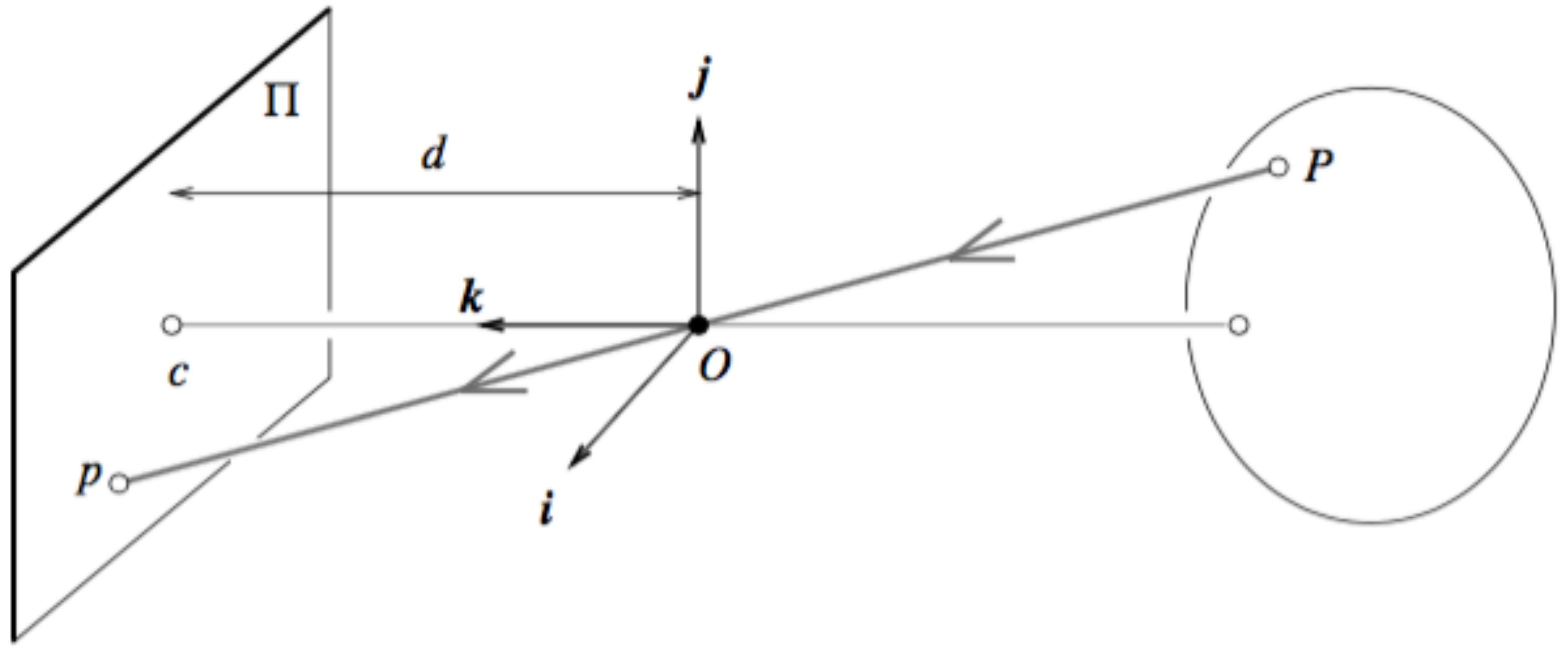
Principles of Robot Autonomy I

Image processing, feature detection, and feature description

From 3D world to 2D images

- So far we have focused on mapping 3D objects onto 2D images and on leveraging such mapping for scene reconstruction
- Next step: how to represent images and infer visual content?



Today’s lecture

- Aim
  - Learn fundamental tools in image processing for filtering and detecting similarities
  - Learn how to detect and describe key features in images

- Readings
  - Siegwart, Nourbakhsh, Scaramuzza. Introduction to Autonomous Mobile Robots. Sections 4.3 – 4.5.4.

How to represent images?

