behavior

- Fourier transform of a 1-D mean filter kernel of width $2W$ is the *sinc* function

$$\text{sinc}(\omega) = \frac{2 \sin(\omega W)}{\omega}$$

- (Box filter)
- Behave like a low-pass filter

Problems of Mean Filtering

- Sharp signal variations are filtered out, image is *blurred*
- *Impulse noise* is attenuated and diffused, **not** removed
- *Secondary lobes* in the Fourier transform contribute noise (compare with Gaussian!)

4/27/2023

- HW #4 will be given at 11:45.

Separable Filters

- Some 2D image filters can actually be implemented as two 1-D filters:
 - One in the horizontal direction followed by another in the vertical direction
- This can significantly lower the computational complexity of the operation
- Gaussian filter
 - A class of linear smoothing filters with the weights chosen according to the shape of a Gaussian function
 - A very good filter for removing noise drawn from a **normal distribution**

Gaussian Filters

- Gaussian filter
 - Linear smoothing filter
 - Weights chosen according to the shape of Gaussian function
- One-dimensional Gaussian (zero mean)

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

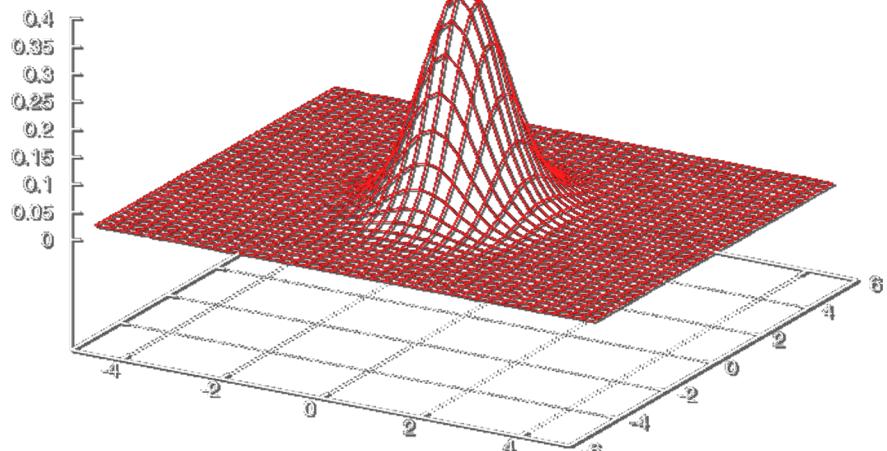
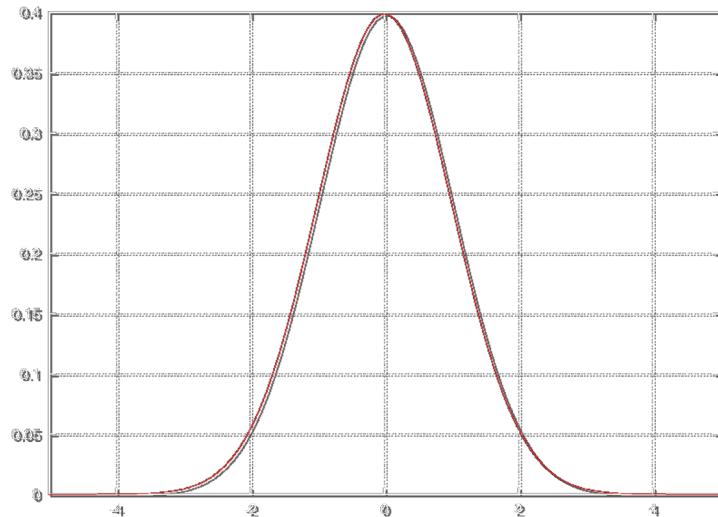
- Two-dimensional Gaussian (zero mean)

