CS 3630!

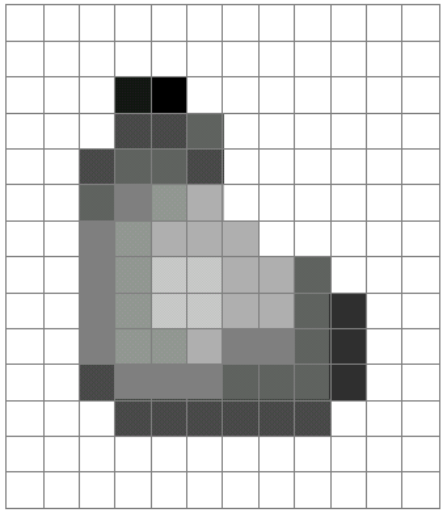
Lecture16: Segmentation

So Many Stolen PPT slides from friends!!

Image Processing: A Quick Review

Grayscale Images

A grid (matrix) of intensity values

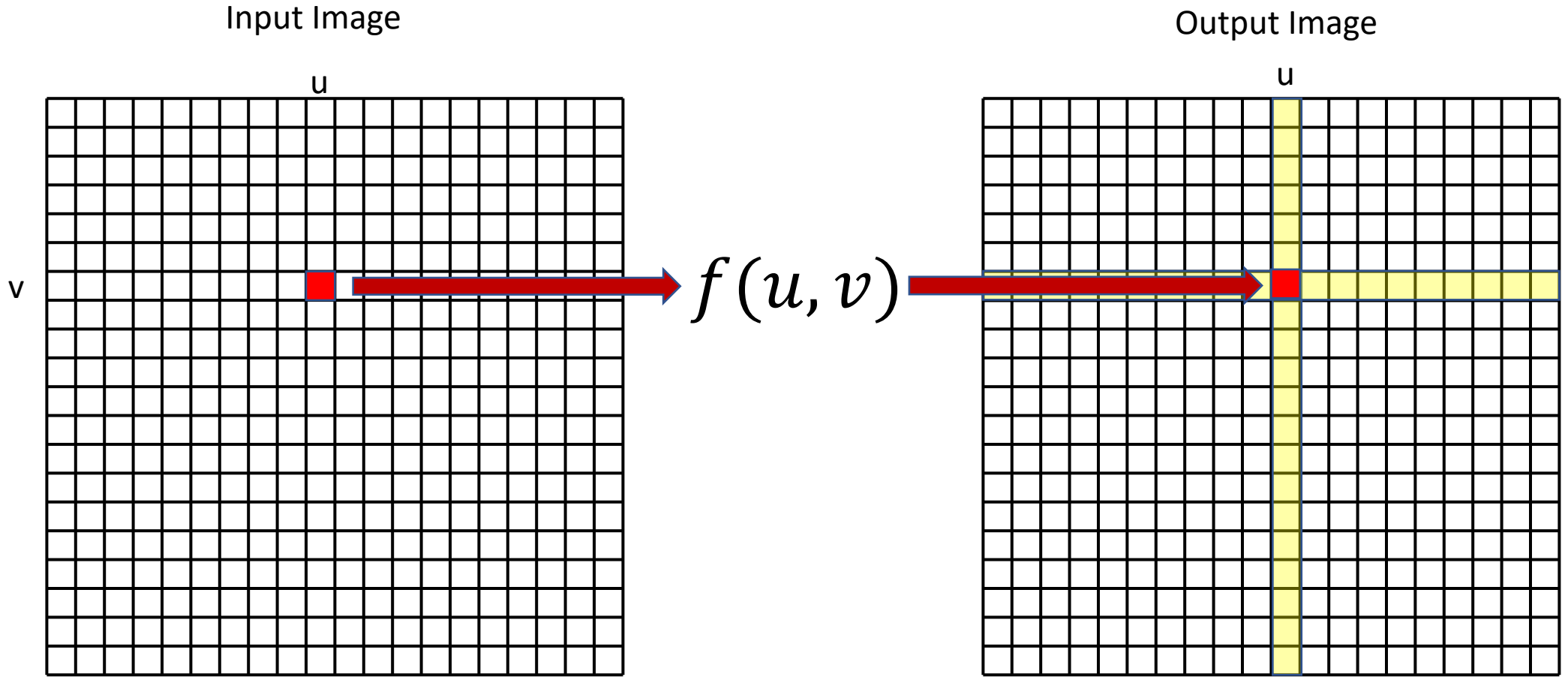


=

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(common to use one byte per value: 0 = black, 255 = white)

Monadic operators for filtering



Monadic operations take a single pixel as input and give a single pixel output, and do not consider neighboring pixel values

Example of some monadic image operations

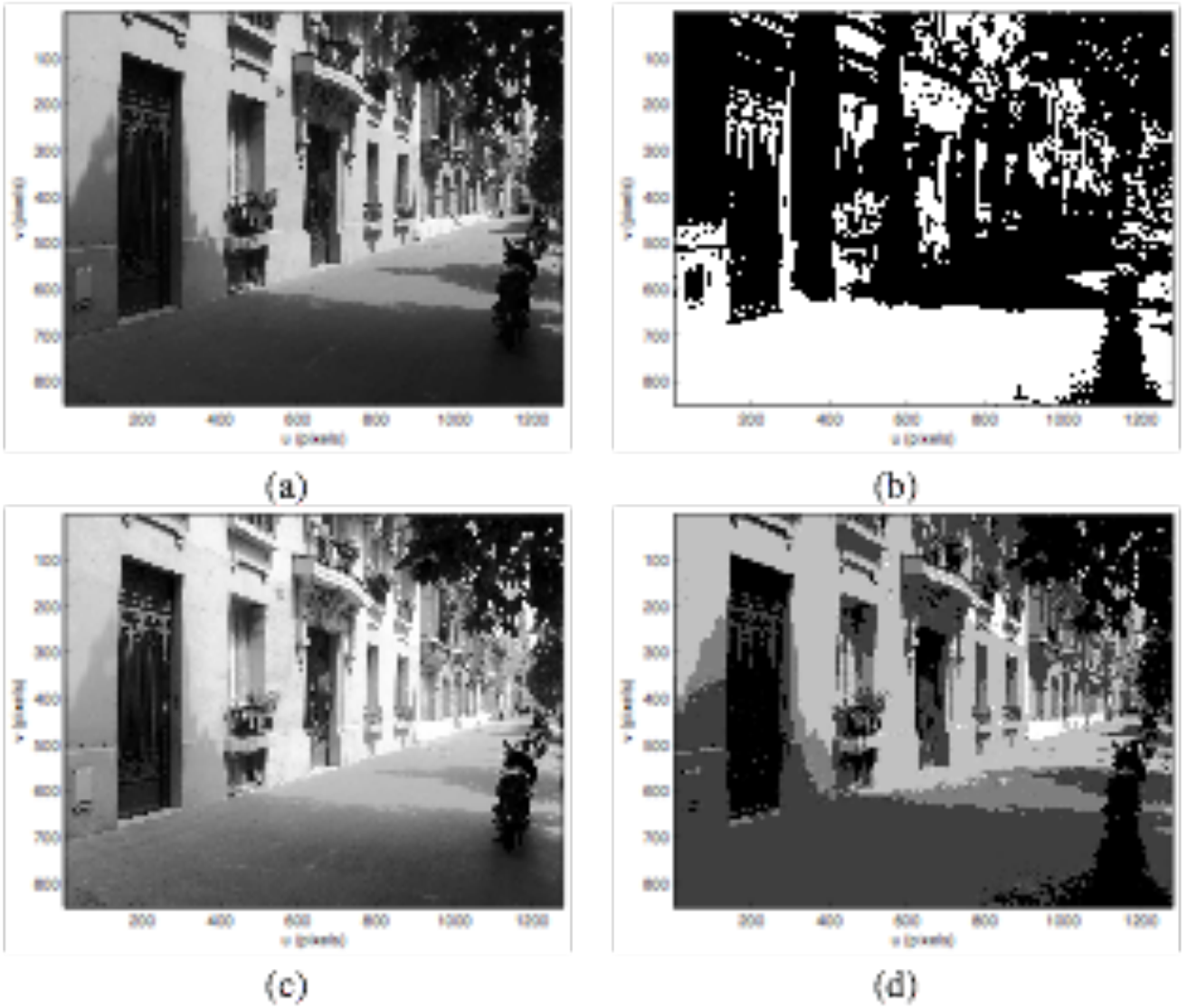
a original

b thresholding (all values greater than some threshold τ are mapped to white, and values lower than the threshold are mapped to black)

lets look at this one more closely

c histogram normalized

d posterization (conversion of a continuous gradation of tone to several regions of fewer tones, with abrupt changes from one tone to another)



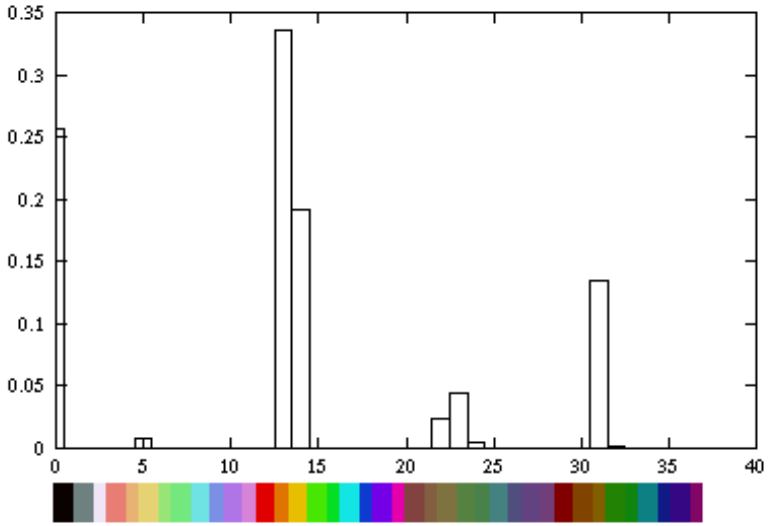
Color Histogram

- A histogram array, $\mathcal{H}$, represents the distribution of colors (or *intensities* if grayscale) in the image $\mathcal{I}$.
- For the i^{th} color C_i, the value $\mathcal{H}[i]$ gives the number of pixels of color C_i in image $\mathcal{I}$.

