Lecture 5

Filtering

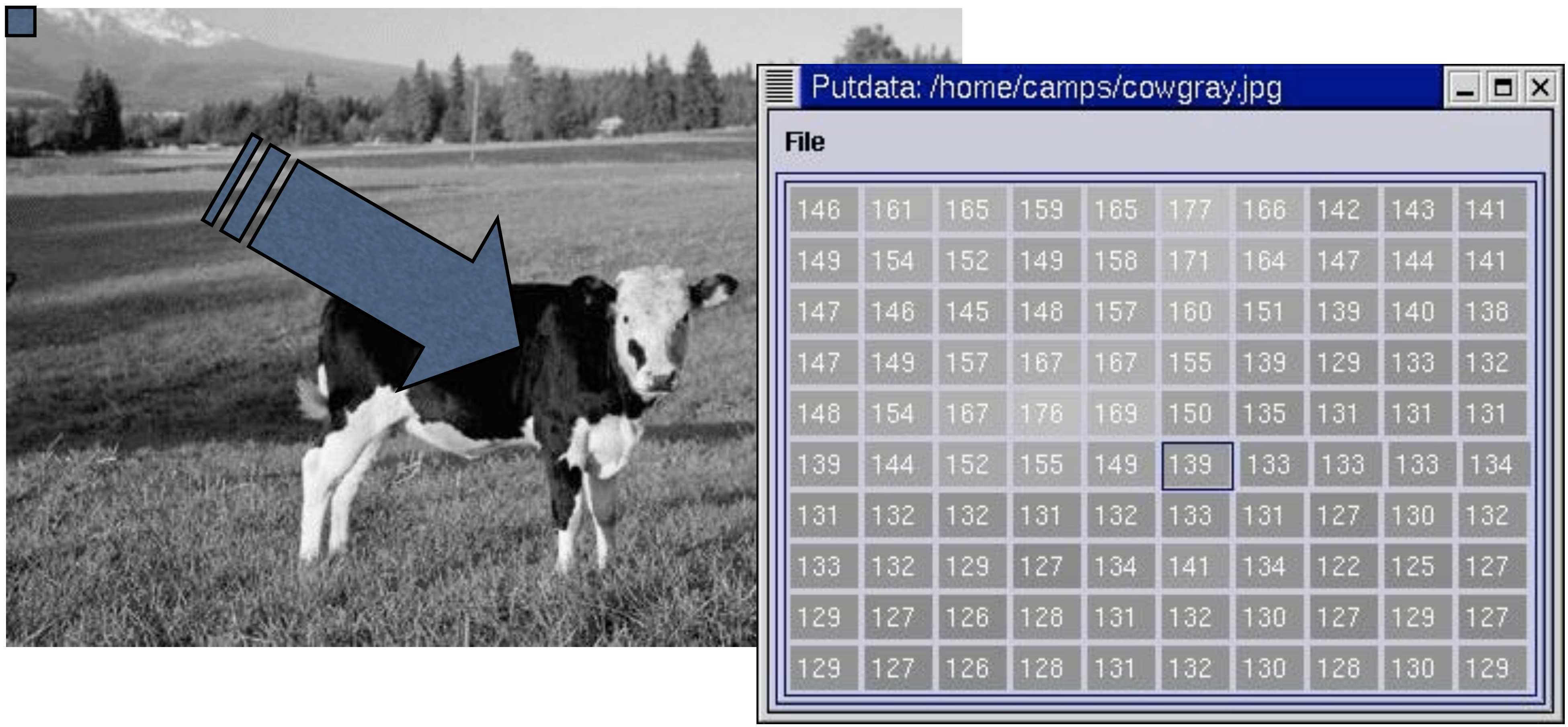
Next Class

Edges, Corners

Image processing: Filtering

Digital Images

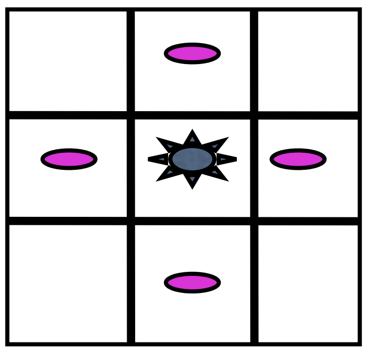
are 2D arrays (matrices) of numbers:



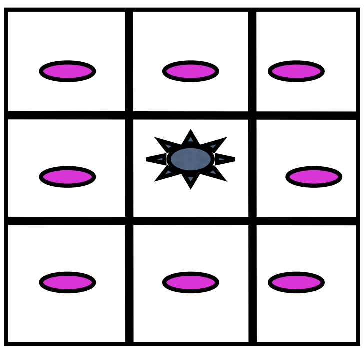
146	161	165	159	165	177	166	142	143	141
149	154	152	149	158	171	164	147	144	141
147	146	145	148	157	160	151	139	140	138
147	149	157	167	167	155	139	129	133	132
148	154	167	176	169	150	135	131	131	131
139	144	152	155	149	139	133	133	133	134
131	132	132	131	132	133	131	127	130	132
133	132	129	127	134	141	134	122	125	127
129	127	126	128	131	132	130	127	129	127
129	127	126	128	131	132	130	128	130	129

Digital Images

- Array of numbers (pixels)
  - Typically integers 0-255 (unsigned byte)
- Pixels have neighbors



4-neighbors

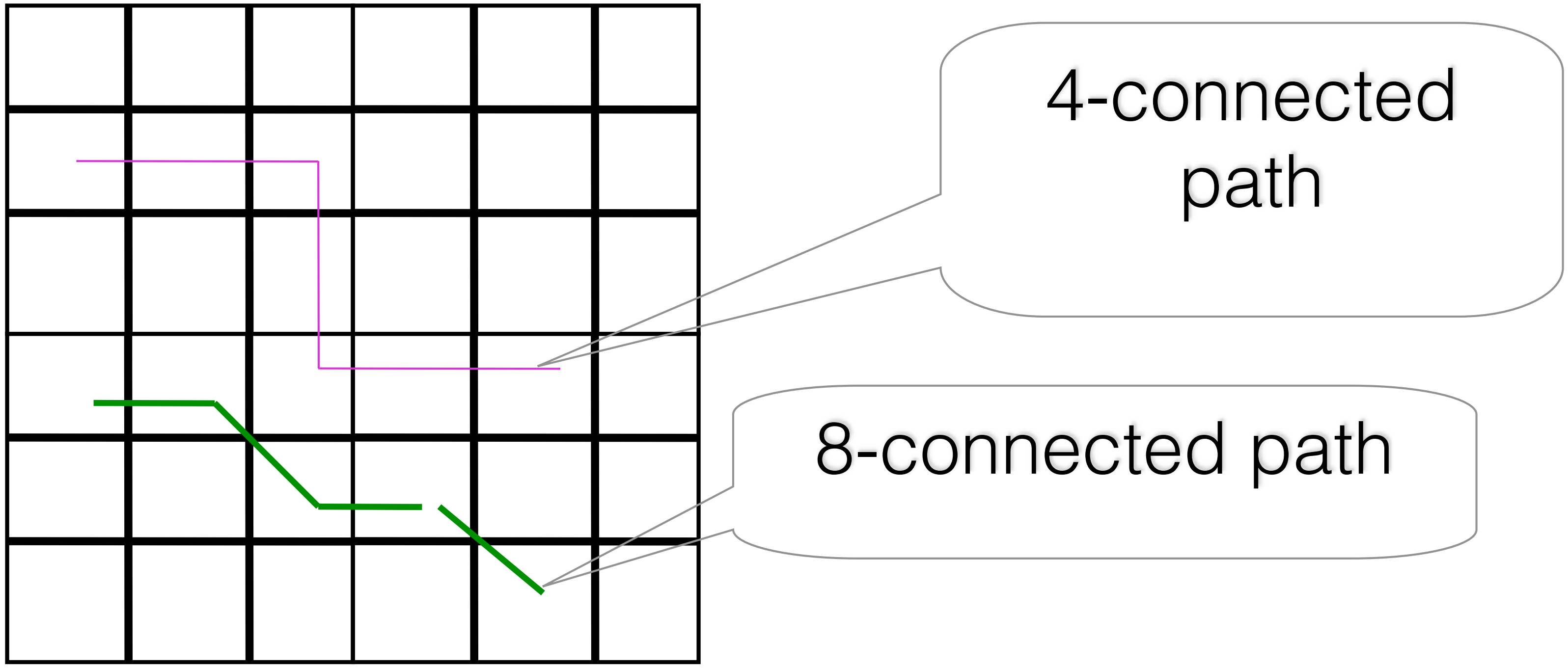


8-neighbors

Image Paths

A path is a sequence of pixel indices $(i_o,j_o)(i_1,j_1)\ldots(i_n,j_n)$ such that

(i_{k},j_{k}) is a neighbor of (i_{k+1},j_{k+1})



Levels of Computation

- Point level
  - Output based only on a single point
    - Ex.: thresholding
- Local level
  - Output based on a neighborhood
    - Ex. : smoothing and edge detection
- Global level
  - Output based on the whole image
    - Ex.: Fourier transform and histogram
- Object level
  - Output based on pixels that belong to an object

Spatial Filtering

Spatial Filtering

- Use of spatial masks (kernels, filters, templates, windows) for image processing (spatial filters)
- Linear and nonlinear filters
- Spatial Filters include:
  - Sharpening
  - Smoothing
  - Edge detection
  - Noise removal
  - etc

Linear Filters

- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)

- Example: smoothing by averaging
  - form the average of pixels in a neighborhood
- Example: smoothing with a Gaussian
  - form a weighted average of pixels in a neighborhood
- Example: finding a derivative
  - form a weighted average of pixels in a neighborhood

Note: The "Linear" in "Linear Filters" means linear combination of neighboring pixel values.

Image Filtering

Low-pass filters eliminate or attenuate high frequency components in the frequency domain (sharp image details), and result in image blurring.

High-pass filters attenuate or eliminate low-frequency components (resulting in sharpening edges and other sharp details).

Band-pass filters remove selected frequency regions between low and high frequencies (for image restoration, not enhancement).

Spatial Filtering

Operations are performed directly on the pixels in the spatial domain.

The process involves sweeping a mask on the image and performing at each point a set of predefined operations on the pixels overlapped by the mask.