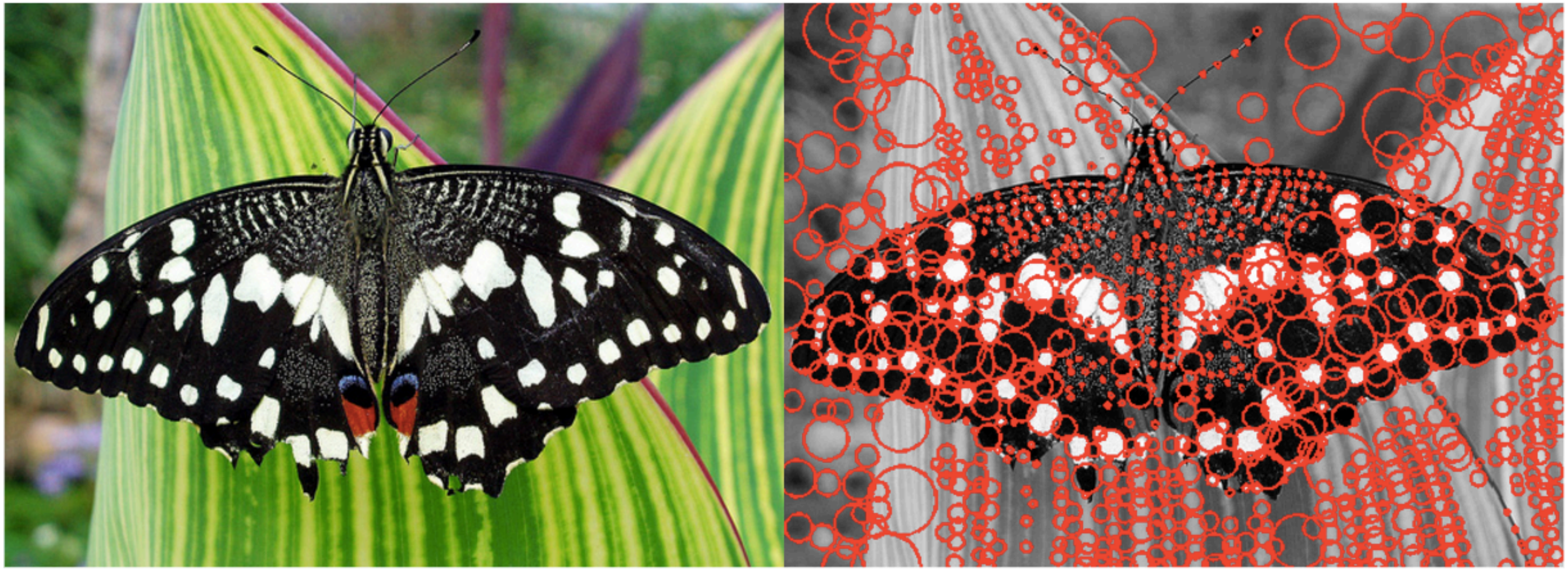
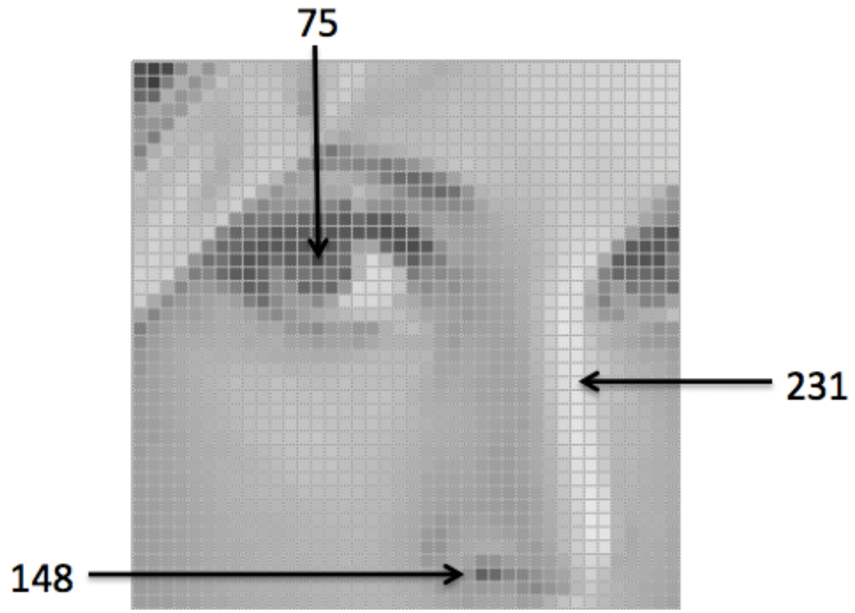