A simple algorithm

Let $C(p)$ denote the color label associated to pixel p

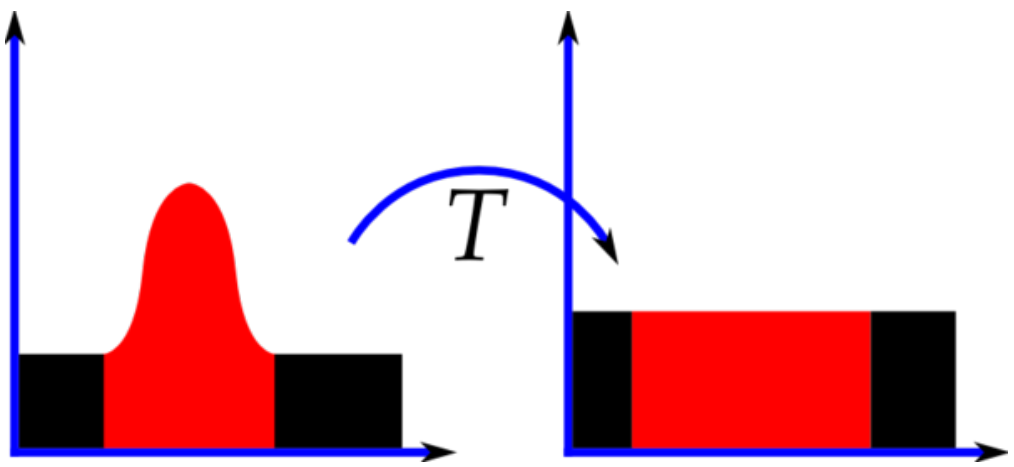
1. Initialize: $\mathcal{H}[i] \leftarrow 0$
2. For pixel $p \in \mathcal{I}$
3. $\mathcal{H}[C(p)] + +$



Histogram filter

- A histogram can be used to adjust (redistribute) the intensity or color distribution of an image

- Usually increases the contrast of an image by effectively spreading out the most frequent intensity values
- Useful in images with backgrounds and foregrounds that are both bright or both dark



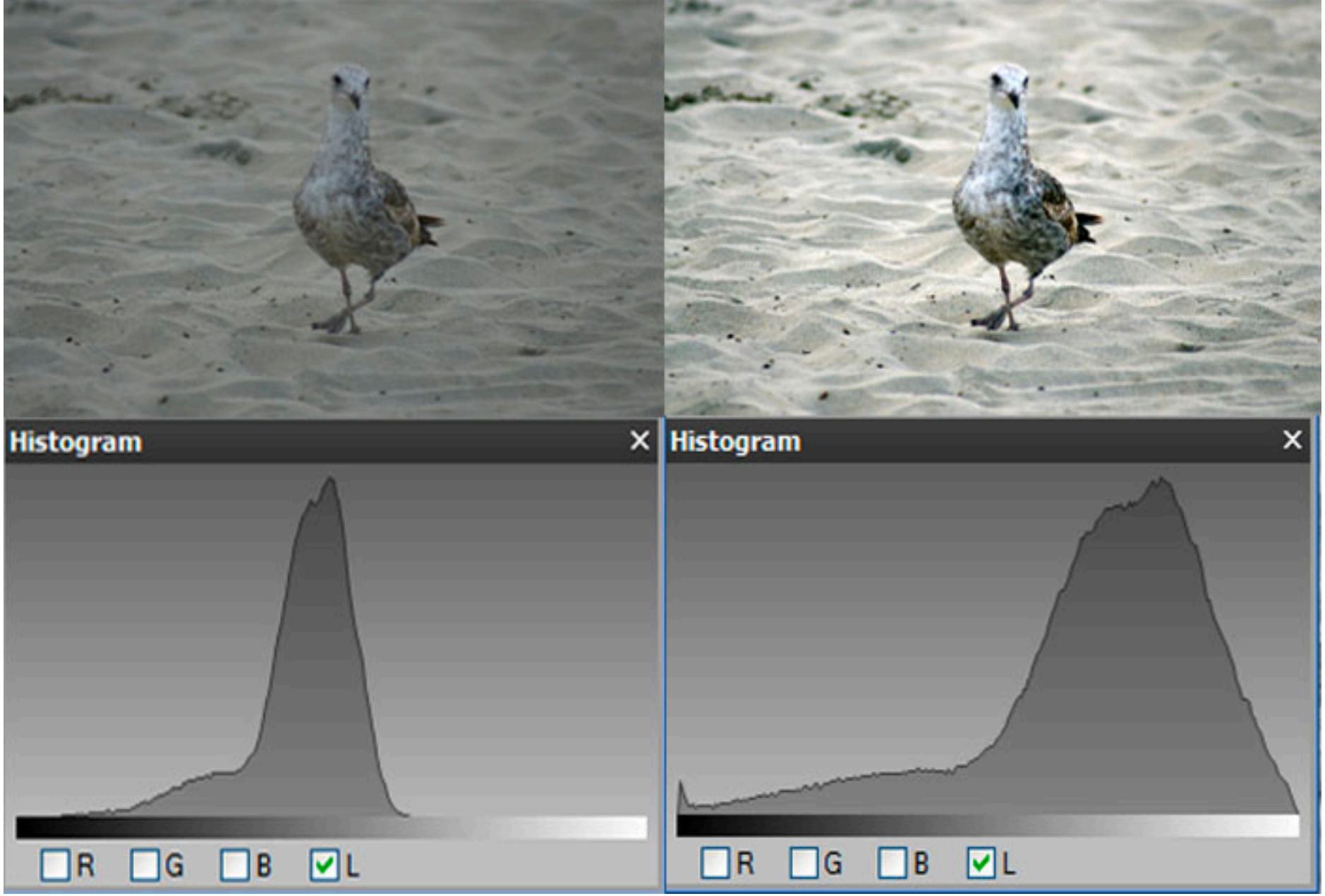
Histograms of an image before and after equalization.

Image Contrast

- Contrast is the difference between maximum and minimum pixel intensity in an image (or in a region of an image).

Low contrast

High contrast



Low contrast

High contrast

Capability provided by [skimage.exposure]

Local Operators

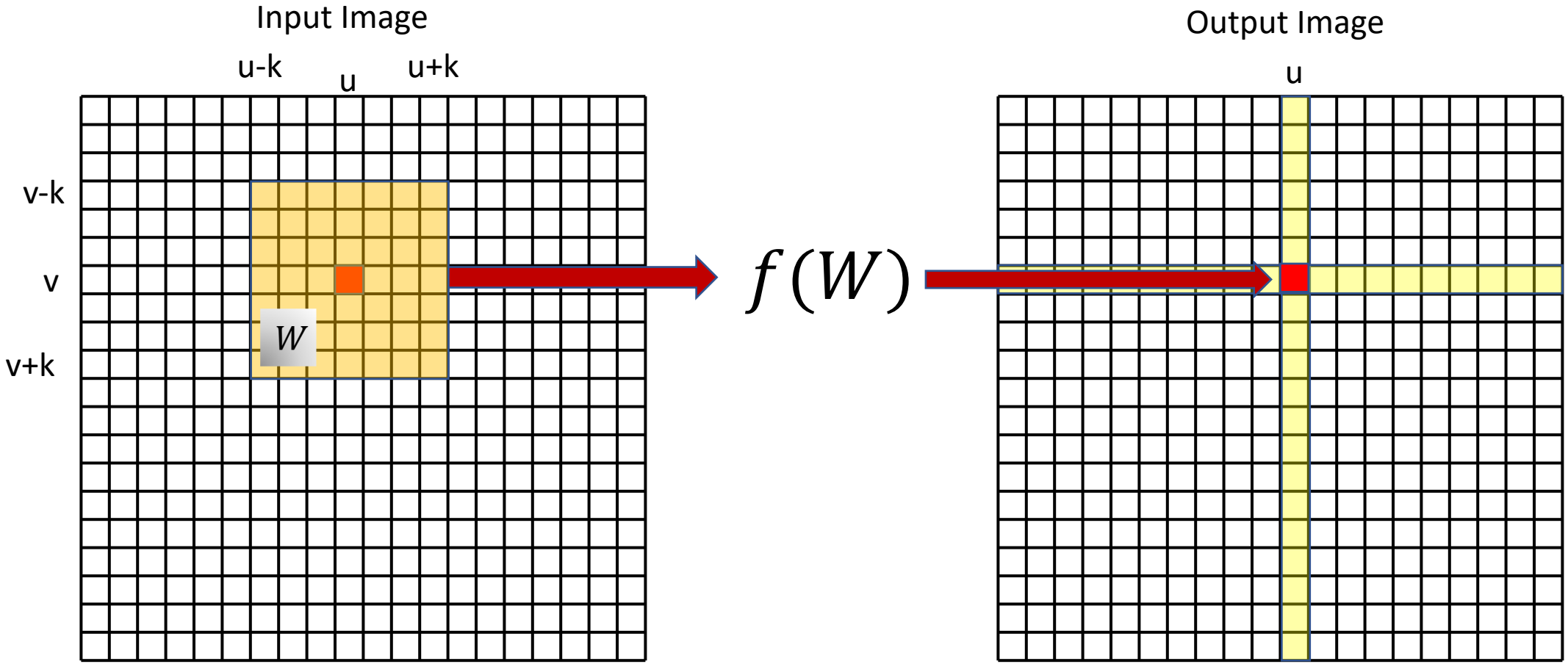
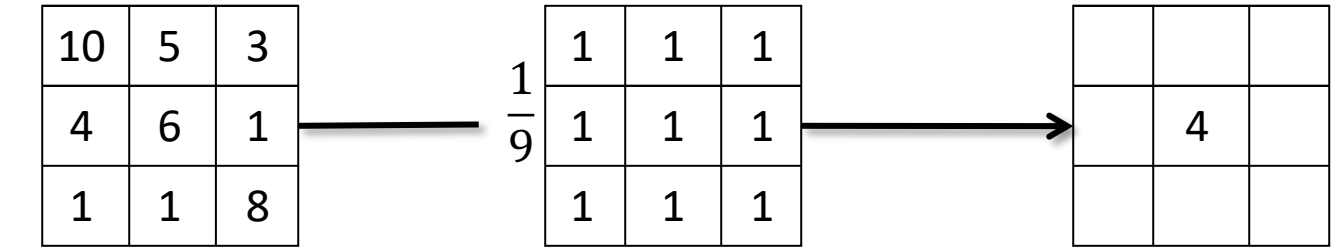


Image processing operations that use a region of pixels in the input image to determine the value of a single pixel in the output image

Example: Averaging using a linear filtering

- Replace each pixel by a linear combination of its neighbors
- The matrix of the linear combination is called the “kernel,” “mask”, or “filter”



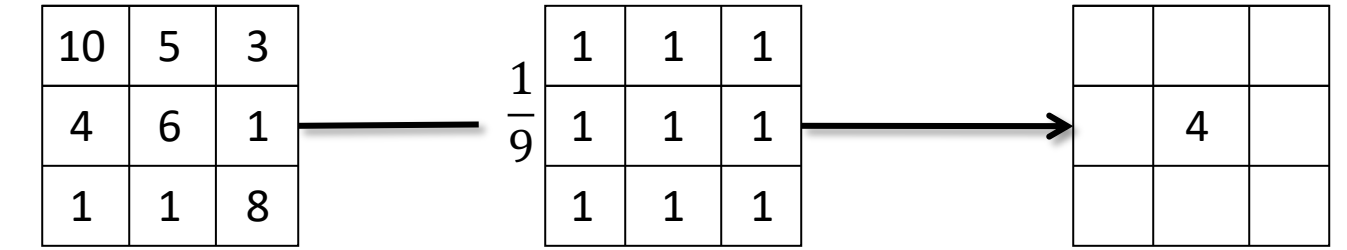
The effect is a blurring of the image.

Source: L. Zhang

Example: Averaging using a linear filtering

Let F be the image, H be the kernel (of size $2k+1$ by $2k+1$), and G be the output image

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

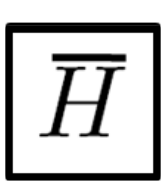


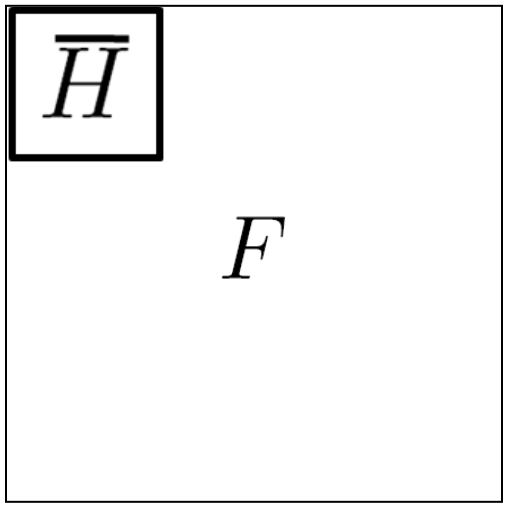
The effect is a blurring of the image.