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

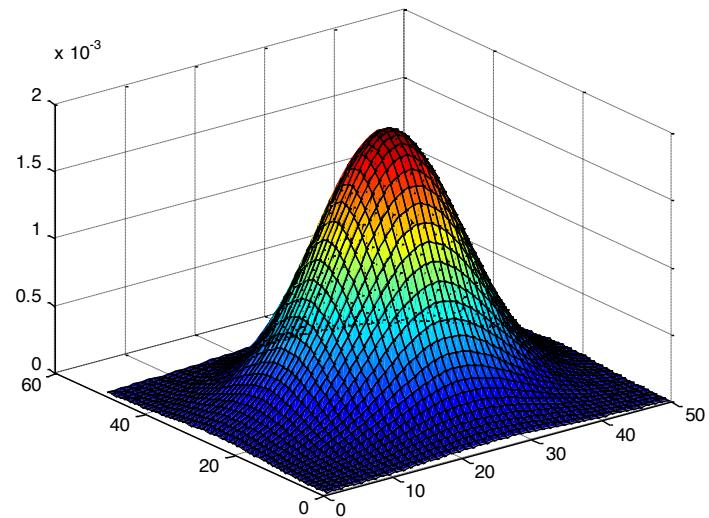
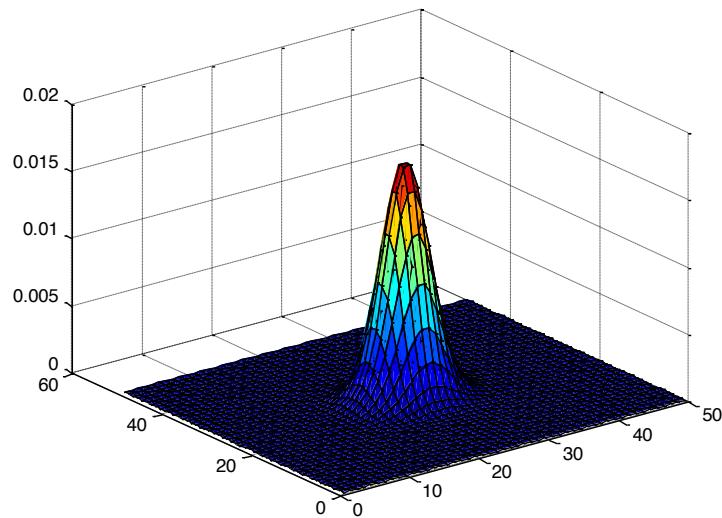
Gaussian Filters

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

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



Gaussian Filters



Properties of Gaussian Filter

- Rotationally symmetric
 - Amount of smoothing is the same in all direction
 - Does not bias subsequent edge detection in any particular direction
- Single lobe in spatial domain
 - Replace each image pixel with a weighted average of the neighbors
 - The weight decreases monotonically with distance from the central pixel

Properties of Gaussian Filter

- Single lobe in frequency domain
 - High-frequency noise is suppressed, etc.
 - No secondary lobe, better low-pass than mean filter
- Degree of smoothing controlled by σ
 - Larger σ or width gives greater smoothing
- Separable (in x and y directions)
 - Effective implementation
 - Large Gaussian filters can be implemented very efficiently because of separability of Gaussian functions

Separability of Gaussian Kernel

- Convolution can be done on all rows and then all columns

$$\begin{aligned} I * G &= \sum_{m=-\frac{w}{2}}^{\frac{w}{2}} \sum_{n=-\frac{w}{2}}^{\frac{w}{2}} G(m, n) I(i - m, j - n) \\ &= \sum_{m=-\frac{w}{2}}^{\frac{w}{2}} \sum_{n=-\frac{w}{2}}^{\frac{w}{2}} e^{-\frac{m^2+n^2}{2\sigma^2}} I(i - m, j - n) \\ &= \sum_{m=-\frac{w}{2}}^{\frac{w}{2}} e^{-\frac{m^2}{2\sigma^2}} \sum_{n=-\frac{w}{2}}^{\frac{w}{2}} e^{-\frac{n^2}{2\sigma^2}} I(i - m, j - n) \end{aligned}$$