Image processing pipeline

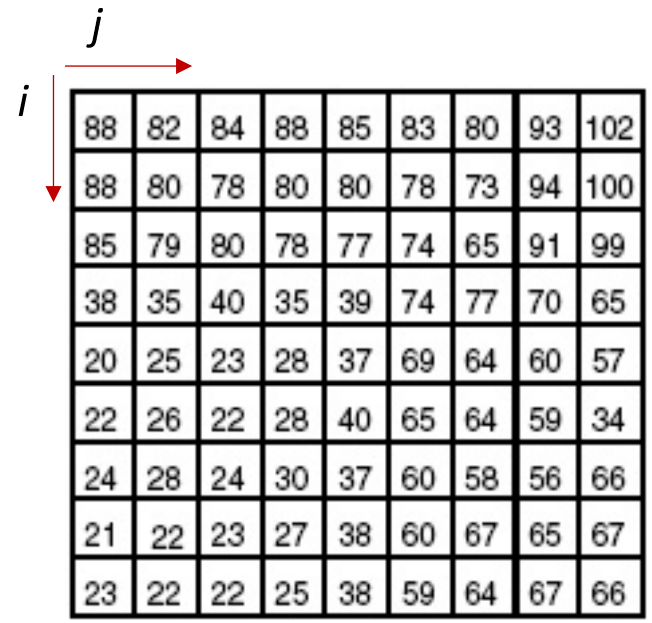
1. Signal treatment / filtering
2. Feature detection (e.g., DoG)
3. Feature description (e.g., SIFT)
4. Higher-level processing

Image filtering

- Filtering: process of accepting / rejecting certain frequency components
- Starting point is to view images as functions $I: [a,b]\times[c,d] \rightarrow [0,L]$, where $I(x,y)$ represents intensity at position (x,y)
- A color image would give rise to a vector function with 3 components