Source: L. Zhang

Convolution

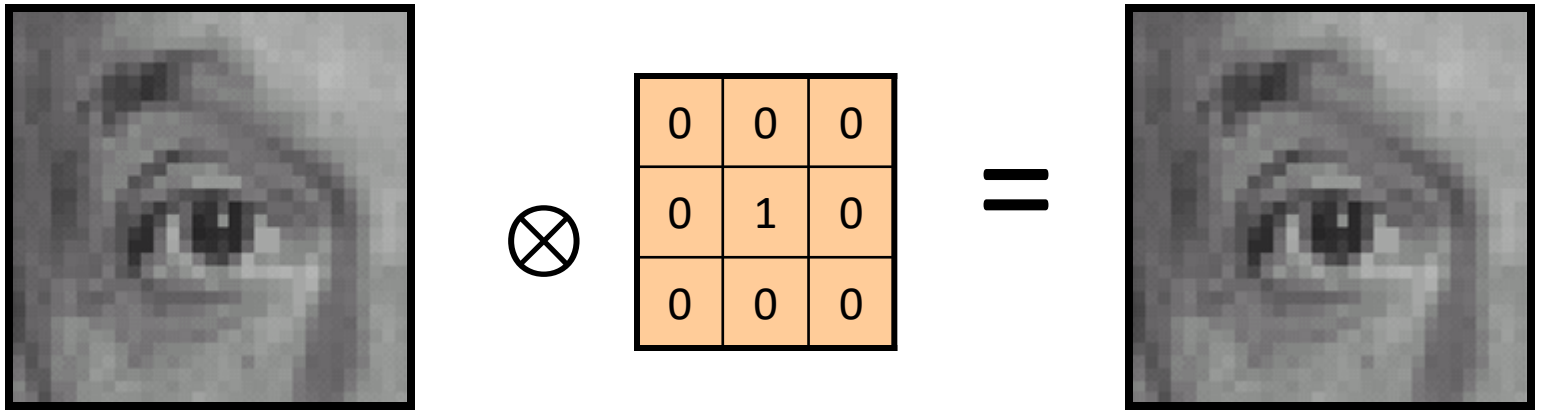
- Applying a kernel to an image in this way is called convolution





Adapted from F. Durand

Linear filters: examples

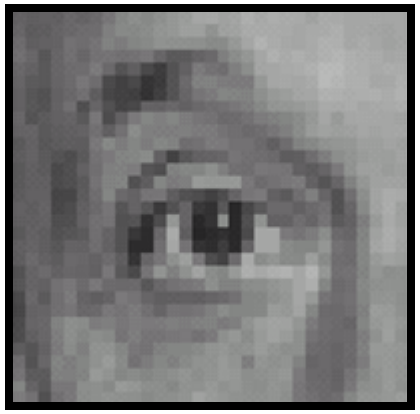


Original

Identical image

Source: D. Lowe

Linear filters: examples

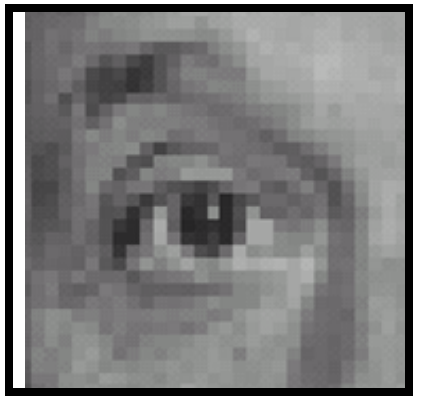


Original

$\otimes$

0	0	0
1	0	0
0	0	0

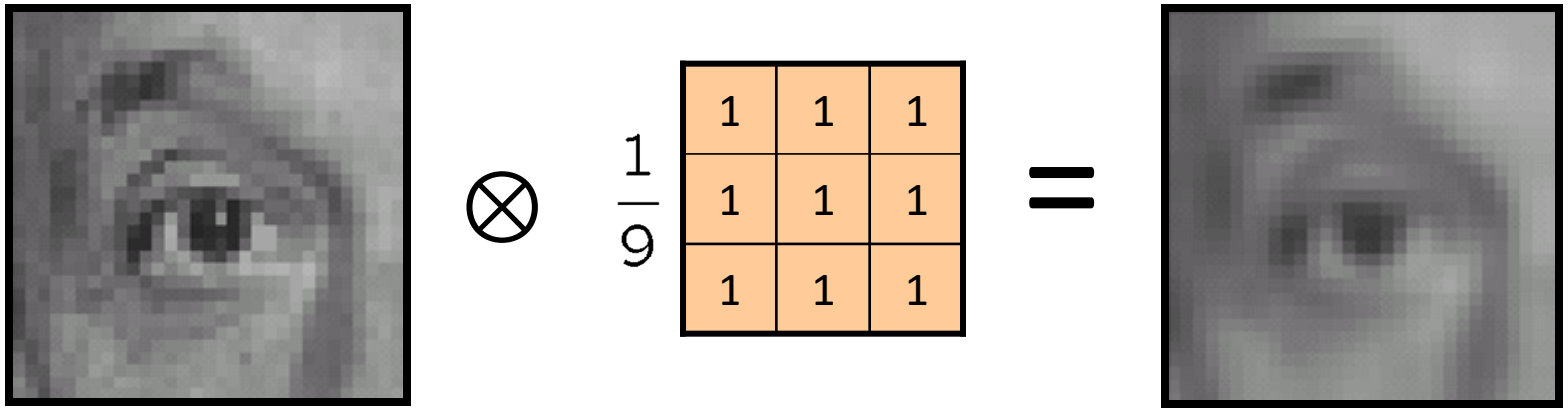
$=$



Shifted right
By 1 pixel

Source: D. Lowe

Linear filters: examples

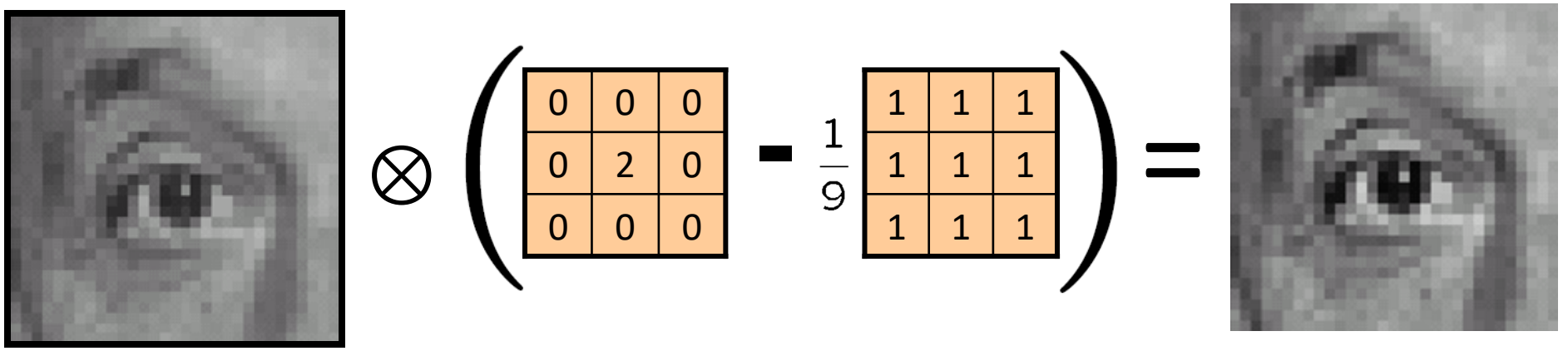


Original

Mean filter (blurring)

Source: D. Lowe

Linear filters: examples

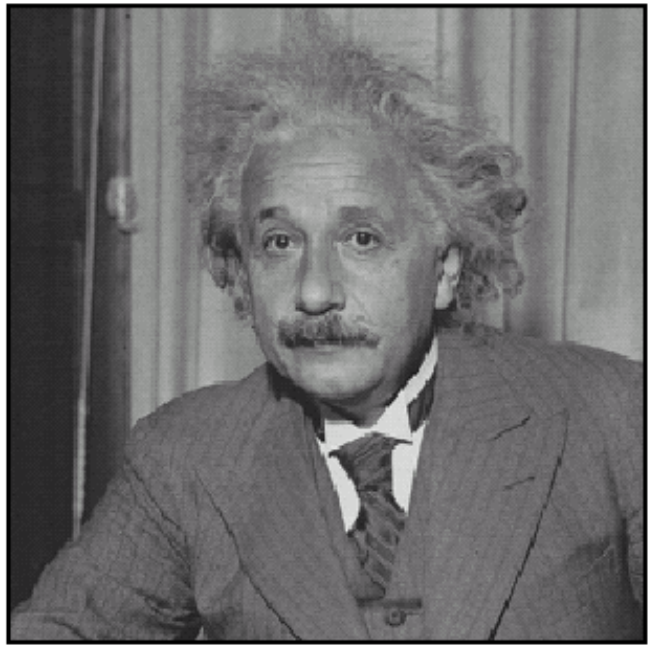


Original

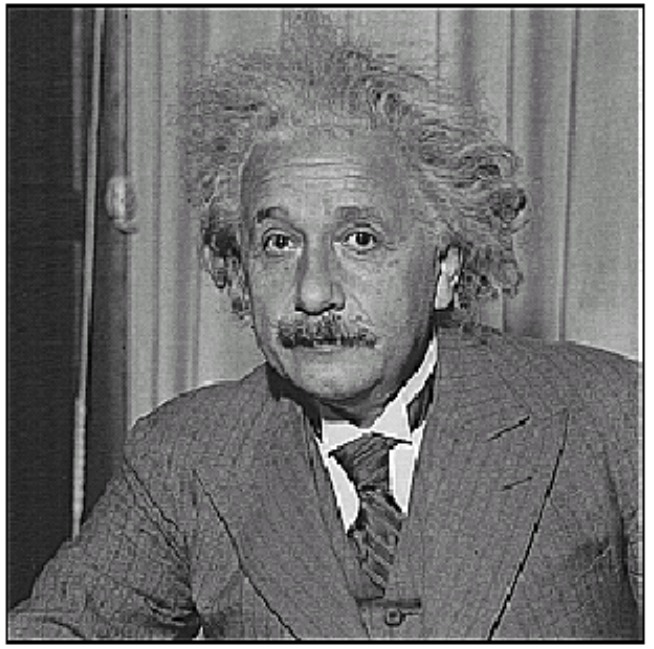
Sharpening filter
(accentuates edges)