Basics of Spatial Filtering

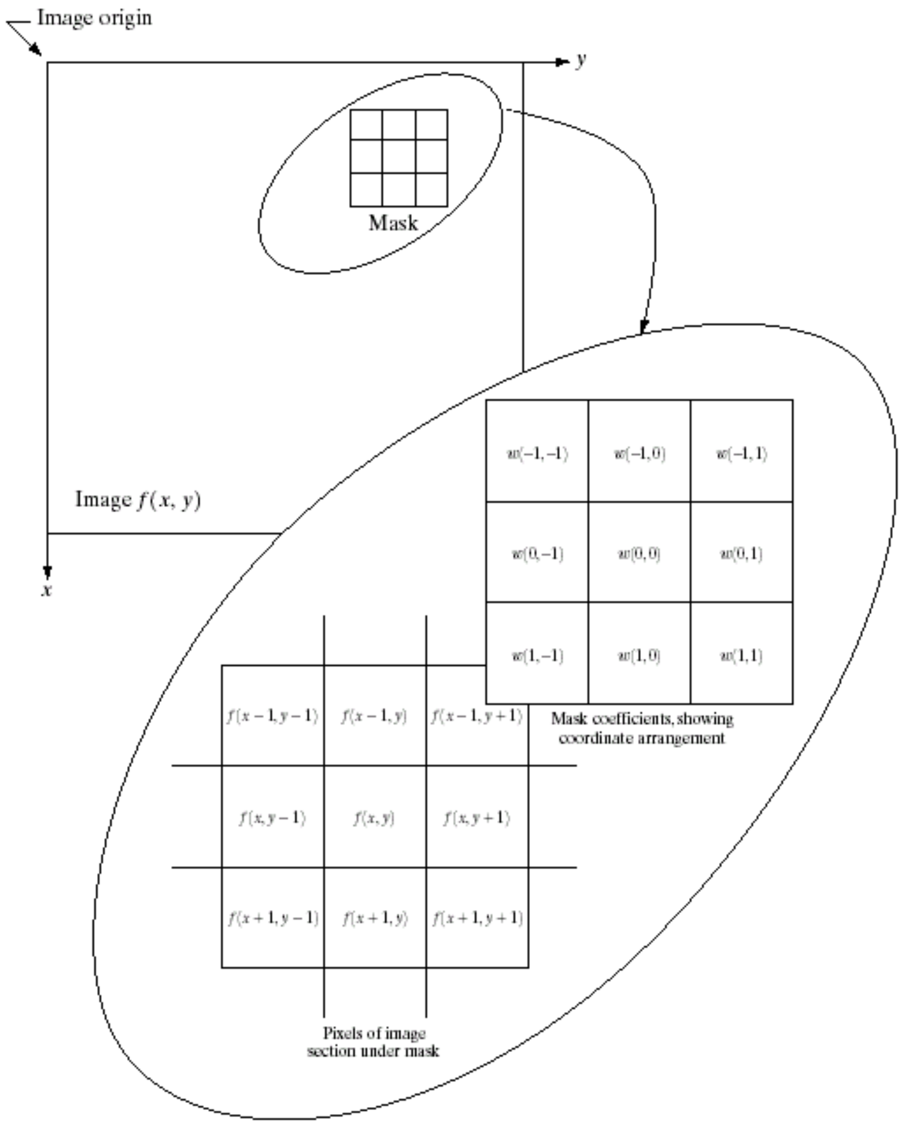


FIGURE 3.32 The mechanics of spatial filtering. The magnified drawing shows a 3×3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

Ex of a 3x3 mask

Linear Spatial Filtering: CORRELATION

Linear filtering of an MxN image f(x,y) with a filter w(s,t) of size mxn is given by:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$$a = (m-1)/2; \quad b = (n-1)/2$$

Where m and n are odd numbers

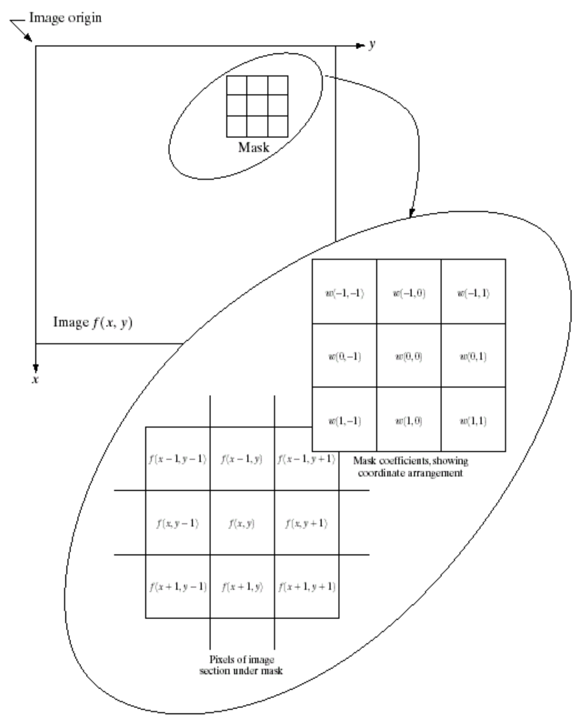


FIGURE 3.32 The mechanics of spatial filtering. The magnified drawing shows a 3 × 3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

This is similar to convolution … and it is often referred as "convolving with a mask"

Linear Spatial Filtering: CORRELATION

Alternative notation:

FIGURE 3.33 Another representation of a general 3×3 spatial filter mask.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$$R = \sum_{i=1}^{m \times n} w_i z_i$$