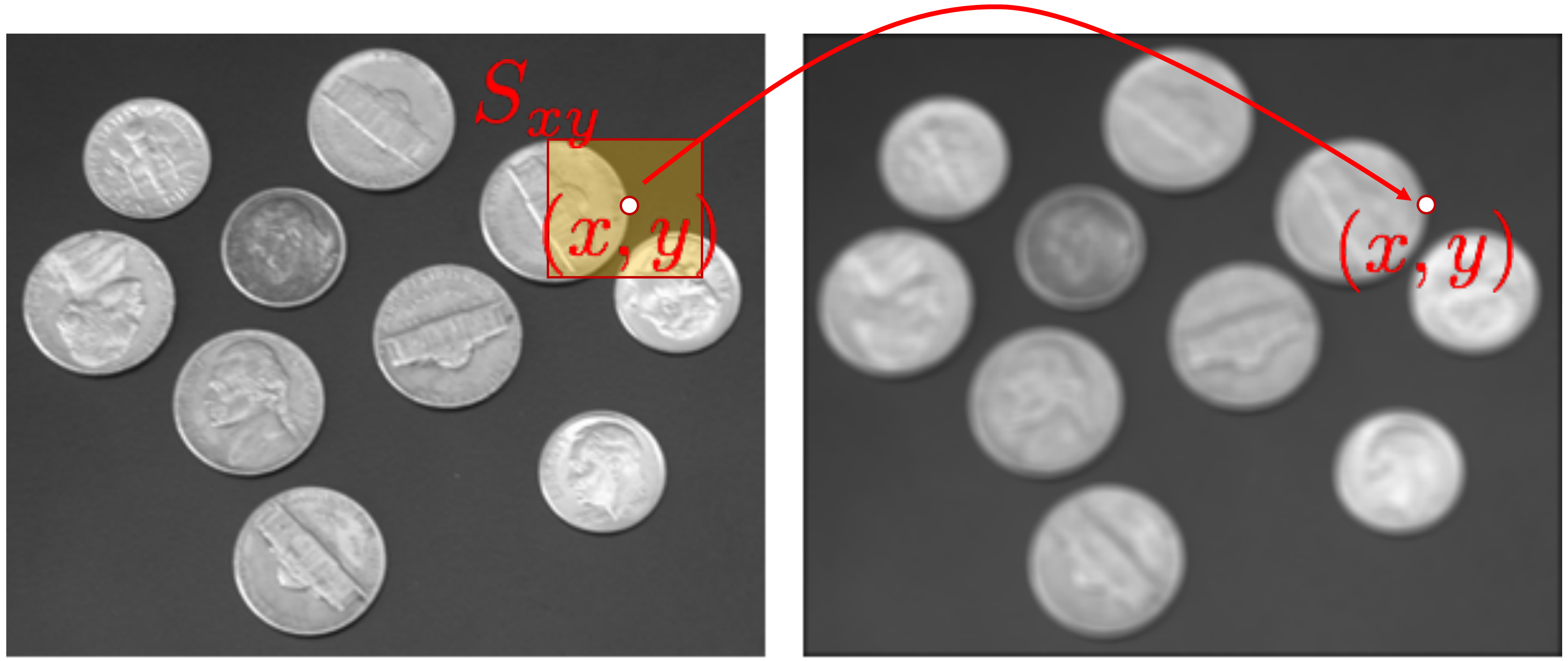
Represented as a matrix



88	82	84	88	85	83	80	93	102
88	80	78	80	80	78	73	94	100
85	79	80	78	77	74	65	91	99
38	35	40	35	39	74	77	70	65
20	25	23	28	37	69	64	60	57
22	26	22	28	40	65	64	59	34
24	28	24	30	37	60	58	56	66
21	22	23	27	38	60	67	65	67
23	22	22	25	38	59	64	67	66

Spatial filters

- A spatial filter consists of
  1. A neighborhood S_{xy} of pixels around the point (x, y) under examination
  2. A predefined operation F that is performed on the image pixels within S_{xy}



Linear spatial filters

- Filters can be linear or non-linear
- We will focus on linear spatial filters

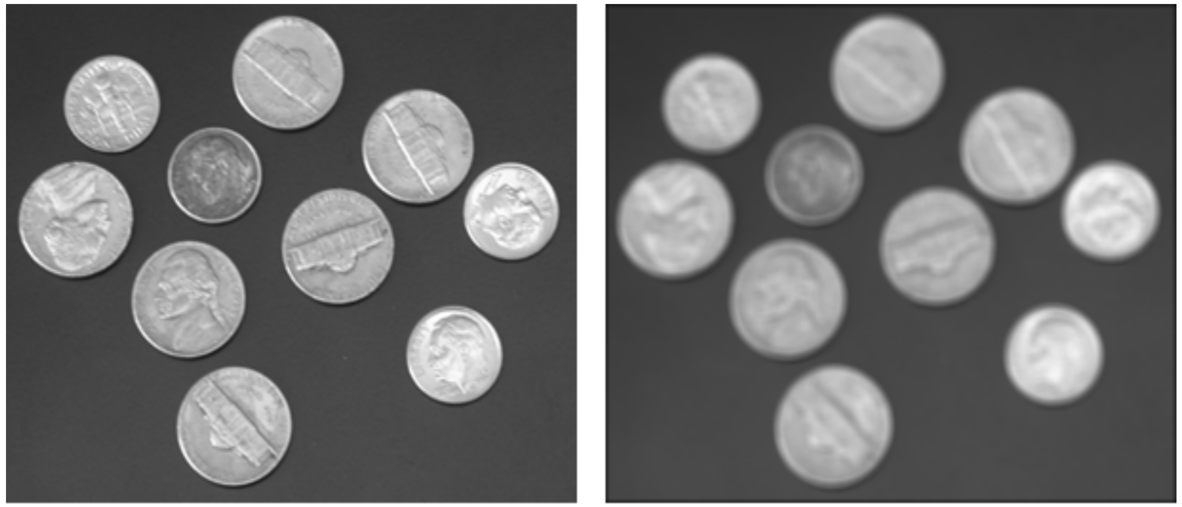
$$\underbrace{I'(x,y)}_{\text{Filtered image}} = F \circ I = \sum_{i=-N}^{N} \sum_{j=-M}^{M} \underbrace{F(i,j)}_{\text{Filter mask}} \underbrace{I(x+i, y+j)}_{\text{Original image}}$$