Source: D. Lowe

Sharpening



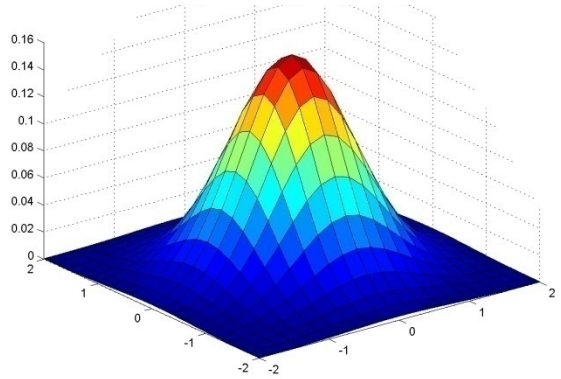
before

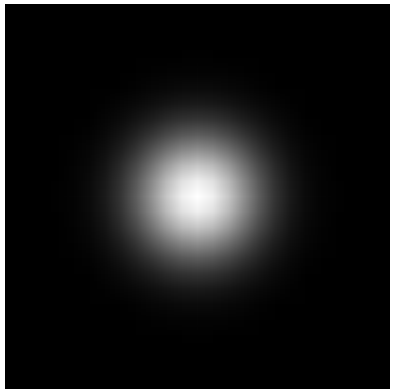


after

Source: D. Lowe

Gaussian Kernel





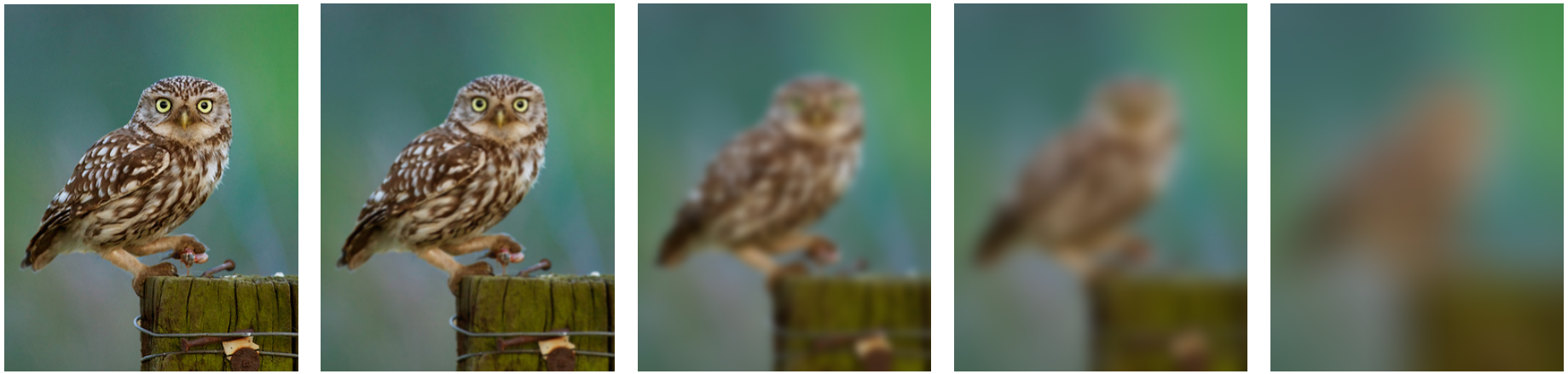
Approximated by: $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

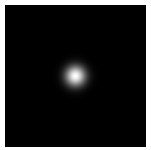
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Source: C. Rasmussen

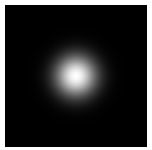
Gaussian filter



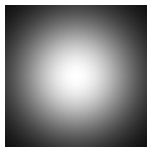
σ = 1 pixel



σ = 5 pixels



σ = 10 pixels



σ = 30 pixels

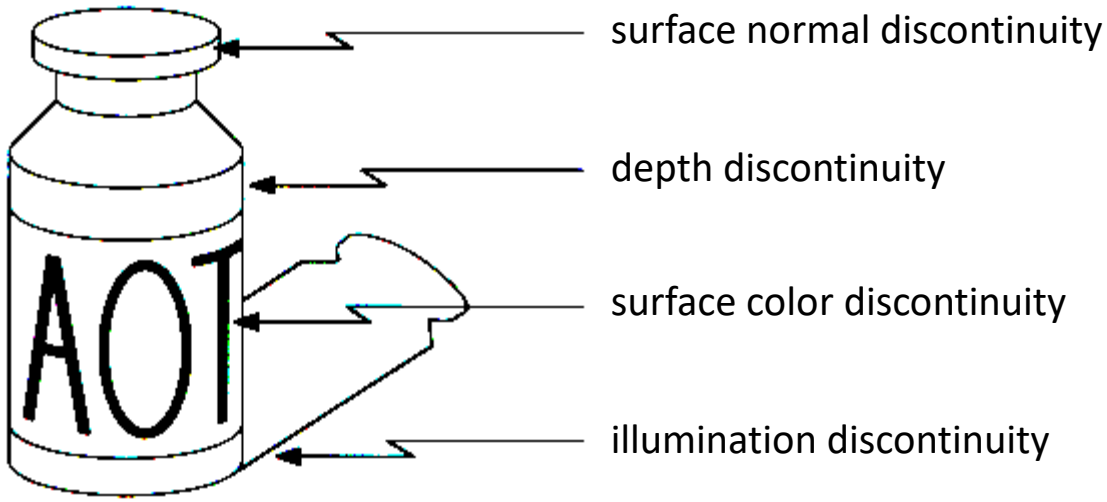
Edge detection

Winter in Kraków photographed by Marcin Ryczek

Edge Detection

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
- Intuitively, edges carry most of the semantic and shape information from the image



Sources: D. Lowe and S. Seitz

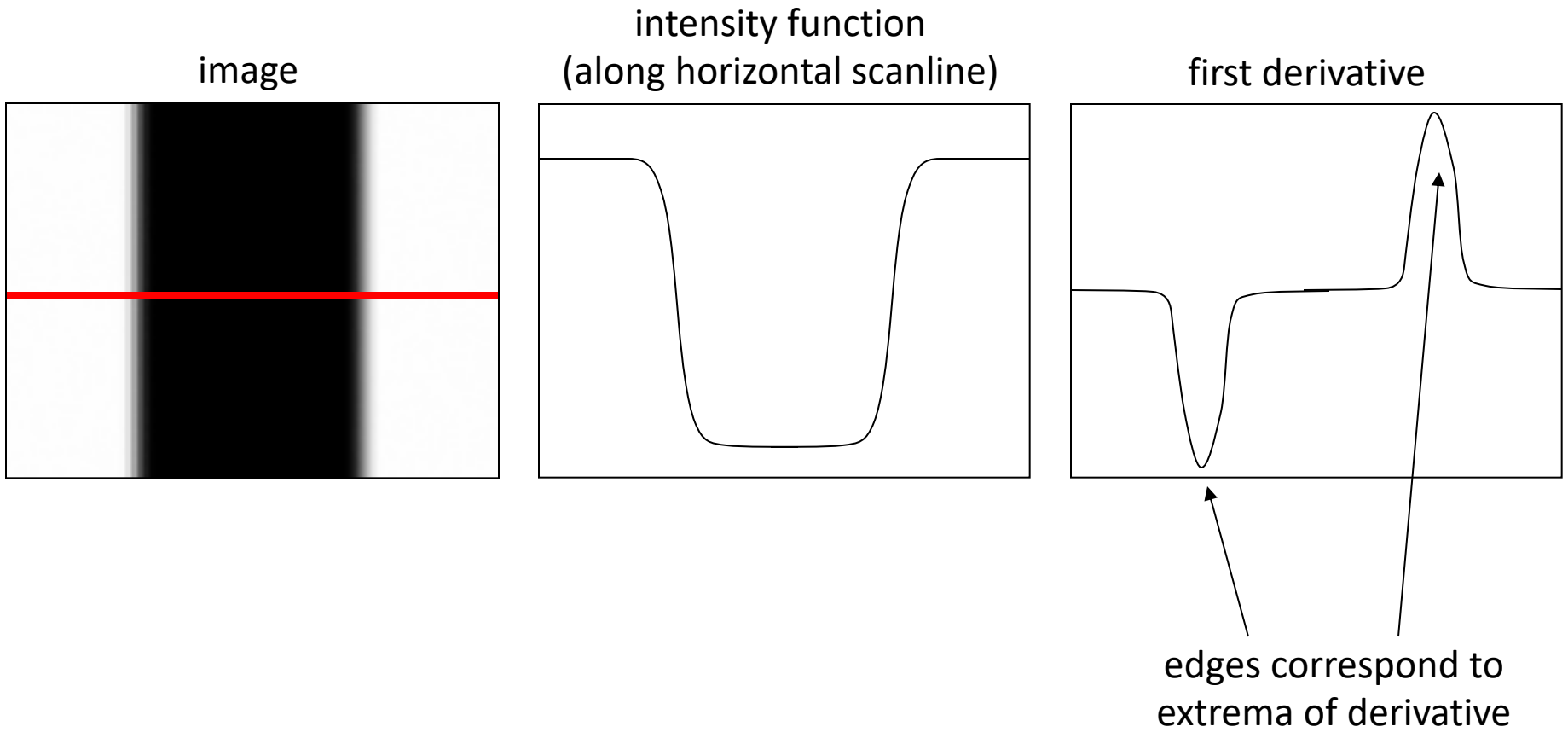
Edge detection

- **Ideal:** artist's line drawing

- **Reality:**

Edge detection

- An edge is a place of rapid change in the image intensity function



Derivatives with convolution

For 2D function f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

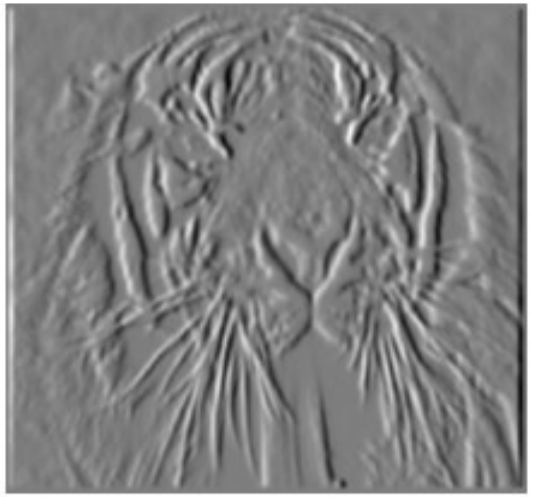
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

To implement the above as convolution, what would be the associated filter?

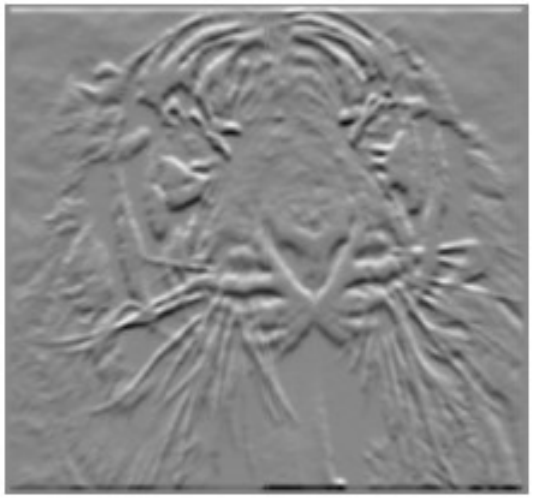
Source: K. Grauman

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$



-1	1



$$\frac{\partial f(x,y)}{\partial y}$$

-1
1

or

1
-1

Which shows changes with respect to x?