Where $\mathbf{z}_i$ are the values of the input image under the mask

Correlation Example

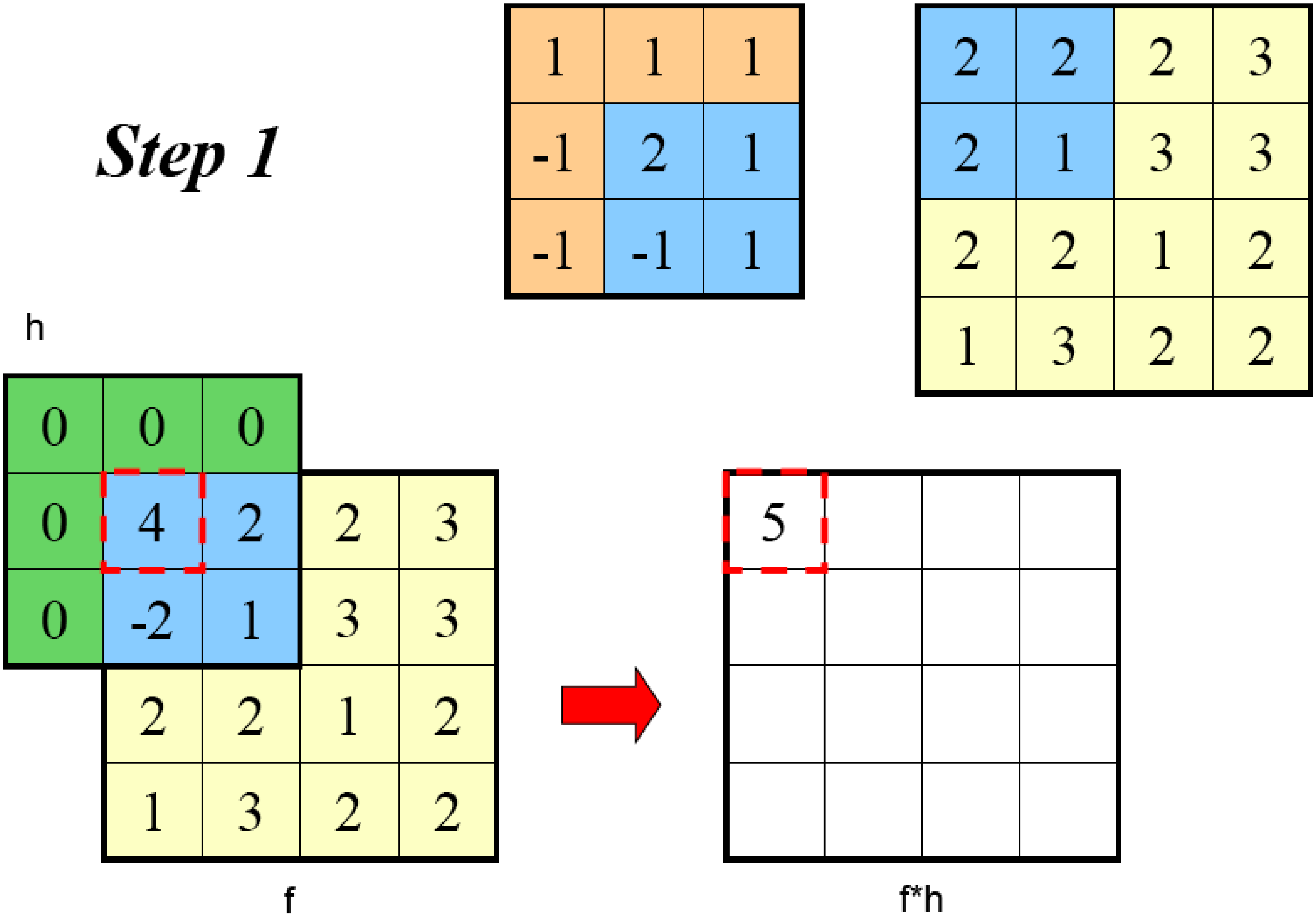
1	1	1
-1	2	1
-1	-1	1

h

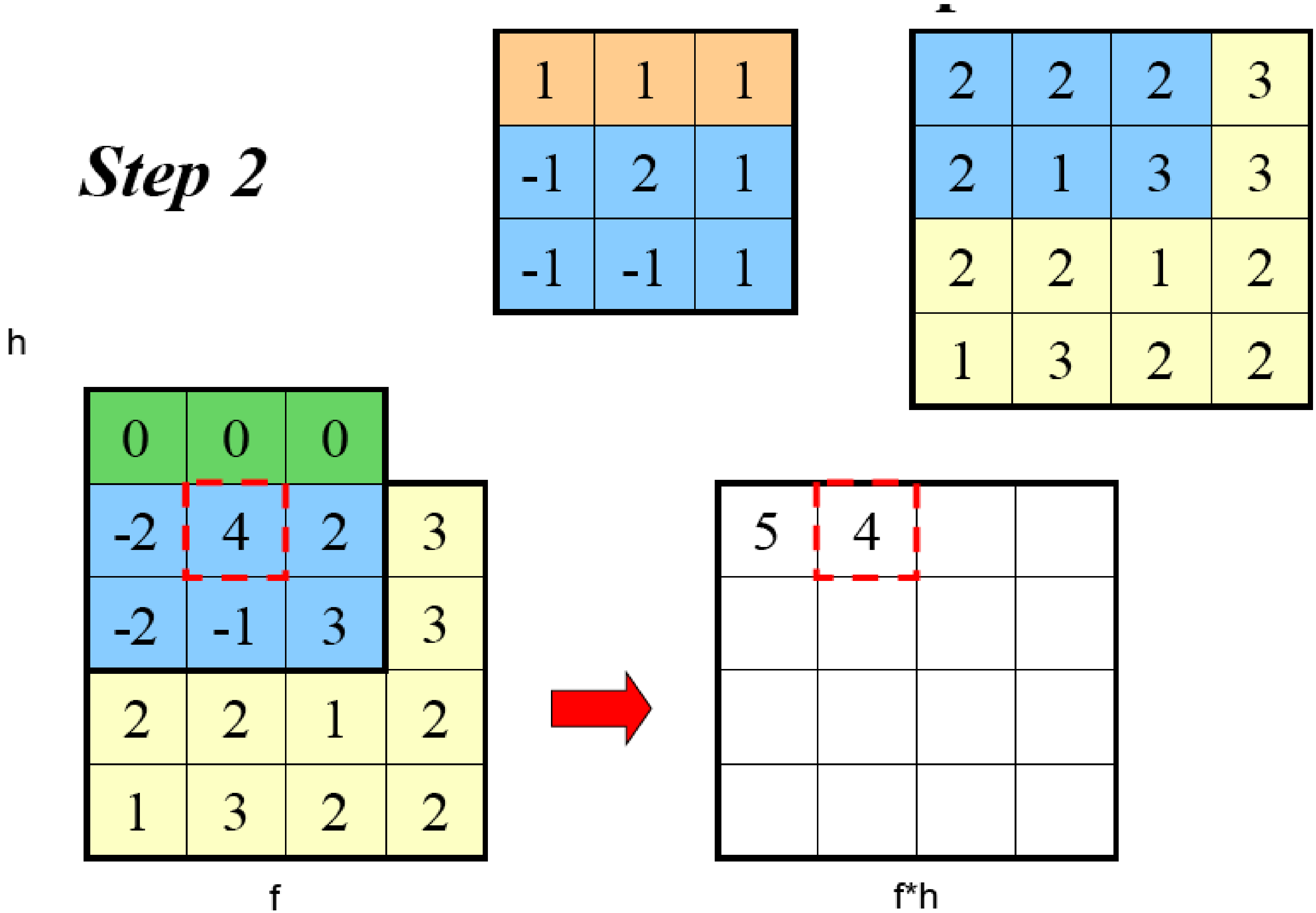
2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

f

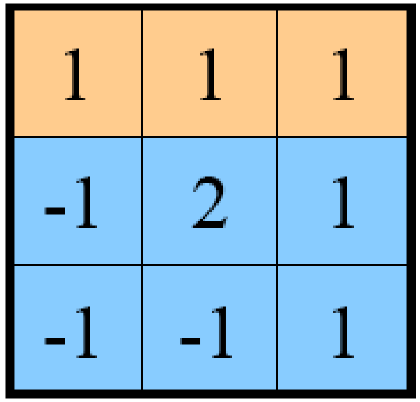
Correlation Example

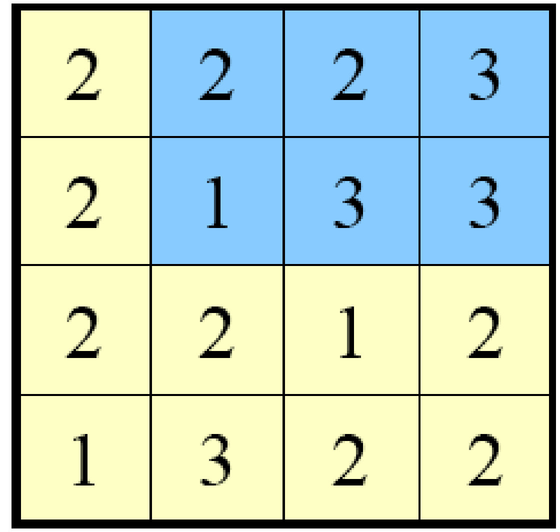


Step 2

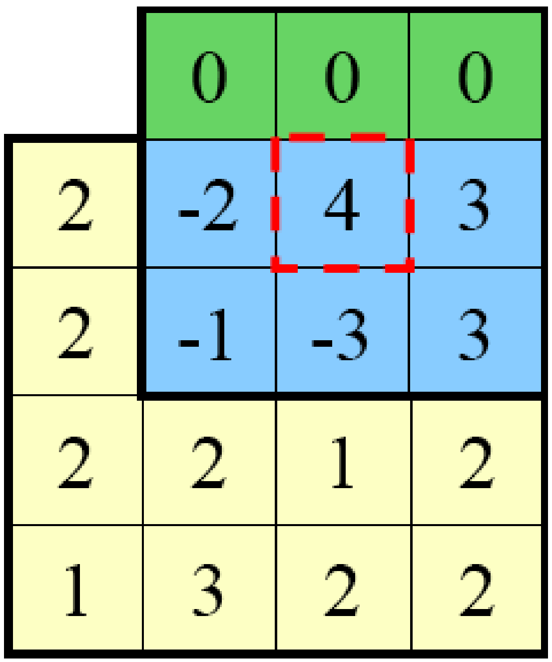

1 1 1
-1 2 1
-1 -1 1
2 2 2 3
2 1 3 3
2 2 1 2
1 3 2 2
h
0 0 0
-2 4 2 3
-2 -1 3 3
2 2 1 2
1 3 2 2
f
5 4
f*h

Step 3

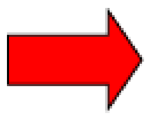


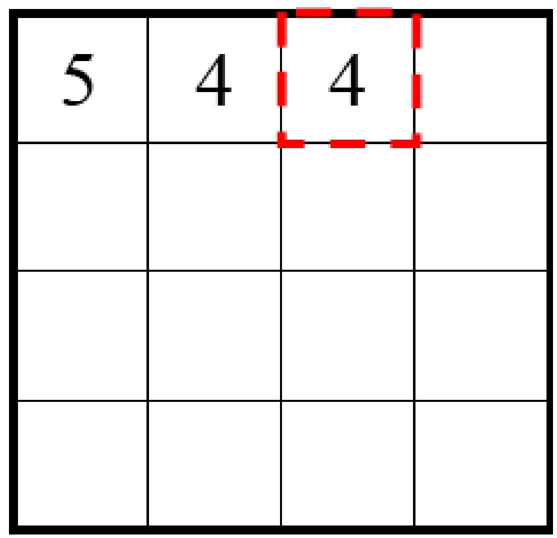


h



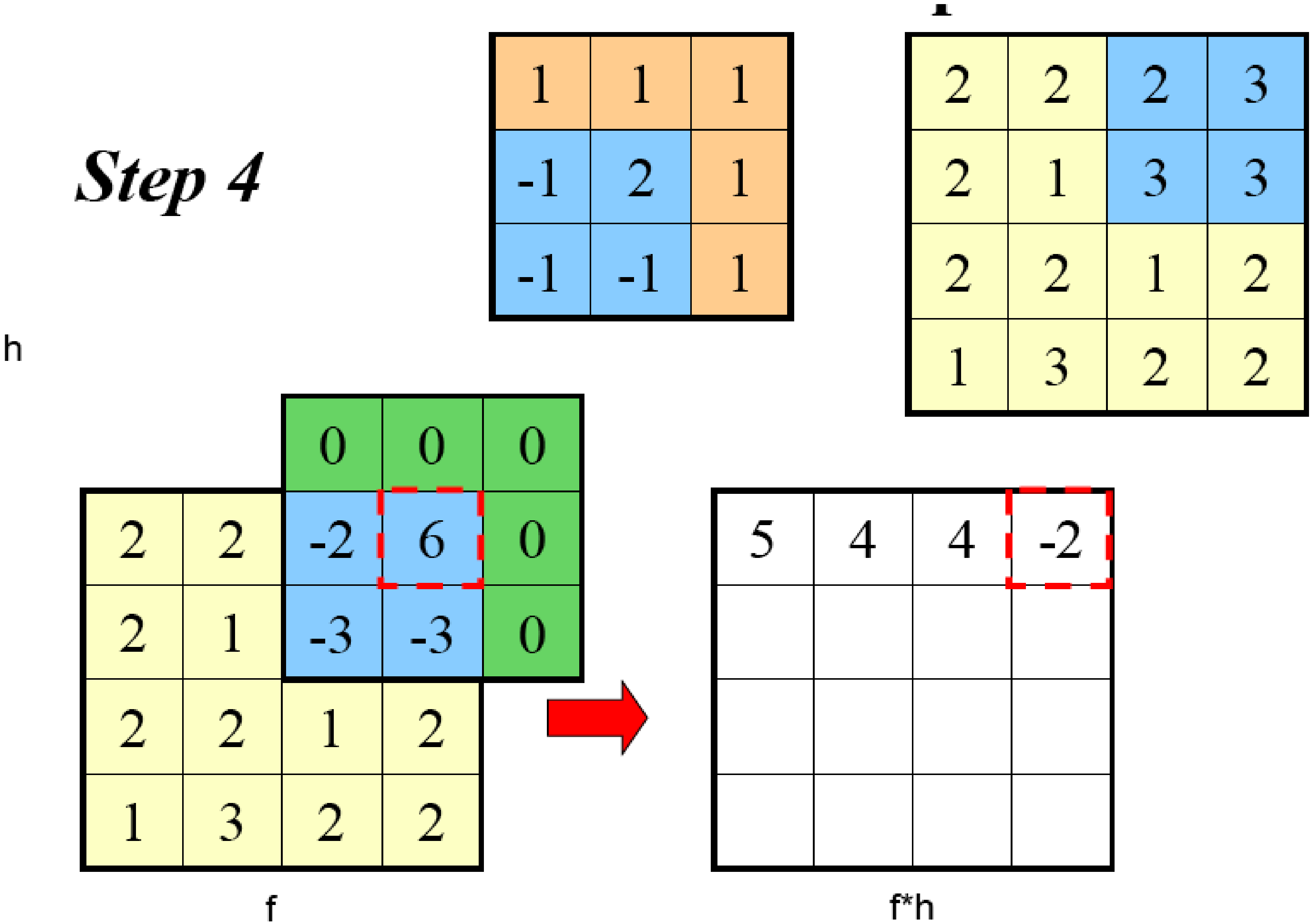
f





f*h

Step 4


1 1 1
-1 2 1
-1 -1 1
2 2 2 3
2 1 3 3
2 2 1 2
1 3 2 2
h
0 0 0
2 2 -2 6 0
2 1 -3 -3 0
2 2 1 2
1 3 2 2
f
5 4 4 -2
f*h