- Filter *F* (of size $(2N+1)\times(2M+1)$) is usually called a mask, kernel, or window
- Dealing with boundaries: e.g., pad, crop, extend, or wrap

Filter example #1: moving average

- The moving average filter returns the average of the pixels in the mask
- Achieves a smoothing effect (removes sharp features)
- E.g., for a *normalized* 3×3 mask

$$F = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Generated with a 5x5 mask

Filter example #2: Gaussian smoothing

- Gaussian function

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

- To obtain the mask, sample the function about its center
- E.g., for a *normalized* 3×3 mask with $\sigma = 0.85$

$$G = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Convolution

- Still a linear filter, defined as

$$I'(x,y) = F * I = \sum_{i=-N}^{N} \sum_{j=-M}^{M} F(i,j) I(x-i, y-j)$$

- Same as correlation, but with negative signs for the image indices
- Correlation and convolution are identical when the filter is symmetric
- Convolution enjoys the associativity property

$$F * (G * I) = (F * G) * I$$

- Example: smooth image & take derivative = convolve derivative filter with Gaussian filter & convolve the resulting filter with the image

Separability of masks

- A mask is separable if it can be broken down into the convolution of two kernels

$$F = F_1 * F_2$$

- If a mask is separable into “smaller” masks, then it is often cheaper to apply F_1 followed by F_2, rather than F directly
- Special case: mask representable as outer product of two vectors (equivalent to two-dimensional convolution of those two vectors)
- If masks is $M \times M$, and image has size $w \times h$, then complexity is
  - $O(M^2 wh)$ with no separability
  - $O(2Mwh)$ with separability into outer product of two vectors

Example of separable masks

- Moving average

$$F = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1\\ 1\\ 1\end{bmatrix}\begin{bmatrix}1 & 1 & 1\end{bmatrix}$$

- Gaussian smoothing

$$\begin{aligned} G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{x^2}{2\sigma^2}\right)\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{y^2}{2\sigma^2}\right) \\ &= g_\sigma(x)\cdot g_\sigma(y) \end{aligned}$$

Differentiation

- Derivative of discrete function (one possible way)

$$\frac{\partial I}{\partial x} = I(x+1,y) - I(x-1,y)$$
$$\frac{\partial I}{\partial y} = I(x,y+1) - I(x,y-1)$$

- Derivative as a convolution operation; e.g., Sobel masks:

Along *x* direction

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Along *y* direction

$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Note: masks are mirrored
In convolution