Finite difference filters

- Other approximations of derivative filters exist:

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Source: K. Grauman

Edge Detection

Through Convolution

Sobel:

-1	0	1
-2	0	2
-1	0	1

Prewitt:

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

Roberts:

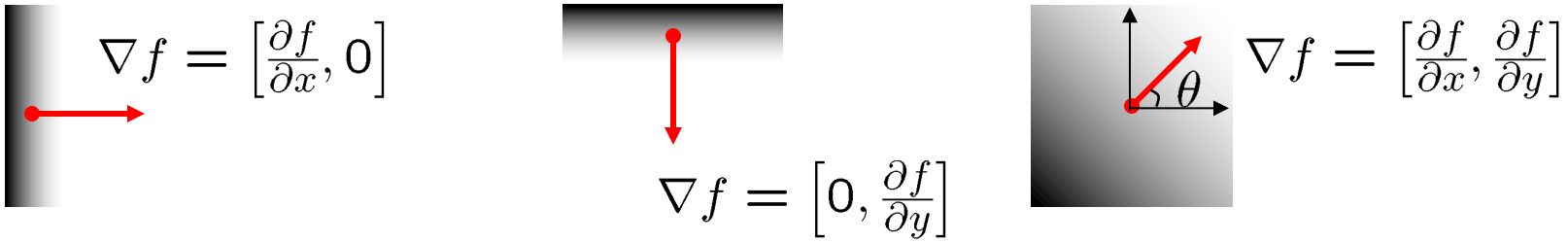
0	1
-1	0

Canny:

more complex multi-stage algorithm that uses a Gaussian filter and the intensity gradient in an image. One of the most widely used techniques.

Image gradient

The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



The gradient points in the direction of most rapid increase in intensity

- How does this direction relate to the direction of the edge?

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

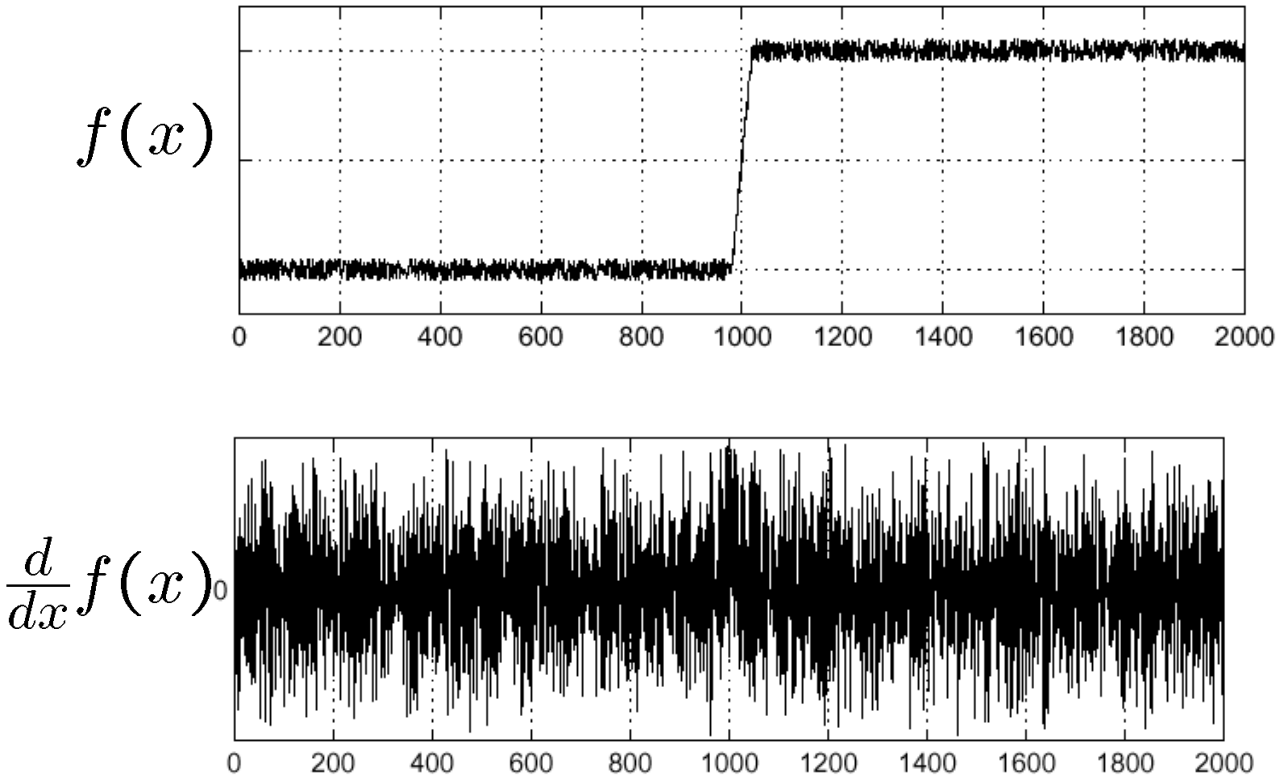
The edge strength is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

Effects of noise

Consider a single row or column of the image



Where is the edge?

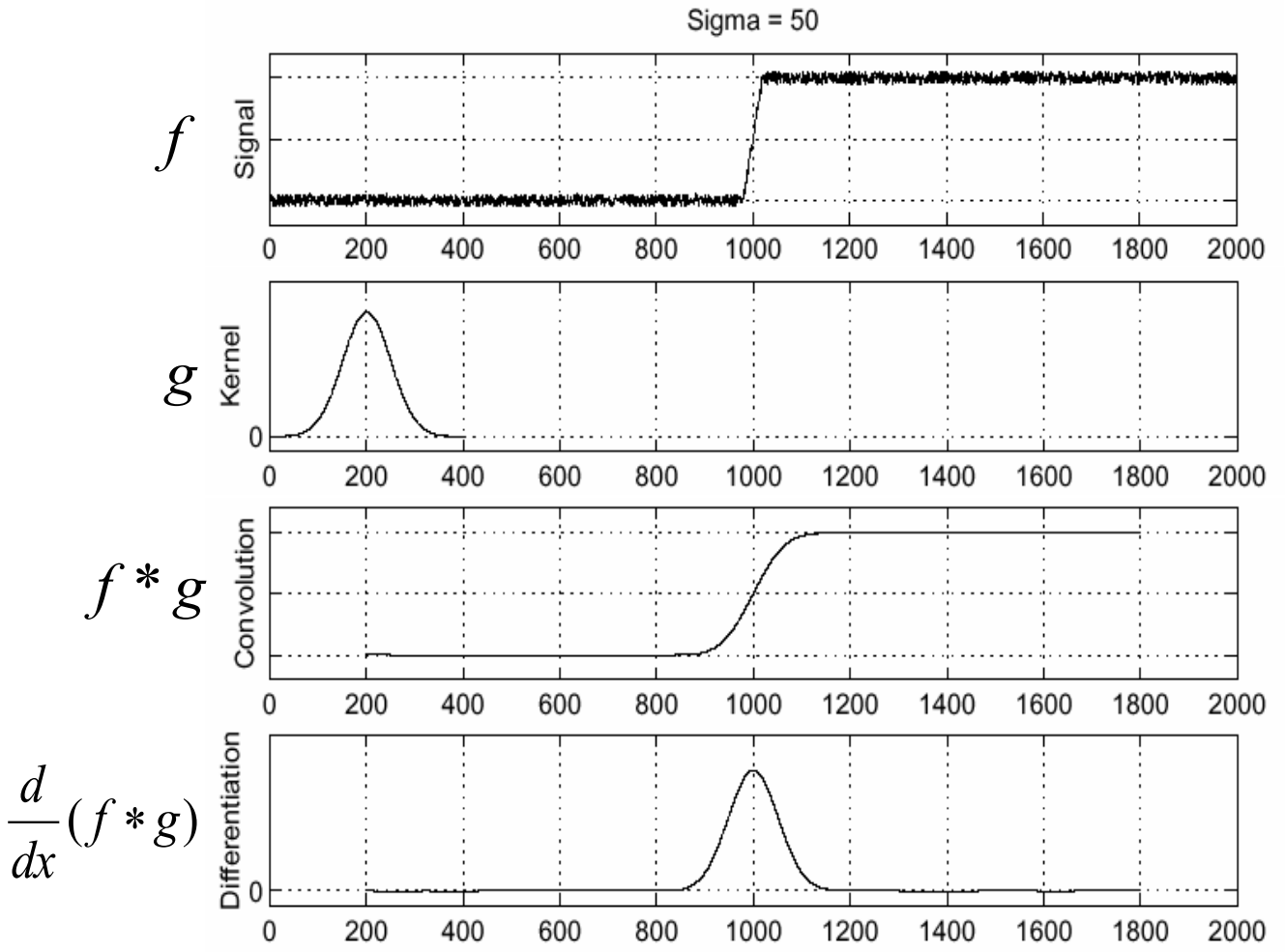
Source: S. Seitz

Gaussian Blurring and Edge Detection

Blurring the image slightly (Gaussian filter) can improve classification performance by smoothing away random noise in the image

Edge detection can help with some approaches to image classification, but it’s not always necessary

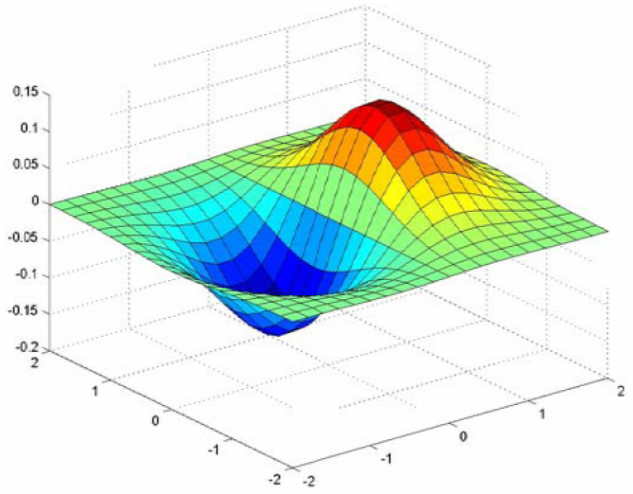
Solution: smooth first



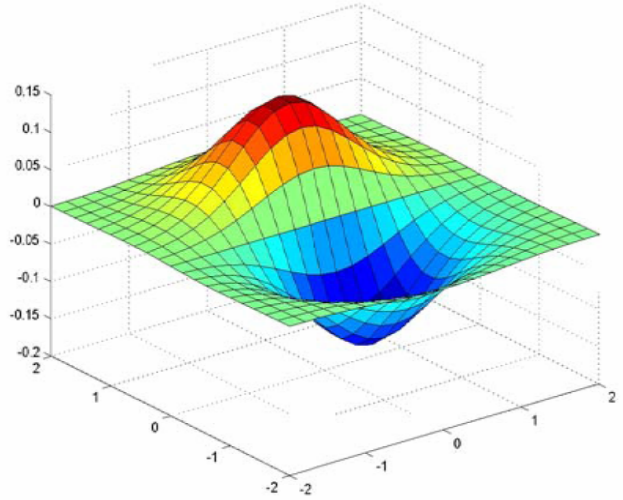
- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