Step 5

h

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

0	2	2	2	3
0	4	1	3	3
0	-2	2	1	2
	1	3	2	2

f

5	4	4	-2
9			

f*h

Step 6

h

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

2	2	2	3
-2	2	3	3
-2	-2	1	2
1	3	2	2

f

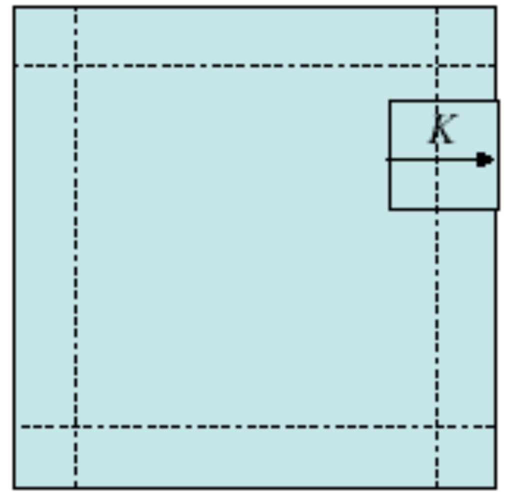
5	4	4	-2
9	6		

f*h

And so on …

Practical Issue: Border Handling

- Border issues:
  - When applying convolution with a KxK kernel, the result is undefined for pixels closer than K pixels from the border of the image

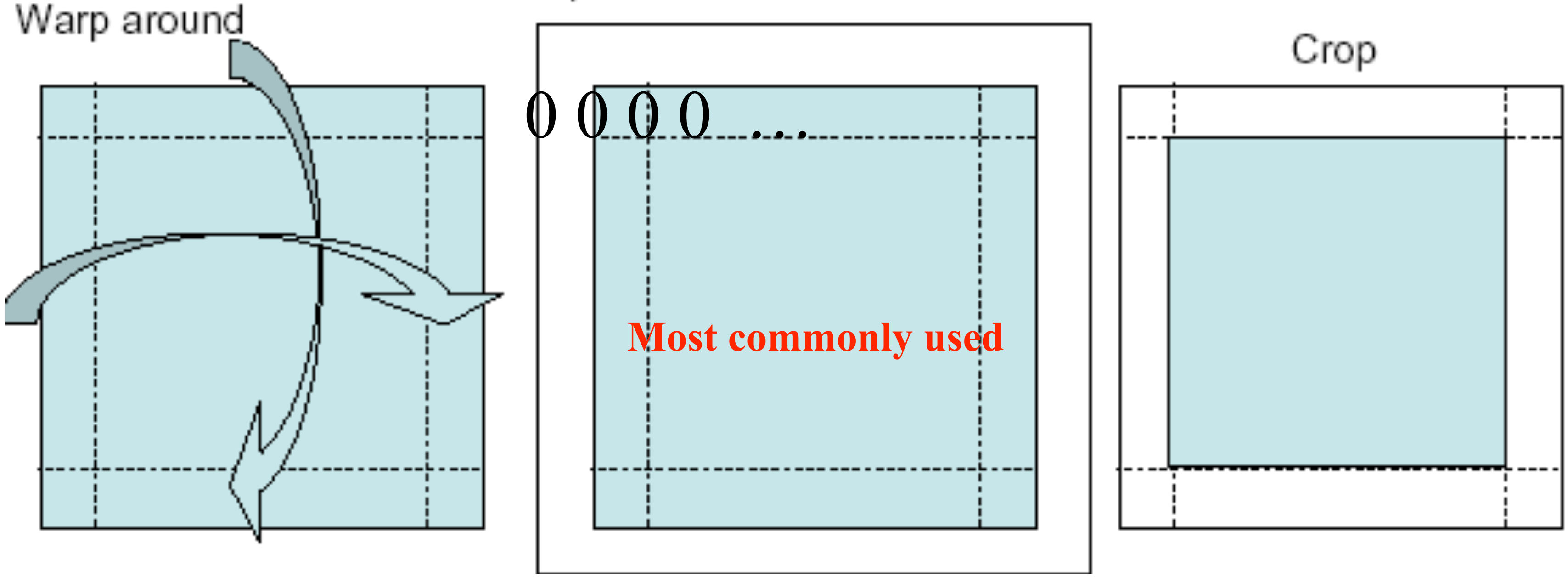


- Options:

Warp around

Expand/Pad

Crop



- Reflection at border also a useful option!

Correlation vs Convolution

Correlation:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$$a = (m-1)/2; \quad b = (n-1)/2$$

FLIP FIRST!

Convolution:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

$$a = (m-1)/2; \quad b = (n-1)/2$$

Correlation vs Convolution

If the mask is symmetric, then there is no difference.

Correlation and Convolution in MATLAB

Could use conv and conv2, but newer versions use:

Imfilter(image,template{,option1,option2,…})

Boundary options: constant, symmetric, replicate, circular

Output size options: same as image, or full size (includes partial values computed when mask is off the image).

Corr or conv option: convolution rotates the template (as we have discussed, correlation does not).

Type “help imfilter” on command line for more details

Smoothing Filters

Image Noise

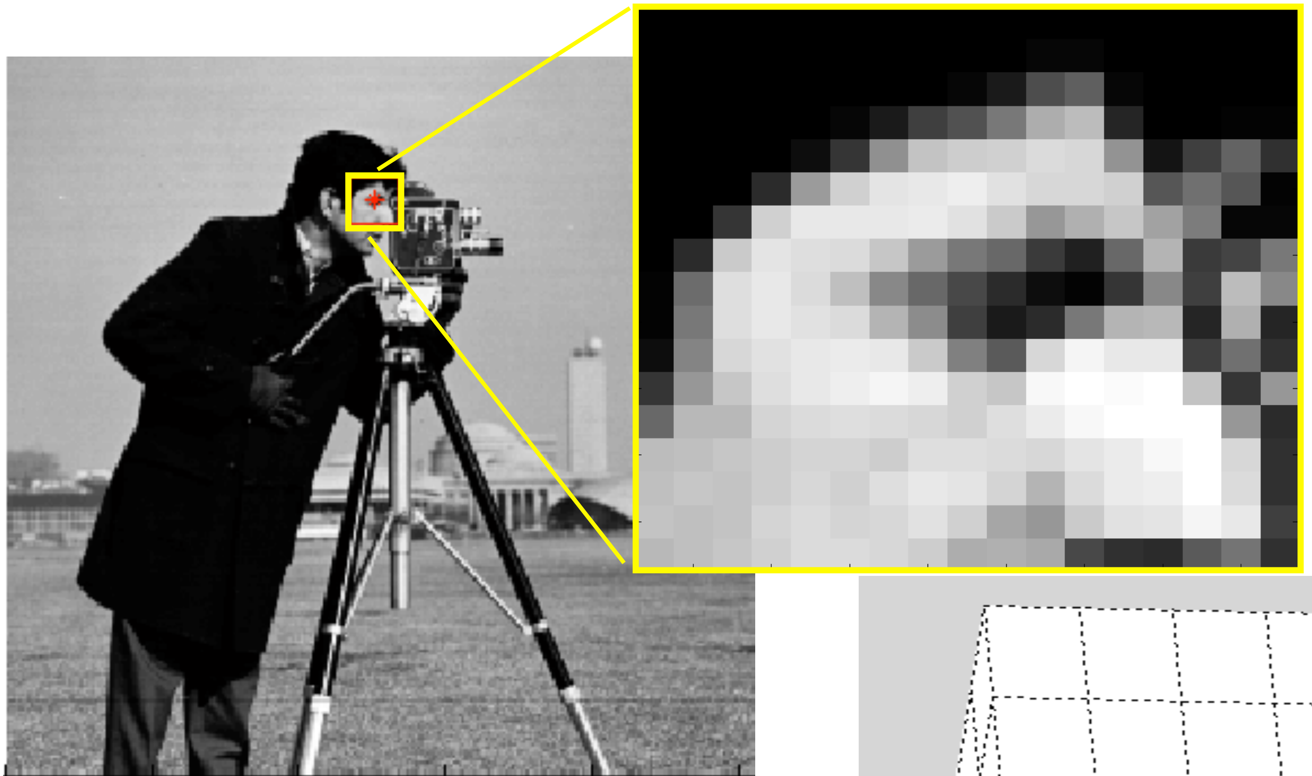
Images are noisy

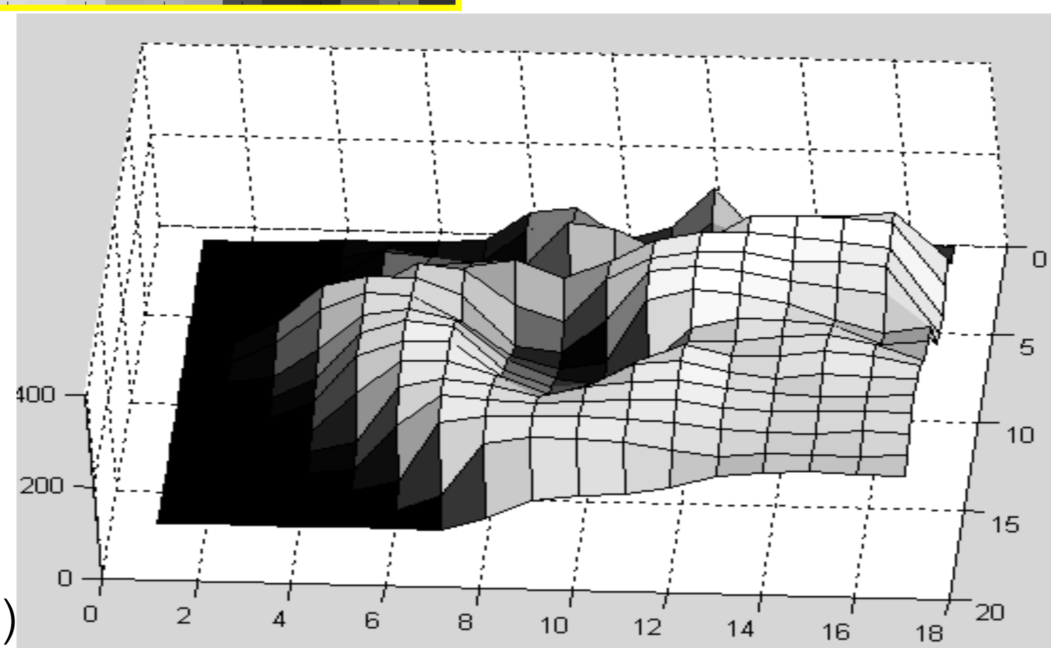
Noise is anything in the image that we are not interested in