Similarity measures

- Filtering can also be used to determine similarity across images (e.g., to detect correspondences)

$$SAD = \sum_{i=-n}^{n} \sum_{j=-m}^{m} |I_1(x+i, y+j) - I_2(x'+i, y'+j)|$$

Sum of absolute differences

$$SSD = \sum_{i=-n}^{n} \sum_{j=-m}^{m} [I_1(x+i, y+j) - I_2(x'+i, y'+j)]^2$$

Sum of squared differences

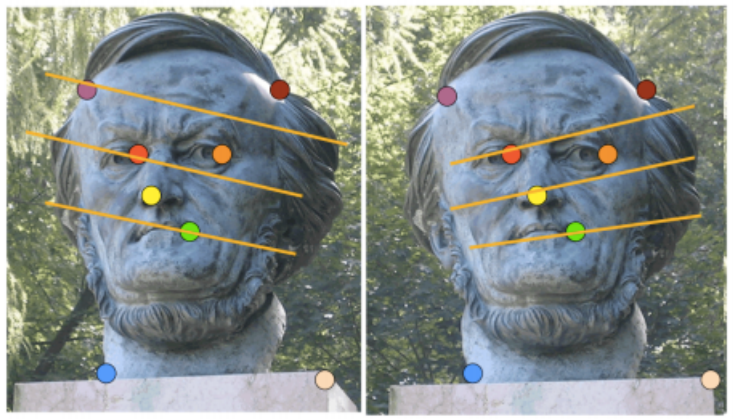
Detectors

- Goal: detect local features, i.e., image patterns that differ from immediate neighborhood in terms of intensity, color, or texture

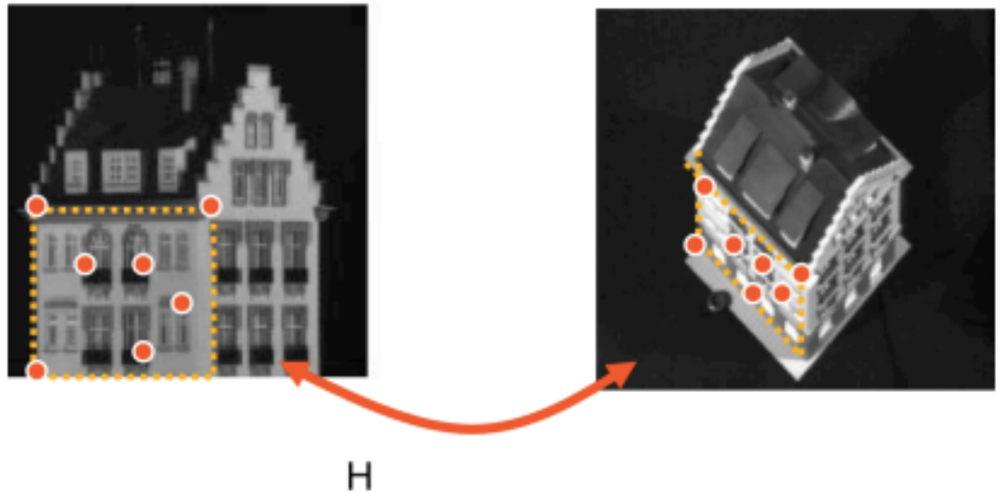
- We will focus on
  - Edge detectors
  - Corner detectors