Cascading Gaussians

- The convolution of a Gaussian with itself yields a scaled Gaussian with larger σ

$$g(x) * g(x) = \int_{-\infty}^{\infty} e^{-\frac{\xi^2}{2\sigma^2}} e^{-\frac{(x-\xi)^2}{2\sigma^2}} d\xi = \sqrt{\pi\sigma} e^{-\frac{x^2}{2(\sqrt{2}\sigma)^2}}$$

- Convolution of two Gaussian with σ gives a Gaussian with $\sqrt{2}\sigma$

Design of Gaussian Filter

- Approximation from **Pascal's triangle**
 - 1 2 1, 1 3 3 1, 1 4 6 4 1, ...
 - Normalization
- Cascading Gaussians give a scaled Gaussian with larger σ
- Compute Gaussian kernel weight directly from the discrete Gaussian function
 - Given σ , calculate the function:
$$g[i, j] = e^{-\frac{i^2 + j^2}{2\sigma^2}}$$
 - Convert to integers
 - Normalization

Gaussian Kernel Example

$$\frac{1}{10.7}$$

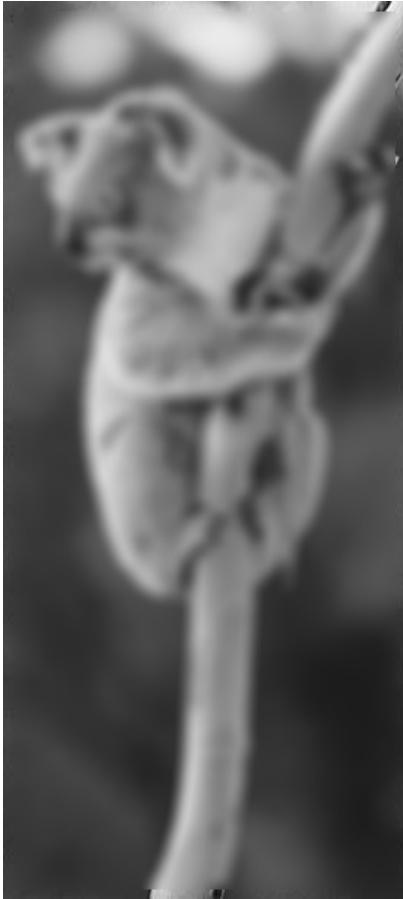
1.3	3.2	3.8	3.2	1.3
-----	-----	-----	-----	-----

$$\sigma = 1.4$$

$$\frac{1}{115}$$

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

Example



Nonlinear Filters

- The response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter

- Example
 - median filter : $R = \text{median}\{z_k \mid k = 1, 2, \dots, n \times n\}$
 - max filter : $R = \max\{z_k \mid k = 1, 2, \dots, n \times n\}$
 - min filter : $R = \min\{z_k \mid k = 1, 2, \dots, n \times n\}$
- Note: $n \times n$ is the size of the mask

Median Filters

- Replace the value of a pixel by the median of the gray levels in the neighborhood of that pixel
- Quite popular because for certain types of random noise (impulse noise \Rightarrow salt and pepper noise), they provide excellent noise-reduction capabilities, with considering less blurring than linear smoothing filters of similar size
- Force the points with distinct gray levels to be more like their neighbors

Median Filters

- Isolated clusters of pixels that are light or dark w.r.t. their neighbors, and whose area is less than $n^2/2$ (one-half the filter area), are eliminated by an $n \times n$ median filter
- Eliminated = forced to have the value equal the median intensity of the neighbors

Example

■ Salt & Pepper Noise



Median Filtering

- Median filtering is a non-linear operation that is particularly effective at removing salt and pepper noise
- Neighborhood values: 115, 119, 120, 123, 124, 125, 126, 127, 150
- Median value: 124

123	234	125	114	102
124	115	123	150	124
125	119	124	127	115
114	120	125	126	134
122	134	115	214	112

Example



Example

■ S & P noise



Reading

- Chapter 4 of Jain's book