Examples:

- Fluctuations of pixel values
- Numerical errors
- Clutter

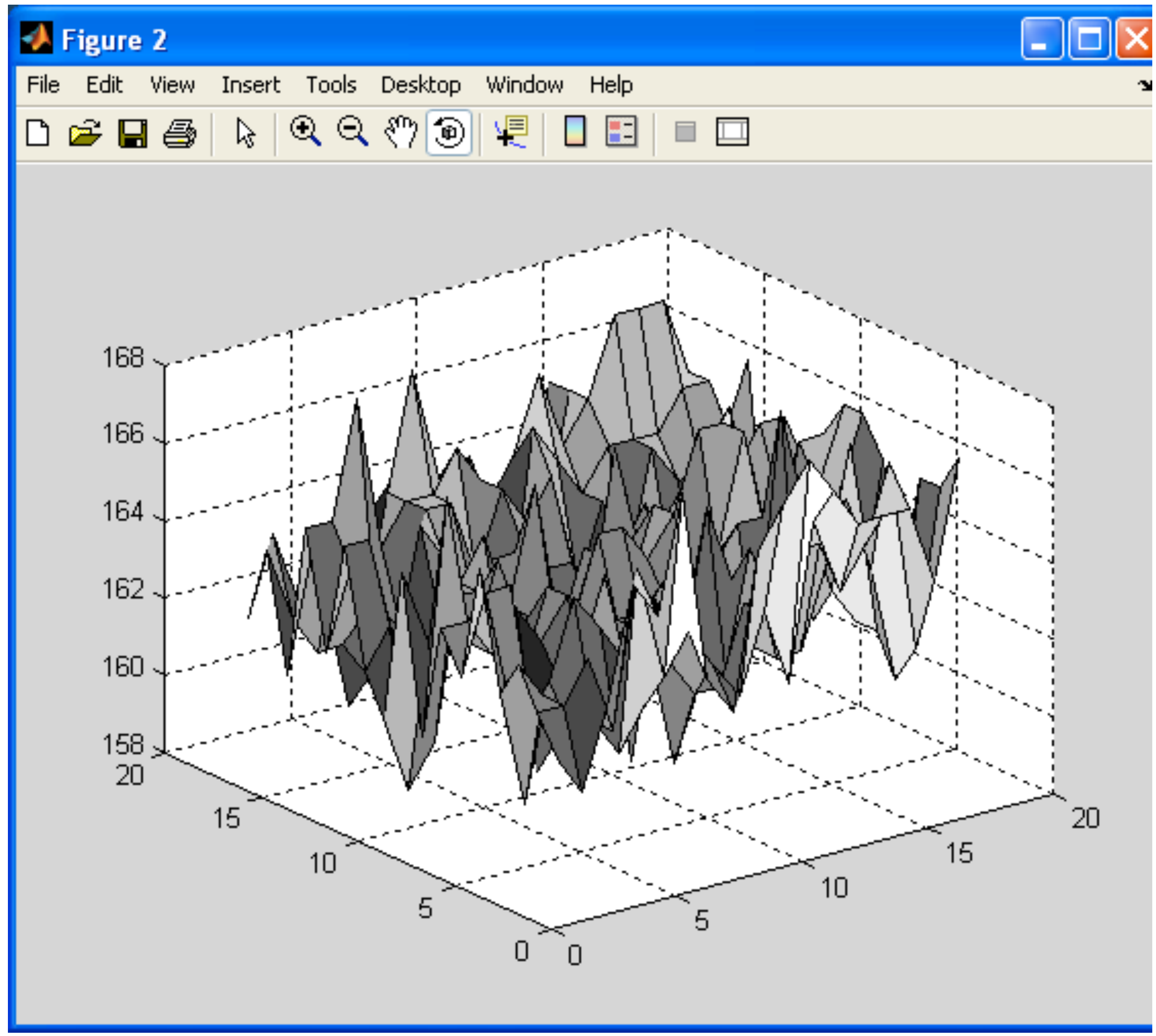
Images as Surfaces





Surface height proportional to pixel grey value (dark=low, light=high)

Examples



Mean = 164 Std = 1.8

Where does noise come from?

Light fluctuations

Sensor noise

Quantization effects

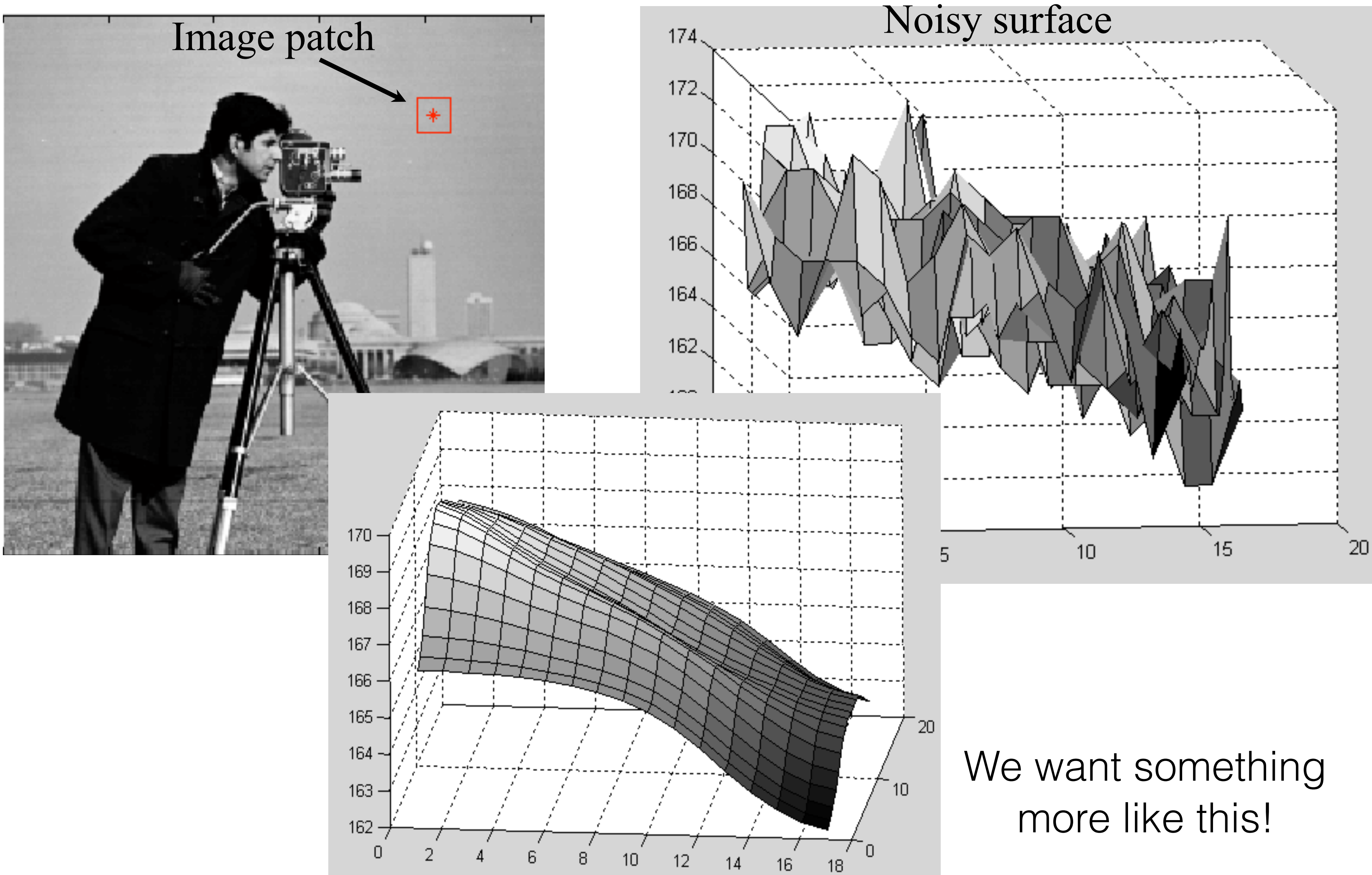
Finite precision

Modeling Noise

We are interested in RANDOM noise.

Deterministic noise (ex: hardware defects) can be corrected.

Dealing with Noise



We want something more like this!

Probability Review

Intuitive Development

Intuitively, the probability of an event **a** could be defined as:

$$P(a) = \lim_{n \to \infty} \frac{N(a)}{n}$$

Where N(a) is the number that event a happens in n trials

More Formal:

Ω is the Sample Space:

Contains all possible outcomes of an experiment

ω in Ω is a single outcome

A in Ω is a set of outcomes of interest

1. $P(A) \geq 0 \forall A \in \Omega$
2. $P(\Omega) = 1$
3. $A_i \cap A_j = \emptyset \forall i, j \Rightarrow P(\cup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i)$
4. $P(\emptyset) = 0$

Independence

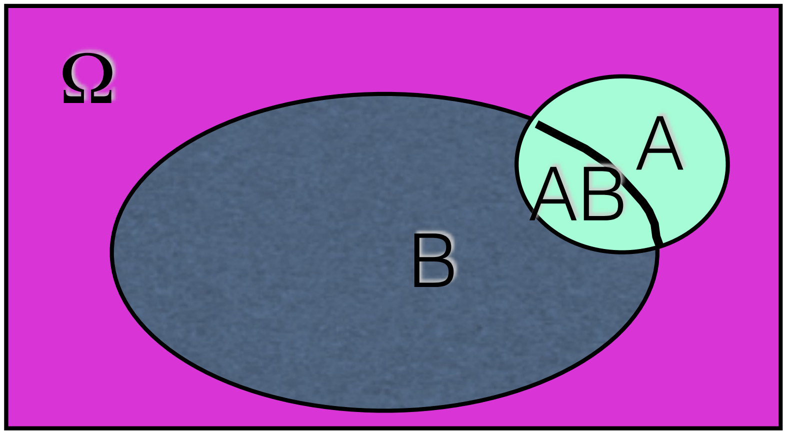
The probability of independent events A, B and C is given by:

$$P(ABC) = P(A)P(B)P(C)$$

A and B are **independent**, if knowing that A has happened does not say anything about B happening

Conditional Probability

One of the most useful concepts!



$$P(A|B) = \frac{P(AB)}{P(B)}$$

Bayes Theorem

Provides a way to convert a-priori probabilities to a-posteriori probabilities:

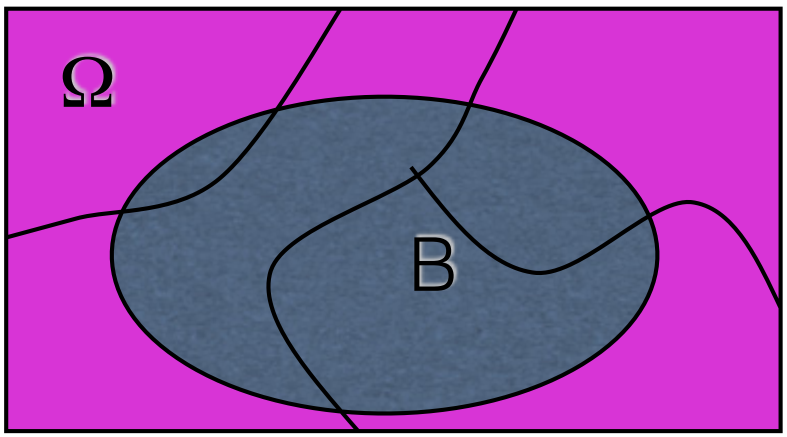
$$P(A|B) = \frac{P(AB)}{P(B)}$$

$$P(A|B)P(B) = P(B|A)P(A)$$

Using Partitions:

If events A_i are mutually exclusive and partition Ω

$$A_i \cap A_j = \emptyset \forall i, j$$

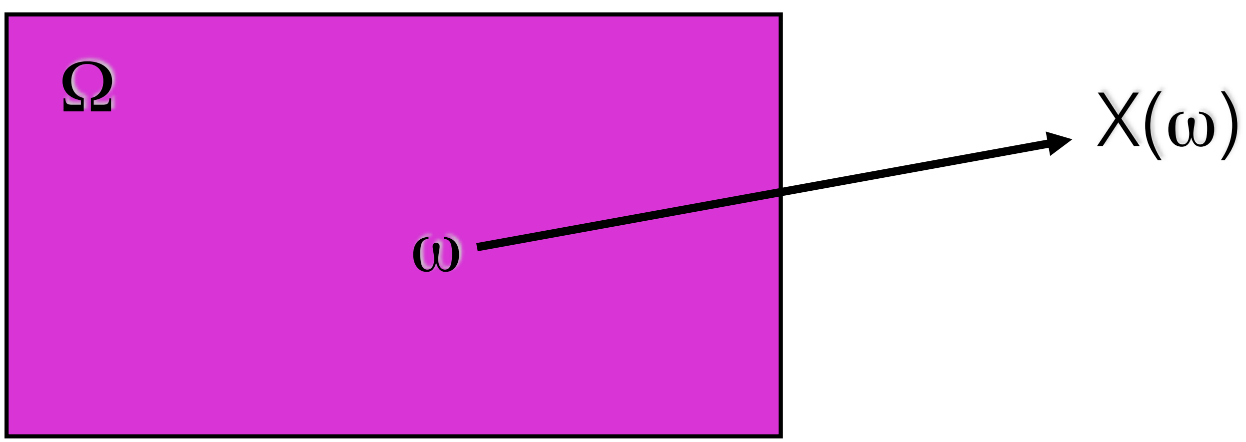


$$\cup_{i=1,n} A_i = \Omega$$

$$P(B) = \sum_{i=1}^{n} P(A_i \cap B)$$

Random Variables

A (scalar) random variable X is a function that maps the outcome of a random event into real scalar values



Random Variables Distributions

Cumulative Probability Distribution (CDF):

$$F_X(x) = P(X \leq x)$$

Random Variable

value

- Probability Density Function (PDF):

$$p_X(x) = \frac{dF_X(x)}{dx}$$

Random Distributions:

From the two previous equations:

$$\int_{-\infty}^{\infty} p_X(x)dx = 1.0$$

"The area under the curve is equal to 1.0."

Statistical Characterizations

Expectation (Mean Value, First Moment):

$$E(X) = \int_{-\infty}^{\infty} x p_X(x) dx$$

"weighted average"

- Second Moment:

$$E(X^2) = \int_{-\infty}^{\infty} x^2 p_X(x) dx$$

Statistical Characterizations

Variance of X:

"how far is x from the mean"

$$
\begin{aligned}
Var(X) &= E\{[X - E(X)]^2\} \\
&= \int_{-\infty}^{\infty} (x - E[X])^2 p_X(x)dx \\
&= E[X^2] - (E[X])^2
\end{aligned}
$$

Second Moment (First Moment)^2

- Standard Deviation of X:

$$\sigma_X = \sqrt{Var(X)}$$

Mean Estimation from Samples

Given a set of N samples from a distribution, we can estimate the mean of the distribution by:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Variance Estimation from Samples

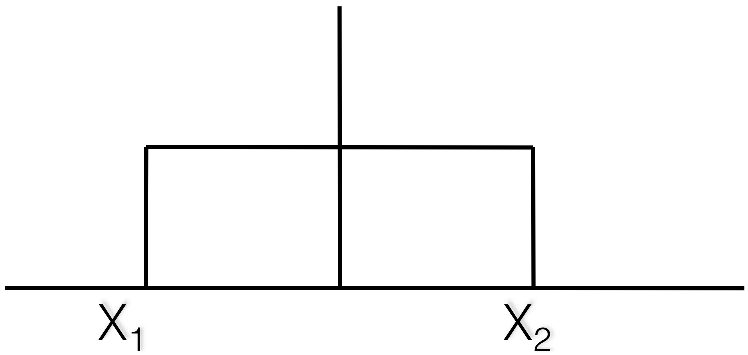
Given a set of N samples from a distribution, we can estimate the variance of the distribution by:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2$$

Uniform Distribution

A R.V. X that is uniformly distributed between x_1 and x_2 has density function:

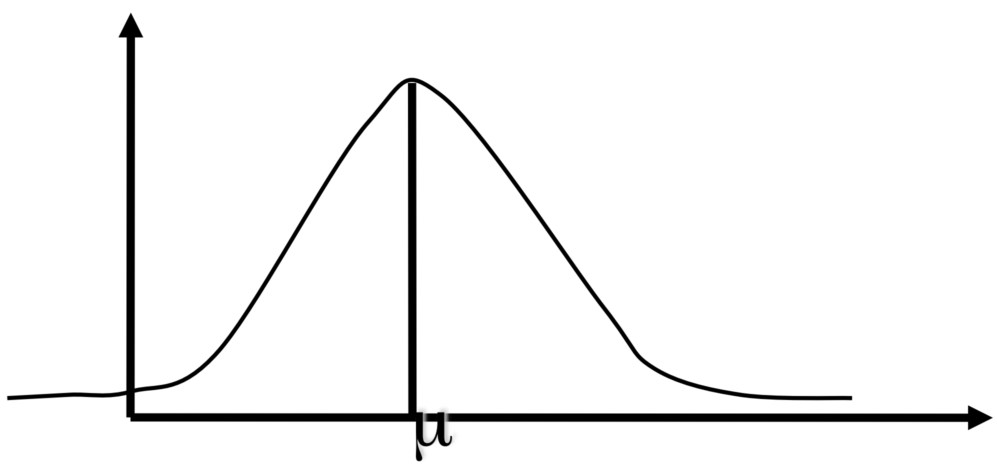
$$p_X(x) = \begin{cases} \frac{1}{x_2 - x_1} & x_1 \leq x \leq x_2 \\ 0 & otherwise \end{cases}$$



Gaussian (Normal) Distribution

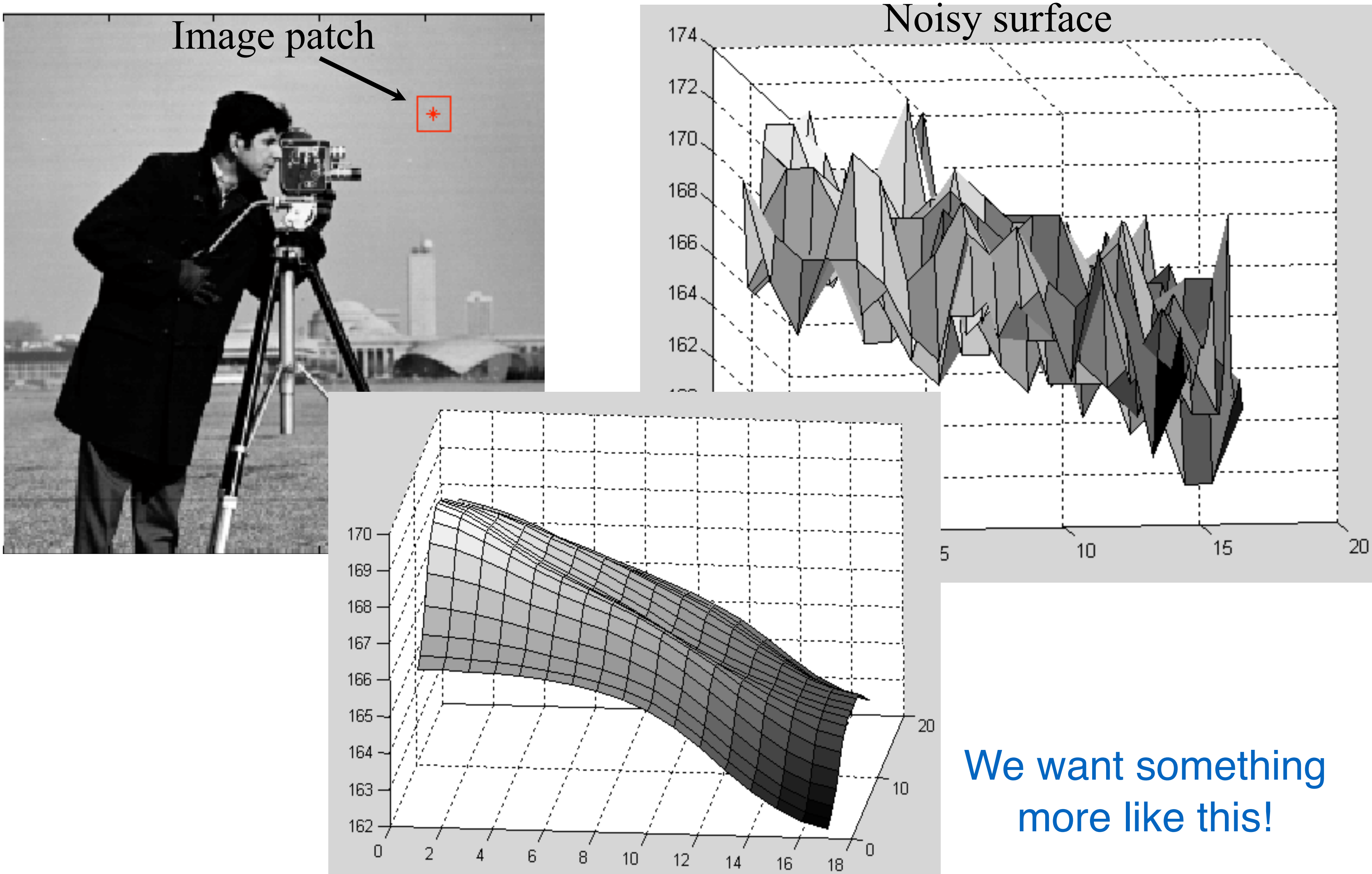
A R.V. X that is normally distributed has density function:

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



… back to images

Dealing with Noise



We want something more like this!