Use of detectors/descriptors: examples

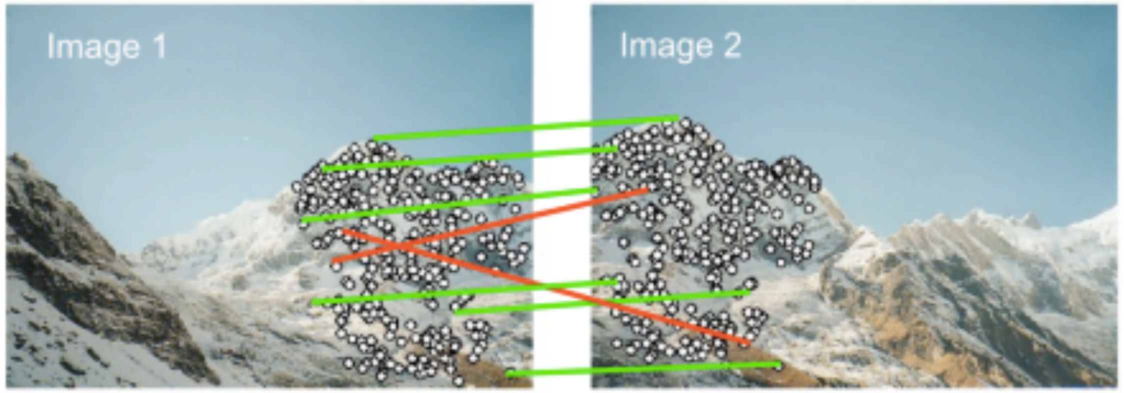
Stereo reconstruction



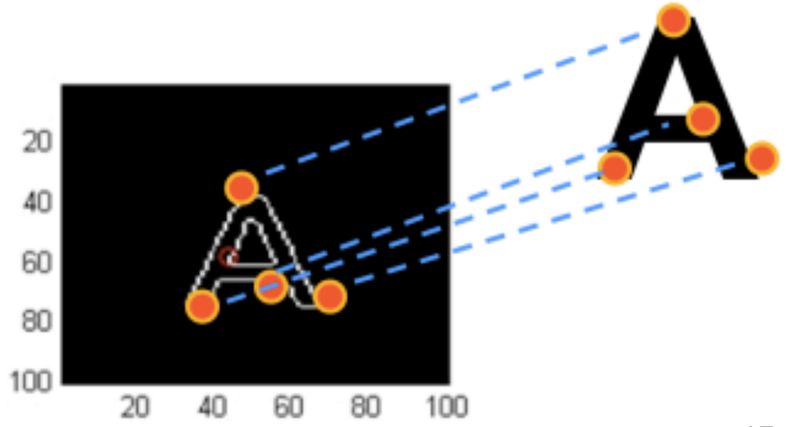
Estimating homographic transformations



Panorama stitching



Object detection

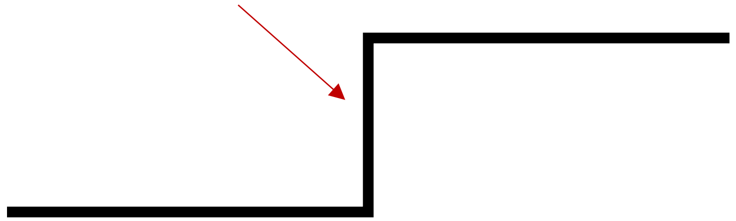


Edge detectors

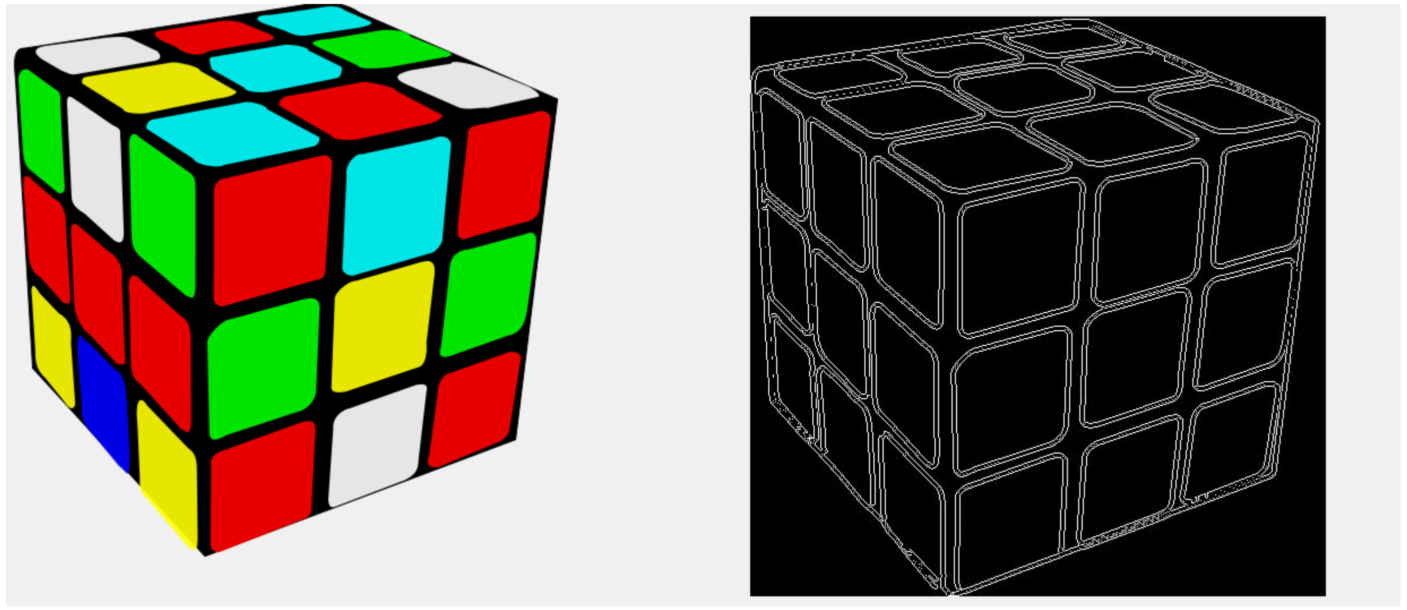
- Edge: region in an image where there is a *significant* change in intensity values along one direction, and *negligible* change along the orthogonal direction