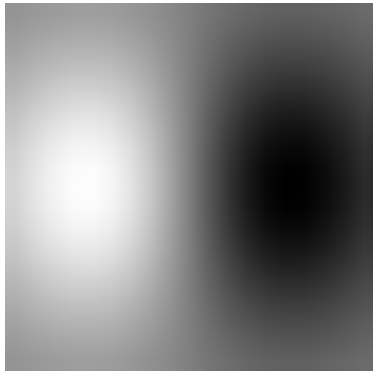
Derivative of Gaussian filters

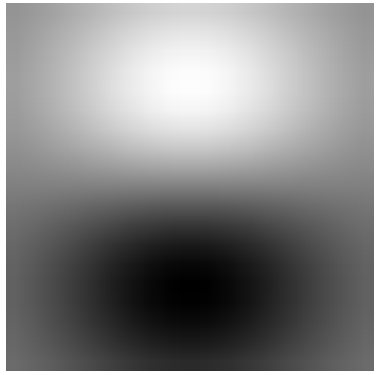


x-direction



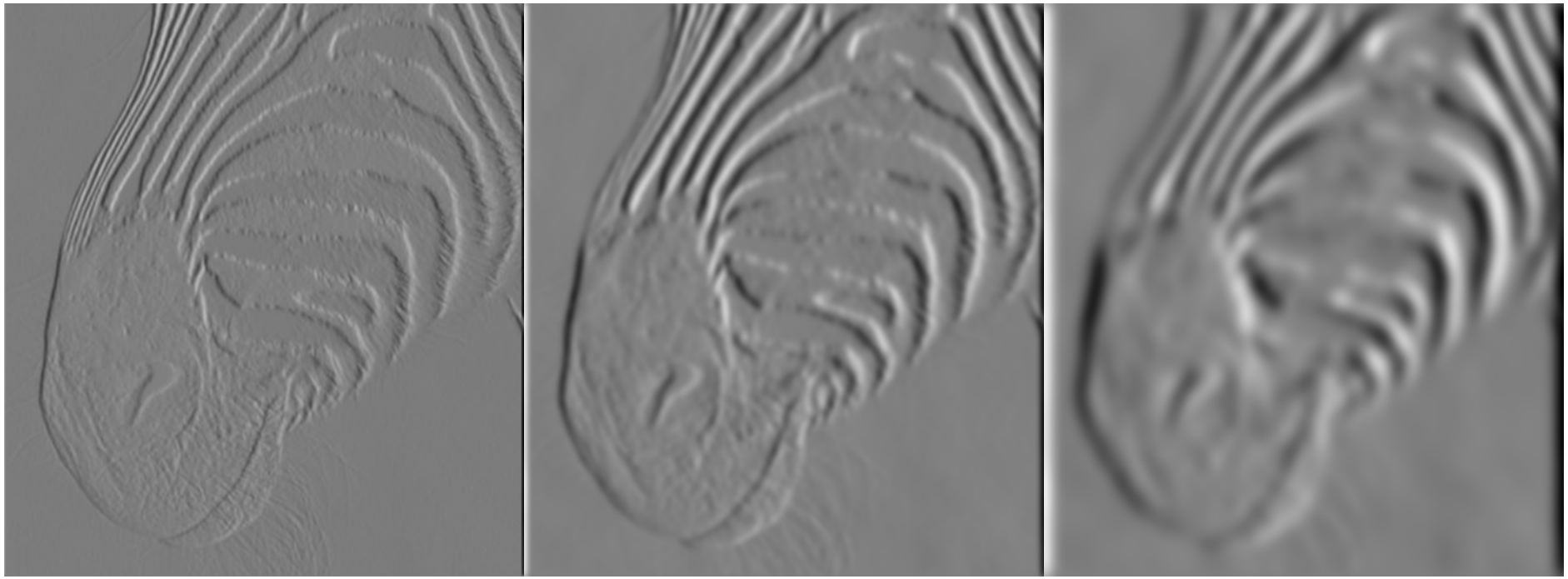
y-direction





Which one finds horizontal/vertical edges?

Scale of Gaussian derivative filter



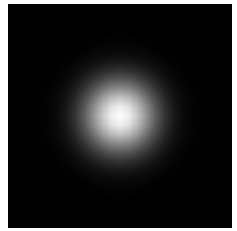
1 pixel 3 pixels 7 pixels

Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”

Source: D. Forsyth

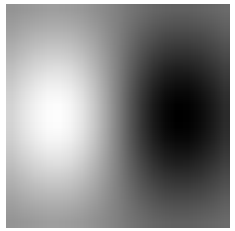
Review: Smoothing vs. derivative filters

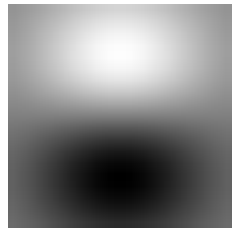
Smoothing filters



- Gaussian: remove “high-frequency” components; “low-pass” filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
  - **One:** constant regions are not affected by the filter

Derivative filters





- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
  - **Zero:** no response in constant regions

Building an edge detector

original image

final output

Building an edge detector

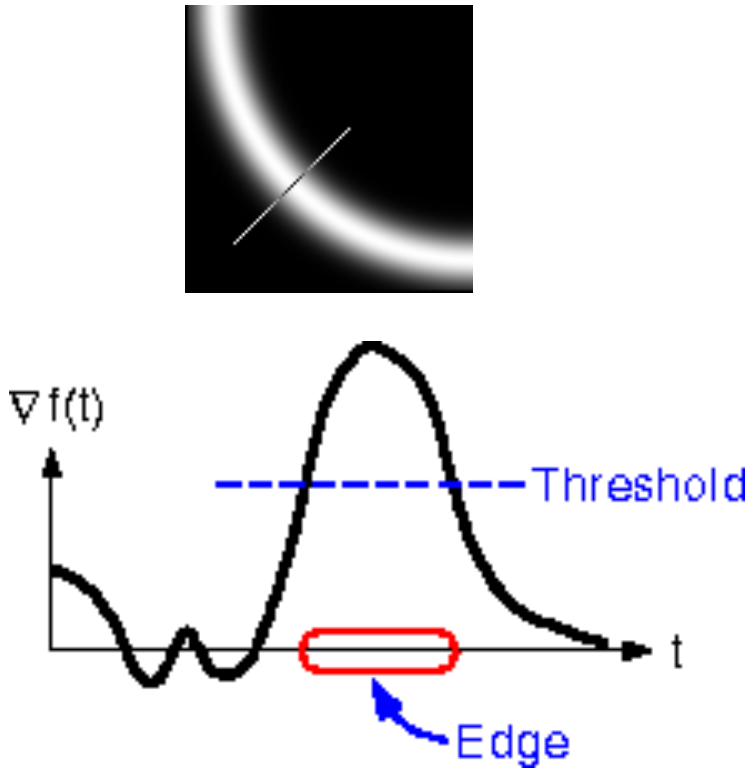
norm of the gradient

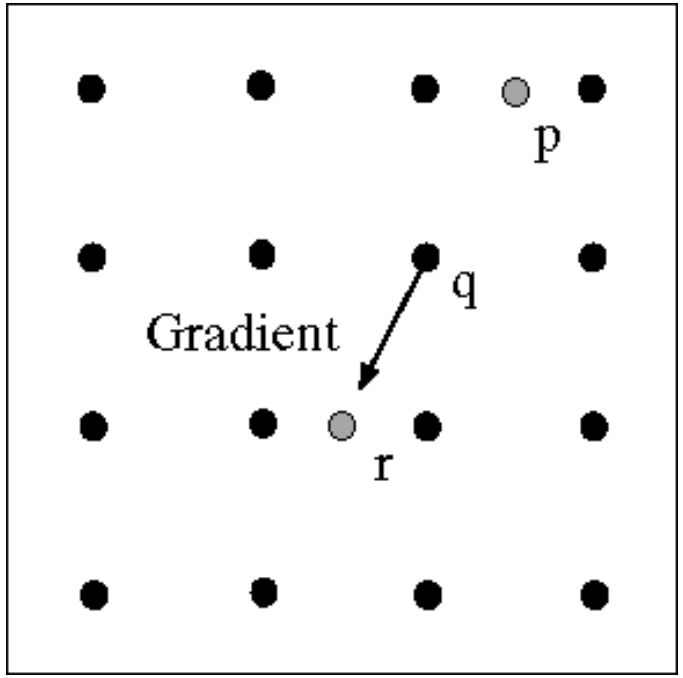
Building an edge detector

How to turn these thick regions of the gradient into curves?

Thresholded norm of the gradient

Non-maximum suppression





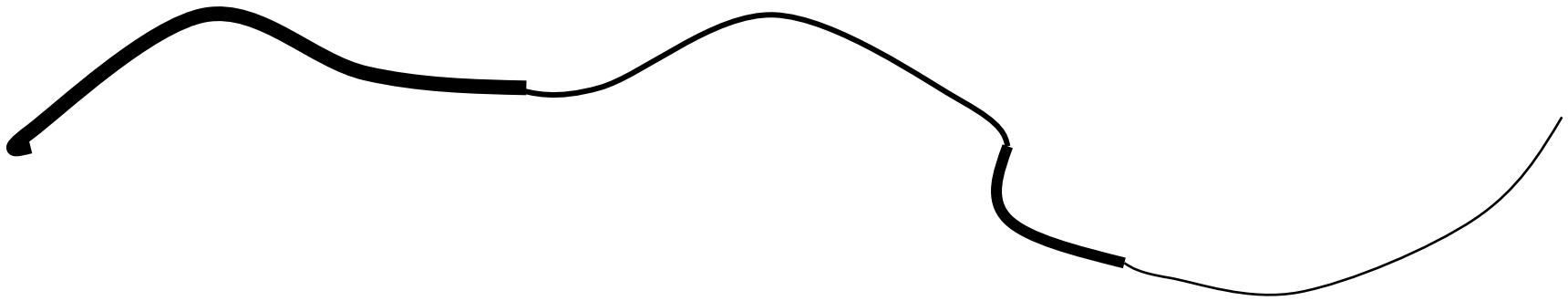
- For each location q above threshold, check that the gradient magnitude is higher than at neighbors p and r along the direction of the gradient
  - May need to interpolate to get the magnitudes at p and r

Non-maximum suppression

Another problem: pixels along this edge didn’t survive the thresholding

Hysteresis thresholding

Use a high threshold to start edge curves, and a low threshold to continue them.



Source: Steve Seitz

Hysteresis thresholding

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei

Recap: Canny edge detector

1. Compute x and y gradient images
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression**:
   - Thin wide “ridges” down to single pixel width
4. **Linking and thresholding** (**hysteresis**):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

MATLAB: `edge(image, 'canny');`

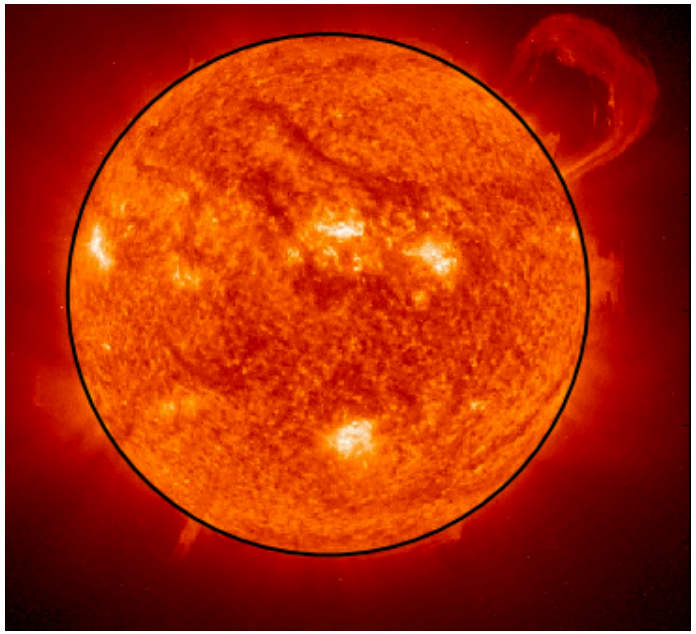
J. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