Image Noise Models

Additive noise:

Most commonly used

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

Multiplicative noise:

$$I(i,j) = \hat{I}(i,j) \times N(i,j)$$

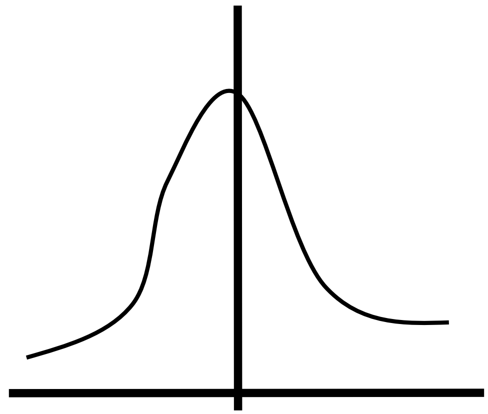
Impulsive noise (salt and pepper):

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

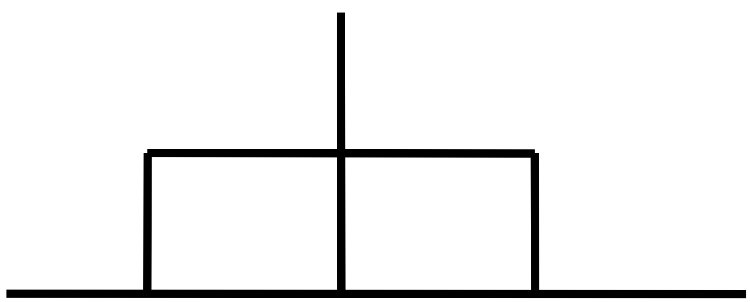
Additive Noise Models

Gaussian

Usually, zero-mean, uncorrelated



Uniform



Measuring Noise

Noise Amount: SNR = σ_s / σ_n

Noise Estimation:

Given a sequence of images $I_0, I_1, \ldots I_{N-1}$

$$\bar{I}(i,j) = \frac{1}{N}\sum_{k=0}^{N-1} I_k(i,j)$$

$$\sigma(i,j) = \sqrt{\frac{1}{N-1}\sum_{k=0}^{N-1}\left(\bar{I}(i,j) - I_k(i,j)\right)^2}$$

$$\sigma_{n} = \frac{1}{RC}\sum_{i=0}^{R-1}\sum_{j=0}^{C-1} \sigma(i,j)$$

How can we reduce noise?

Image acquisition noise due to light fluctuations and sensor noise can be reduced by acquiring a sequence of images and averaging them.

WHY?

Smoothing Filters

Smoothing Spatial Filters

They are used for blurring and noise reduction.

Blurring is performed as pre-processing to remove small detail or bridge curve gaps

Noise reduction can be done by linear or nonlinear filtering

Linear Smoothing Filters

They are simply averaging filters: they compute the average of the filters under the mask.

They reduce sharp transitions.

They are low pass filters.

Average Filter

Mask with positive entries, that sum 1.

Replaces each pixel with an average of its neighborhood.

If all weights are equal, it is called a BOX filter.

F

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

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

O

X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

F

1/9

1	1	1
1	1	1
1	1	1

$$1/9.(10\times1 + 11\times1 + 10\times1 + 9\times1 + 10\times1 + 11\times1 + 10\times1 + 9\times1 + 10\times1) =$$

$$1/9.(\,90) = 10$$

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

X	X	X	X	X	X
X	10	7	4	1	X
X					X
X					X
X					X
X	X	X	X	X	X

$$1/9.(10x1 + 0x1 + 0x1 + 11x1 + 1x1 + 0x1 + 10x1 + 0x1 + 2x1) =$$

$$1/9.(\ 34) = 3.7778$$

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

X	X	X	X	X	X
X	10	7	4	1	X
X					X
X					X
X				20	X
X	X	X	X	X	X

1/9.(10x1 + 9x1 + 11x1 + 9x1 + 99x1 + 11x1 + 11x1 + 10x1 + 10x1) =
1/9.(180) = 20

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

X	X	X	X	X	X
X	10	7	4	1	X
X					X
X			18		X
X				20	X
X	X	X	X	X	X

$$1/9.(10\times1 + 0\times1 + 2\times1 + 9\times1 + 10\times1 + 9\times1 + 11\times1 + 9\times1 + 99\times1) =$$

$$1/9.(159) = 17.6667$$

Does it reduce noise?

Intuitively, takes out small variations.

Consider the Image pixel values under the mask and the corresponding output at the center:

$$I_i = \hat{I}_i + N_i \quad i = 1, \ldots, mn$$

$$O = \frac{1}{mn} \sum_{i}^{mn} (\hat{I}_i + N_i)$$

Is the output better? How?

Does it reduce noise?

Assume that the noise in the image is uncorrelated, zero mean, with stdev sigma.
The expected value of a pixel **before** filtering is:

$$E[I_i] = E[\hat{I}_i + N_i] = E[\hat{I}_i] + E[N_i] = \hat{I}_i + 0 = \hat{I}_i$$

$$i = 1, \ldots, mn$$

Does it reduce noise?

Assume that the noise in the image is uncorrelated, zero mean, with stdev sigma.
The expected value of a pixel **after** filtering is:

$$\begin{aligned}
E[O] &= E[\frac{1}{mn}\sum_{i}^{mn}(\hat{I}_i + N_i)] \\
&= \frac{1}{mn}\sum_{i}^{mn} E[\hat{I}_i] + \frac{1}{mn}\sum_{i}^{mn} E[N_i] \\
&= \frac{1}{mn}\sum_{i}^{mn} \hat{I}_i + 0 \\
&= \frac{1}{mn}\sum_{i}^{mn} \hat{I}_i
\end{aligned}$$

Does it reduce noise?

Assume that the noise in the image is uncorrelated, zero mean, with stdev sigma.

The variance of a pixel **before** filtering is:

$$E[(I_i - E[I_i])^2] \quad = \quad E[(\hat{I}_i + N_i - \hat{I}_i)^2] = E[N_i^2] = \sigma^2$$

$$i = 1, \ldots, mn$$

Does it reduce noise?

Assume that the noise in the image is uncorrelated, zero mean, with stdev sigma.

The variance of the pixel **after** filtering is:

$$
\begin{aligned}
E[(O - E[O])^2] &= E[(\frac{1}{mn}\sum_i^{mn}(\hat{I}_i + N_i) - \frac{1}{mn}\sum_i^{mn}(\hat{I}_i))^2] \\
&= \frac{1}{(mn)^2}E[(\sum_i^{mn} N_i)^2] \\
&= \frac{1}{(mn)^2}mn\sigma^2 = \frac{\sigma^2}{mn}
\end{aligned}
$$

How big should the mask be?

The bigger the mask,

- more neighbors contribute.
- smaller noise variance of the output.
- bigger noise spread.
- more blurring.
- more expensive to compute.

Weighted Average Filter

Gives more weight at the central pixel and less weights to the neighbors.

The farther away the neighbors, the smaller the weight.

Less blurring of edges

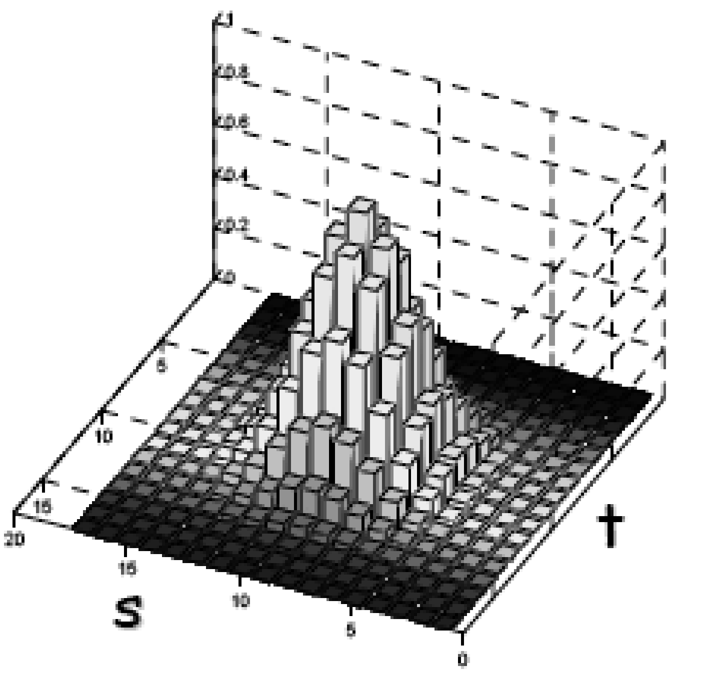
$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$g(x, y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)}$$

Gaussian Filter

A particular case of weighted averaging:

The coefficients are a 2D Gaussian.



$$w(s,t) = Ke^{\frac{s^2+t^2}{2\sigma^2}}$$

(0,0) is the center of the mask

σ determines how fast the weights decay

K is s.t. the sum of the coefficients is 1

How big should the mask be?

The std. dev of the Gaussian σ determines the amount of smoothing.

The samples should adequately represent a Gaussian

For a 98.76% of the area, we need

$m = 5\sigma$

$5.(1/\sigma) \leq 2\pi \Rightarrow \sigma \geq 0.796$, $m \geq 5$

Efficient Implementation

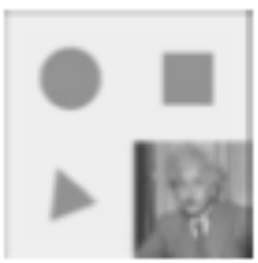
Both, the BOX filter and the Gaussian filter are separable:

- First convolve each row with a 1D filter
- Then convolve each column with a 1D filter.