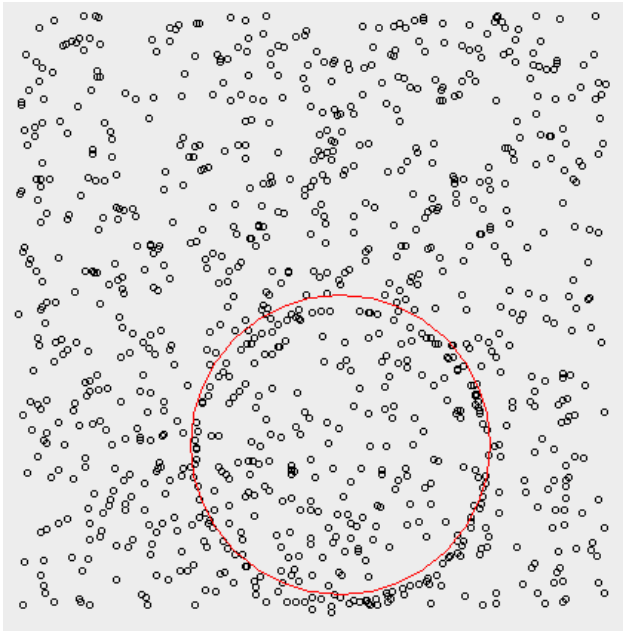
Image gradients vs. meaningful contours

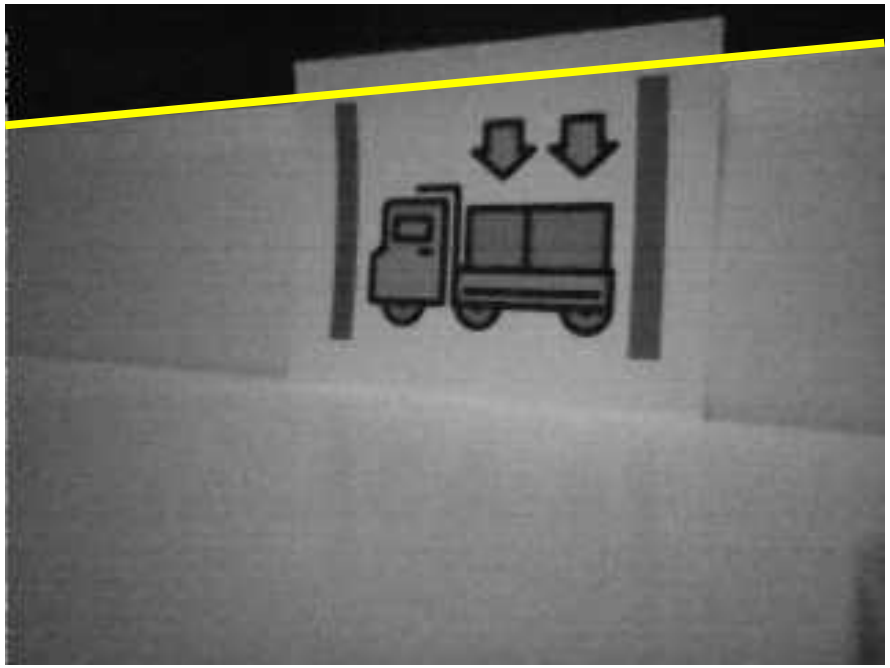
image | human segmentation | gradient magnitude

[Berkeley segmentation database]

RANSAC

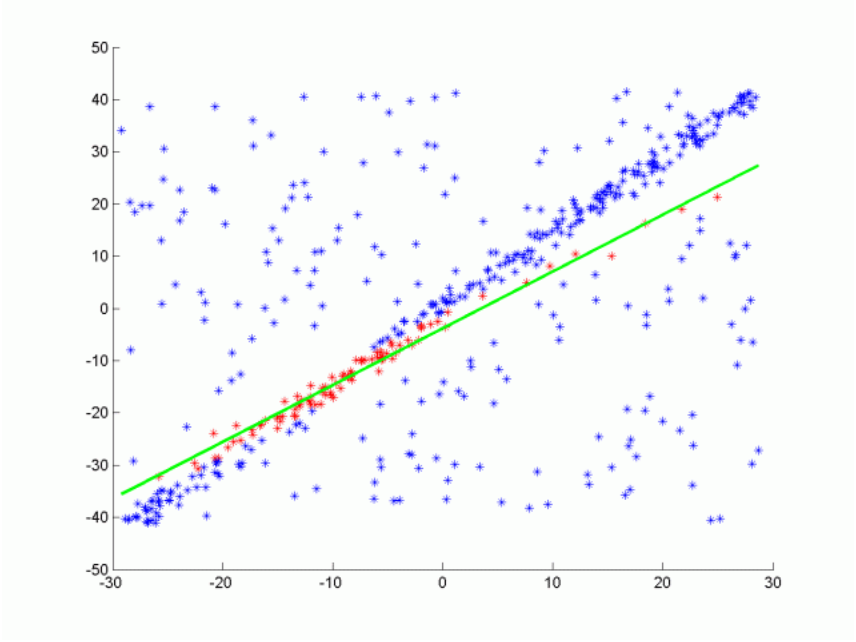






RANSAC

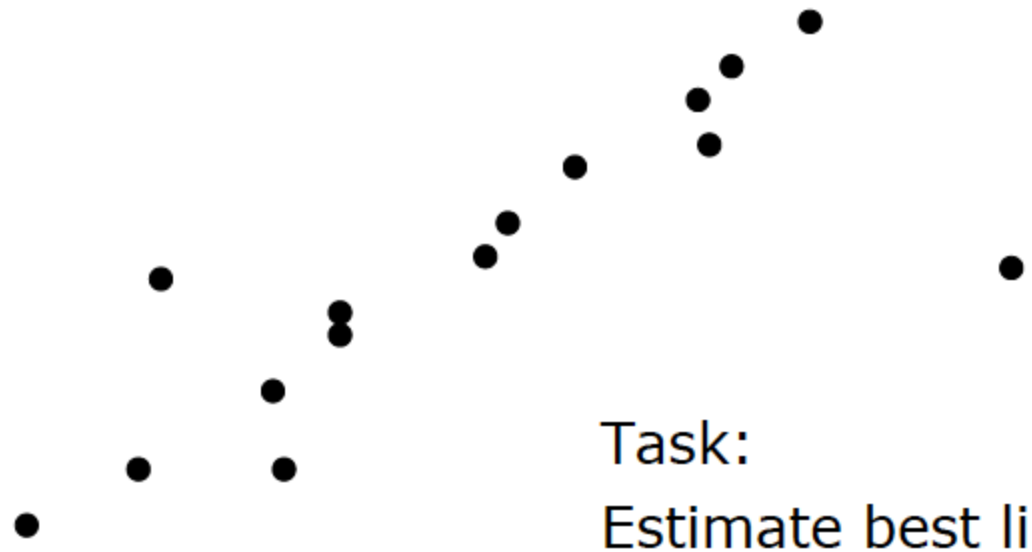
“an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates”



RANdom SAmple Concensus (RANSAC)

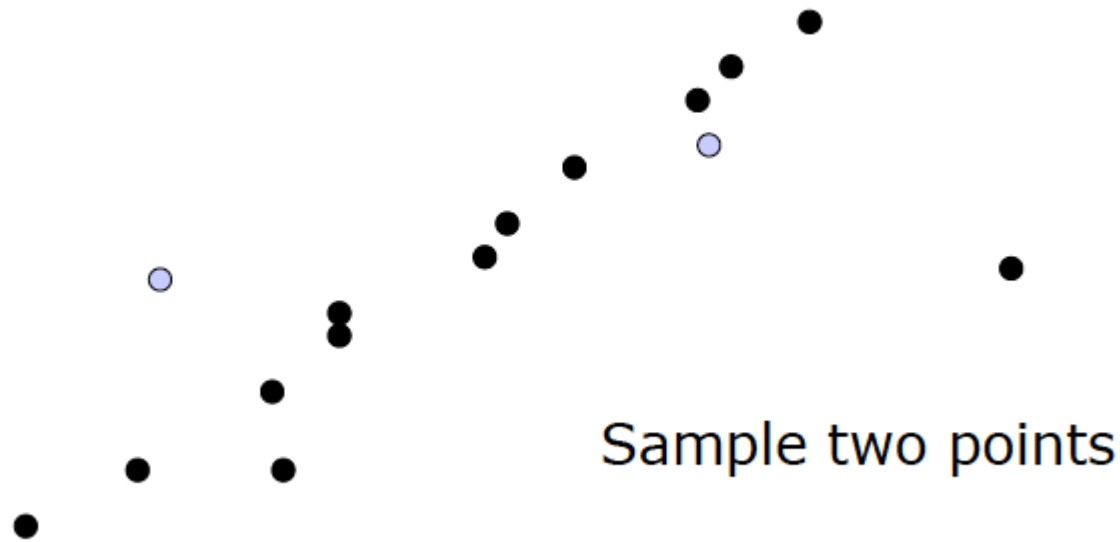
- Randomly choose s samples/points
  - Typically s = minimum sample size that lets you fit a model
- Fit a model (e.g., line) to those samples
- Count the number of inliers that approximately fit the model
- Repeat N times
- Choose the model that has the largest set of inliers

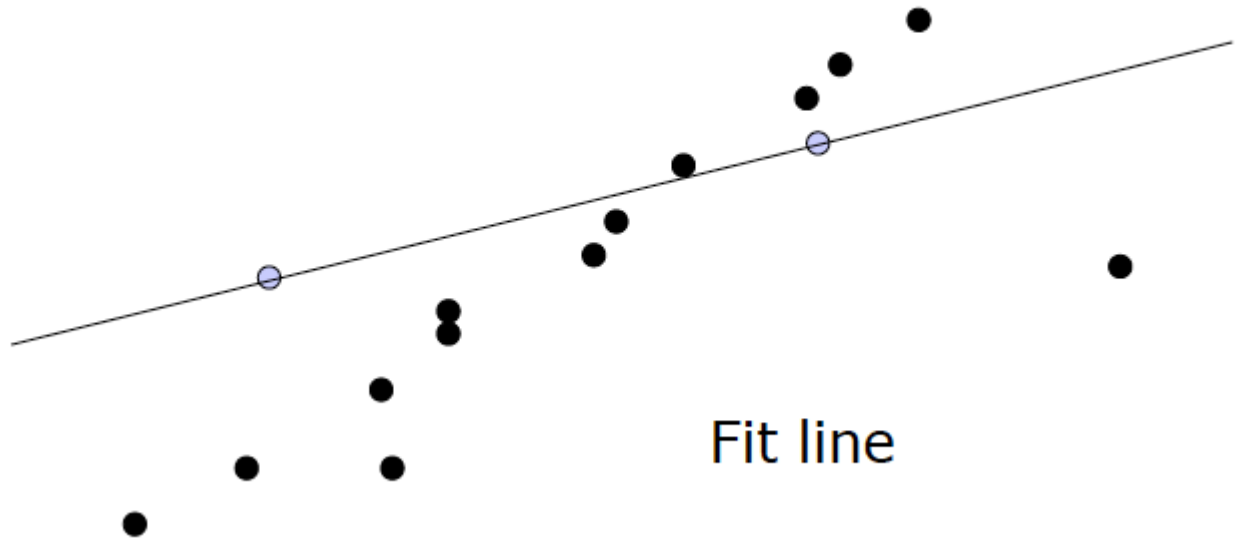
*better results are often achieved by using all the inliers from this step to create an updated model.