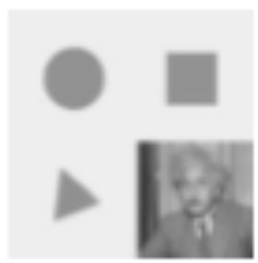
Separable Filters

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad \frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} \qquad \frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$



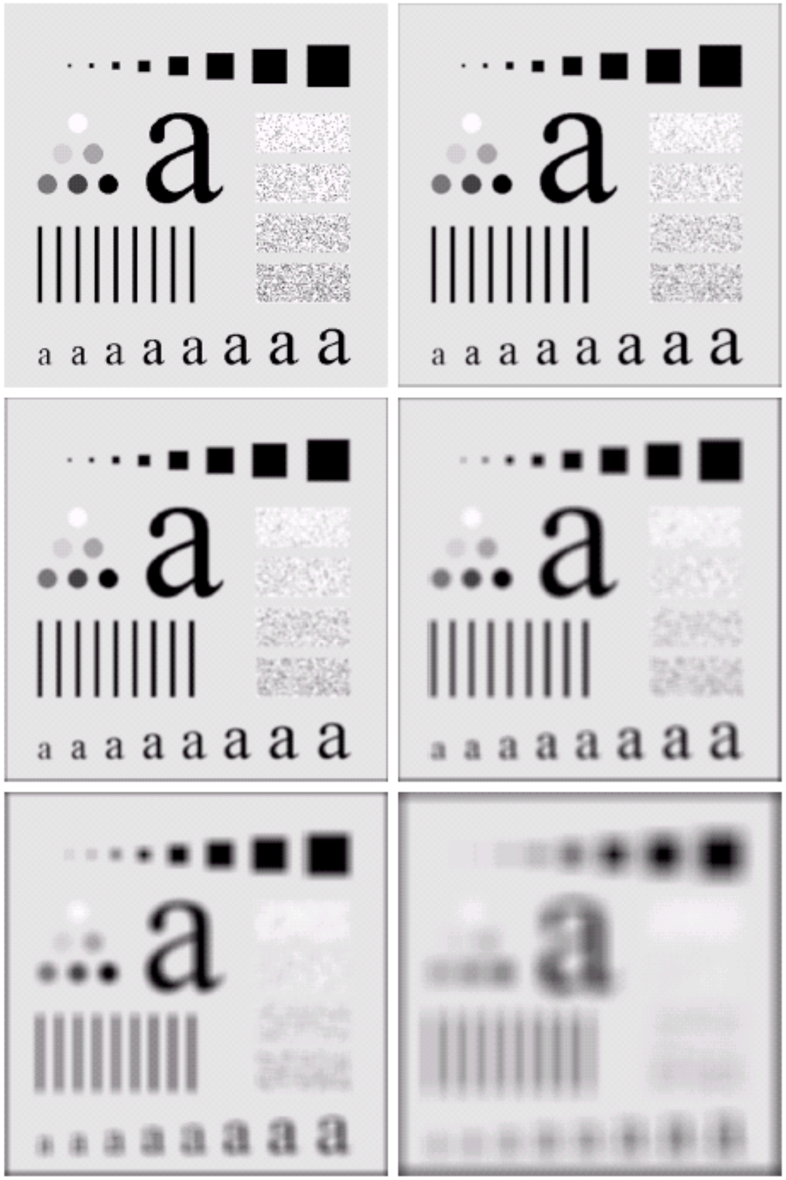
(a) box, $K = 5$



(b) bilinear

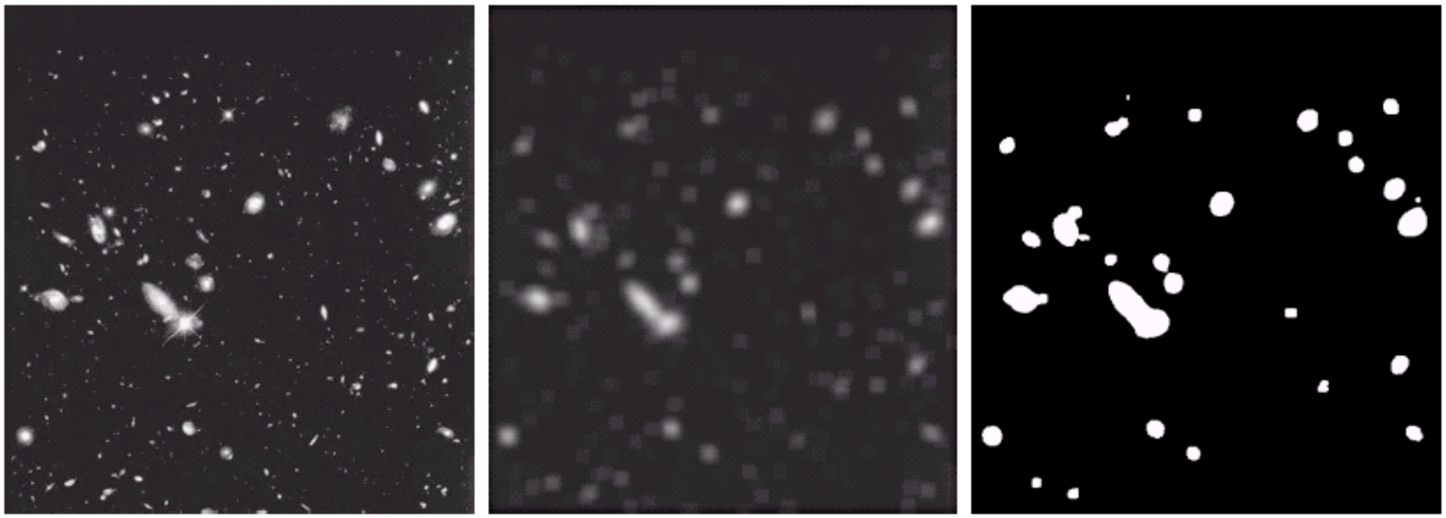
(c) "Gaussian"

Effects of increasing mask size



a b
c d
e f

FIGURE 3.35 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $n = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their gray levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.



a b c

FIGURE 3.36 (a) Image from the Hubble Space Telescope. (b) Image processed by a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Efficient Implementation: Integral Image

If an image is going to be repeatedly convolved with different box filters a pre-computed summed area table can save computations for future use.

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l)$$

3	2	7	2	3
1	5	1	3	4
5	1	3	5	1
4	3	2	1	6
2	4	1	4	8

Efficient Implementation: Integral Image

If an image is going to be repeatedly convolved with different box filters a pre-computed summed area table can save computations for future use.

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l)$$

3	2	7	2	3
1	5	1	3	4
5	1	3	5	1
4	3	2	1	6
2	4	1	4	8

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

$$s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(k,l)$$

Efficient Implementation: Integral Image

If an image is going to be repeatedly convolved with different box filters a pre-computed summed area table can save computations for future use.

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l)$$

3	2	7	2	3
1	5	1	3	4
5	1	3	5	1
4	3	2	1	6
2	4	1	4	8

(a) S = 24

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

(b) s = 28

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

(c) S = 24

$$S = 48 - 14 - 13 + 3$$

Limitations of averaging

Signal frequencies shared with noise are lost, resulting in blurring.

Impulsive noise is diffused but not removed.

It spreads pixel values, resulting in blurring.

Non-linear Filtering

Replace each pixel with the MEDIAN value of all the pixels in the neighborhood.

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

O

X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

10,11,10,9,10,11,10,9,
10

sort →

9,9,10,10,10,10,10,11,11

median

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

O

X	X	X	X	X	X
X	10	10	1	1	X
X					X
X					X
X					X
X	X	X	X	X	X

11,10,0,10,11,1,9,10
,0

sort →

0,0,1,9,10,10,10,11,11

median

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	$\boxed{99}$	11
10	9	9	11	10	10

O

X	X	X	X	X	X
X	10				X
X					X
X			9		X
X				$\boxed{10}$	X
X	X	X	X	X	X

10,9,11,9,99,11,11,10,
10

sort →

9,9,10,10,10,11,11,11,99

median

Median Filter Properties

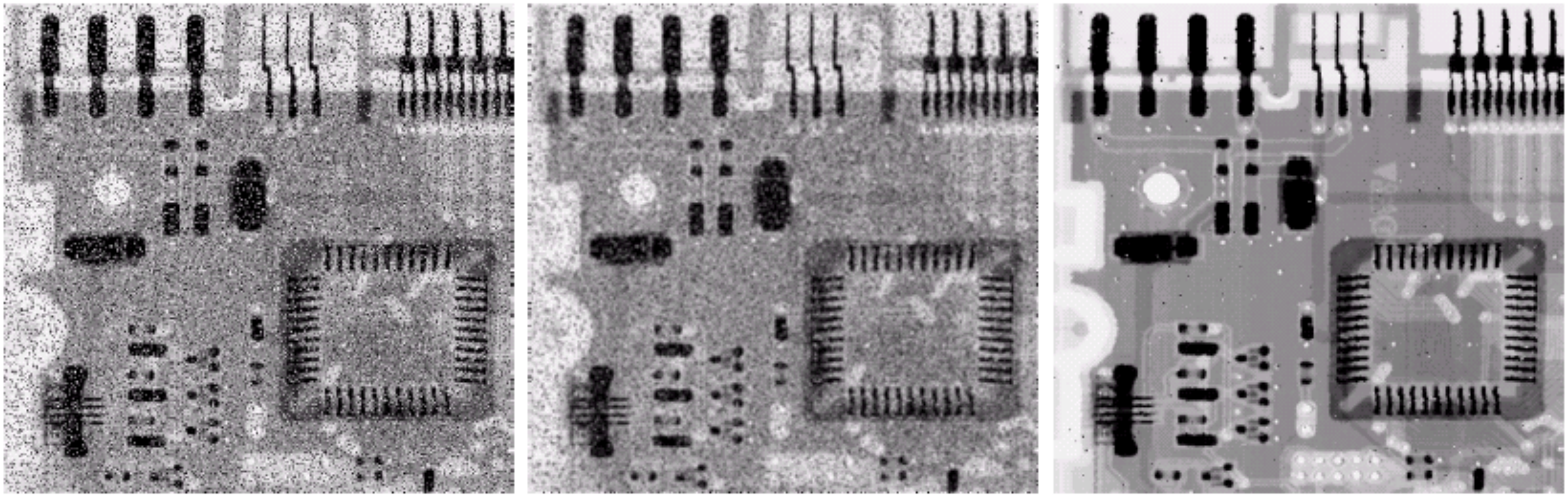
Non-linear

Does not spread the noise

Can remove spike noise

Expensive to run

Median Filter



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3 × 3 averaging mask. (c) Noise reduction with a 3 × 3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)