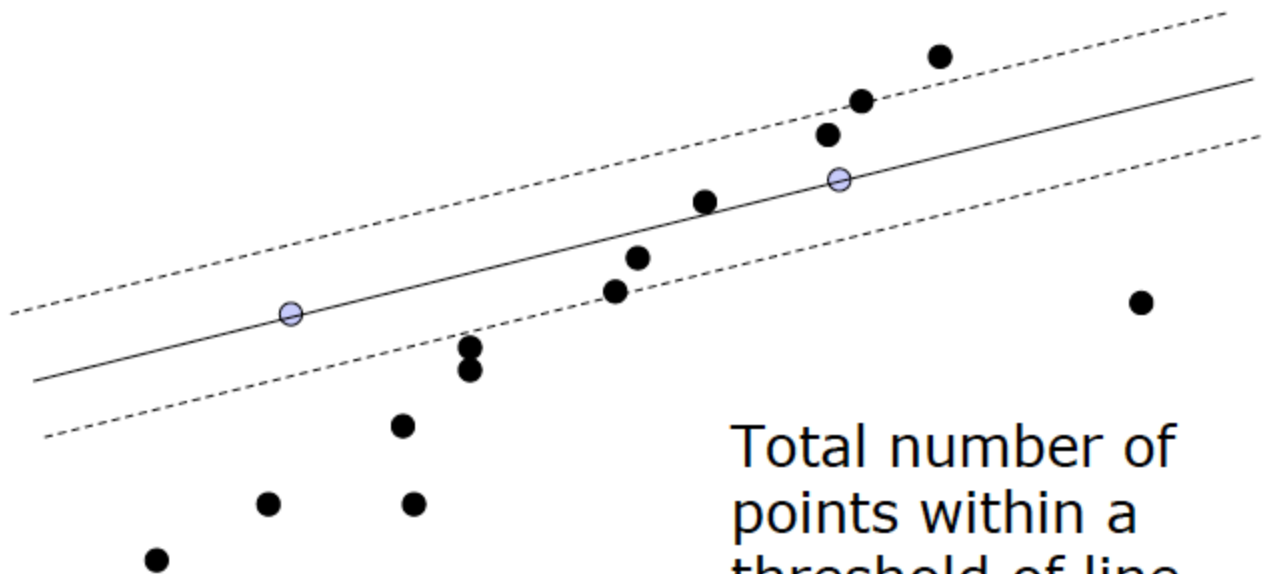


Task:

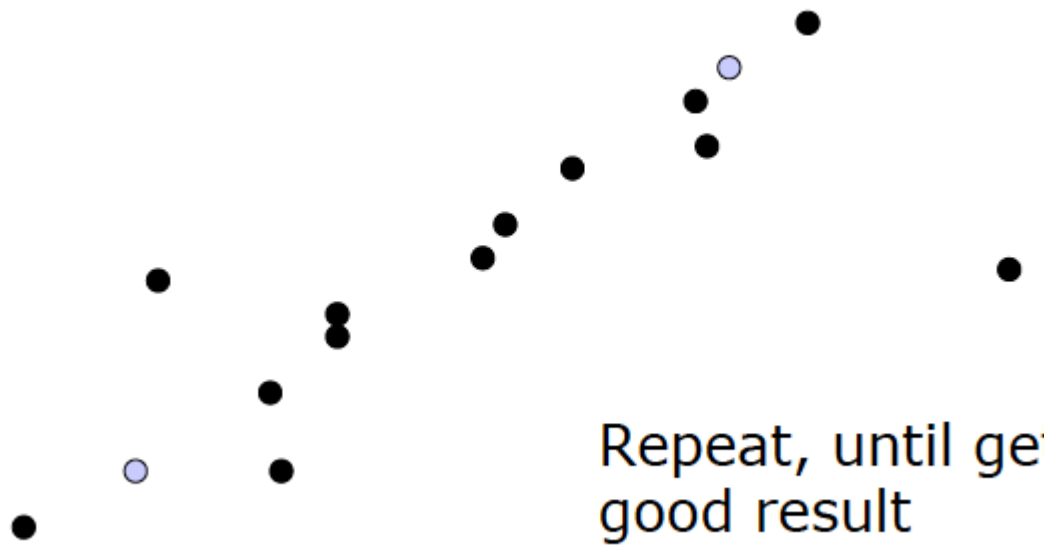
Estimate best line

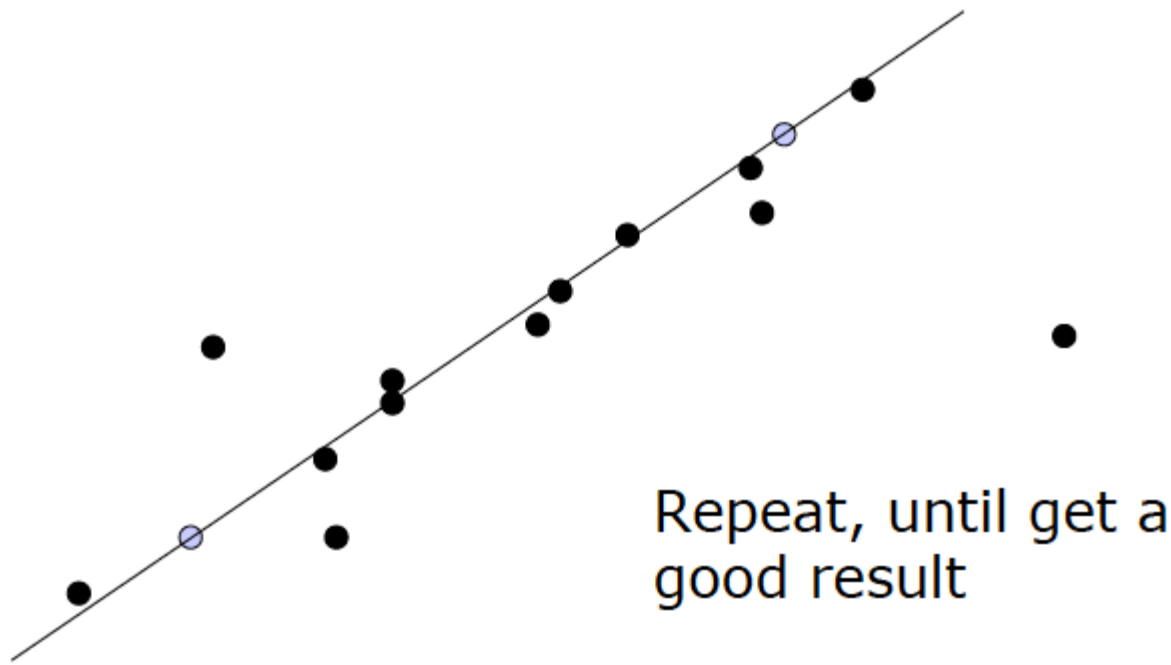

Sample two points


Fit line

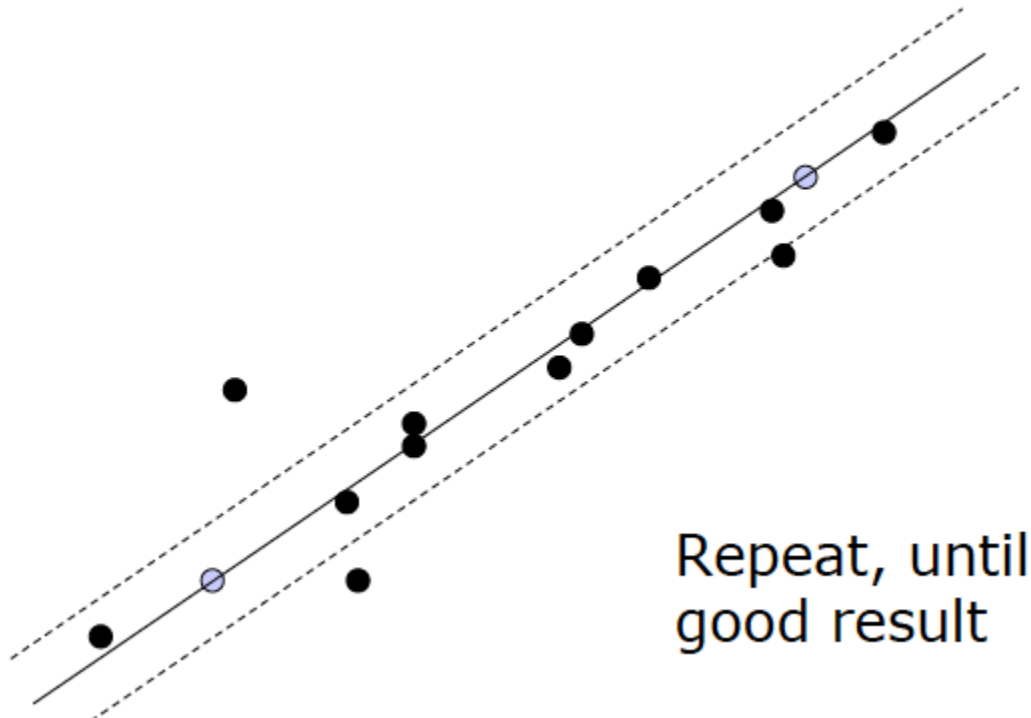


Total number of points within a threshold of line


Repeat, until get a
good result



Repeat, until get a good result



Repeat, until get a good result

RANSAC parameters

- **Inlier threshold** related to the amount of noise we expect in inliers

- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
  - Suppose 20% of the data points are outliers, and we want to fit the correct line with 99% probability
  - How many rounds do we need?

Suppose 20% of the points are outliers, and we want to fit the correct line with 99% probability. How many rounds do we need?

- Let w be the probability of selecting an inlier (.8 in this example)
- We need $n = 2$ points to fit a model (line)
- What is the probability of selecting $n = 2$ inliers to create a good model? $w^2 = .8^2 = 0.64$
- Then the probability of selecting two points that result in a bad model is $1 - w^2 = .36$
- The probability that all of the N tries will be bad is $(1 - w^2)^N$
- If we run N iterations of RANSAC and want the correct answer with some probability p (99% above):

$$(1 - w^n)^N \leq 1 - p$$

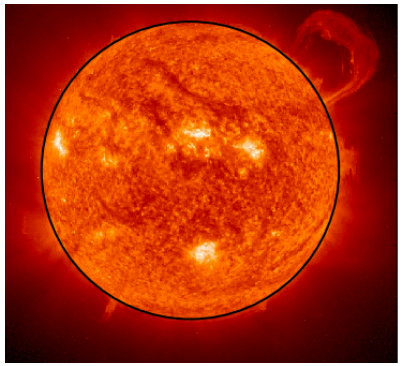
$$N \geq \frac{\log(1 - p)}{\log(1 - w^n)}$$

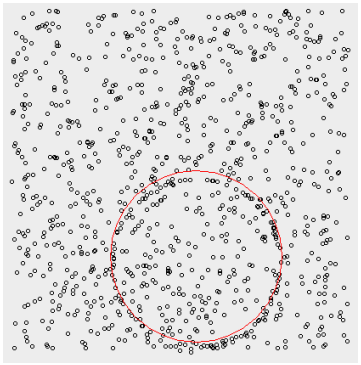
$$N \geq \frac{\log(1 - .99)}{\log(1 - .8^2)} = 4.5$$

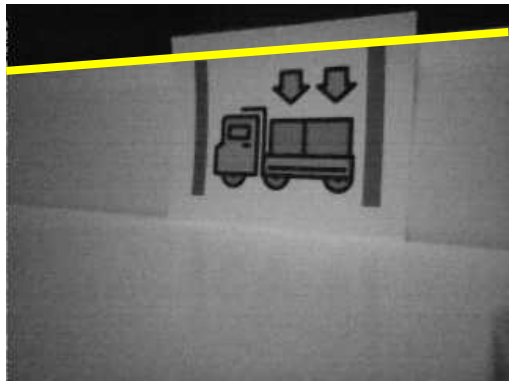
Note: $\log(1 - w^n) < 0$!

RANSAC

RANSAC works extremely well for both low and high-dimensional problems







References

- RANSAC
  - Autonomous Mobile Robots 4.7.2.4
  - Robotics, Vision and Control 14.2.3