In 1D

Magnitude of 1st order derivative is large, 2nd order derivative is equal to zero



In 2D



Criteria for “good” edge detection

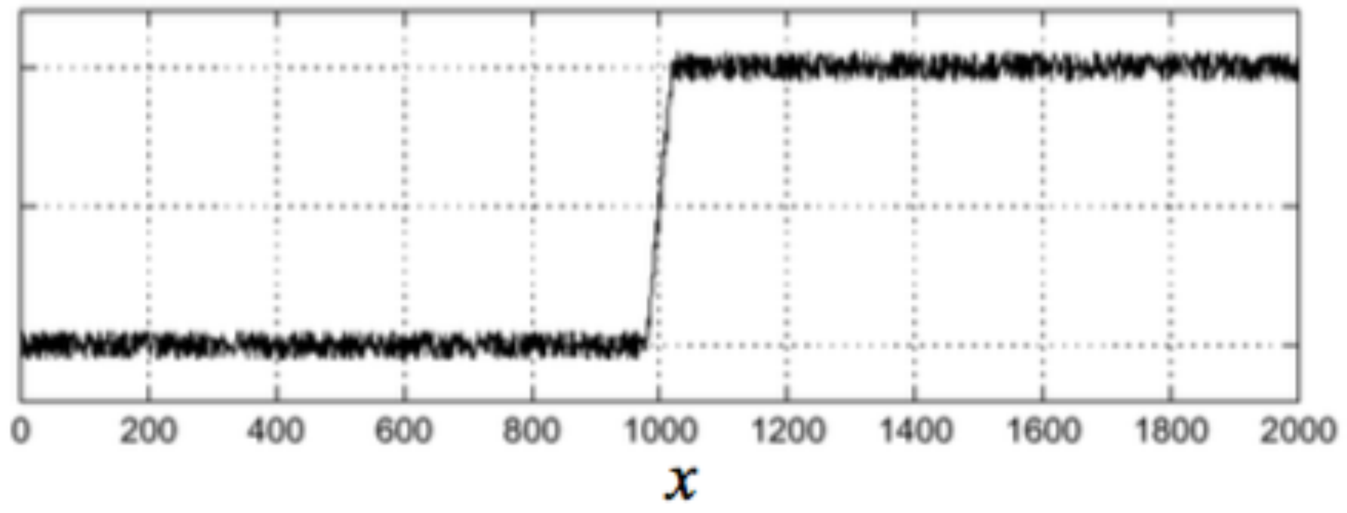
- Accuracy: minimize false positives and negatives
- Localization: edges must be detected as close as possible to the true edges
- Single response: detect one edge per real edge in the image

Strategy to design an edge detector

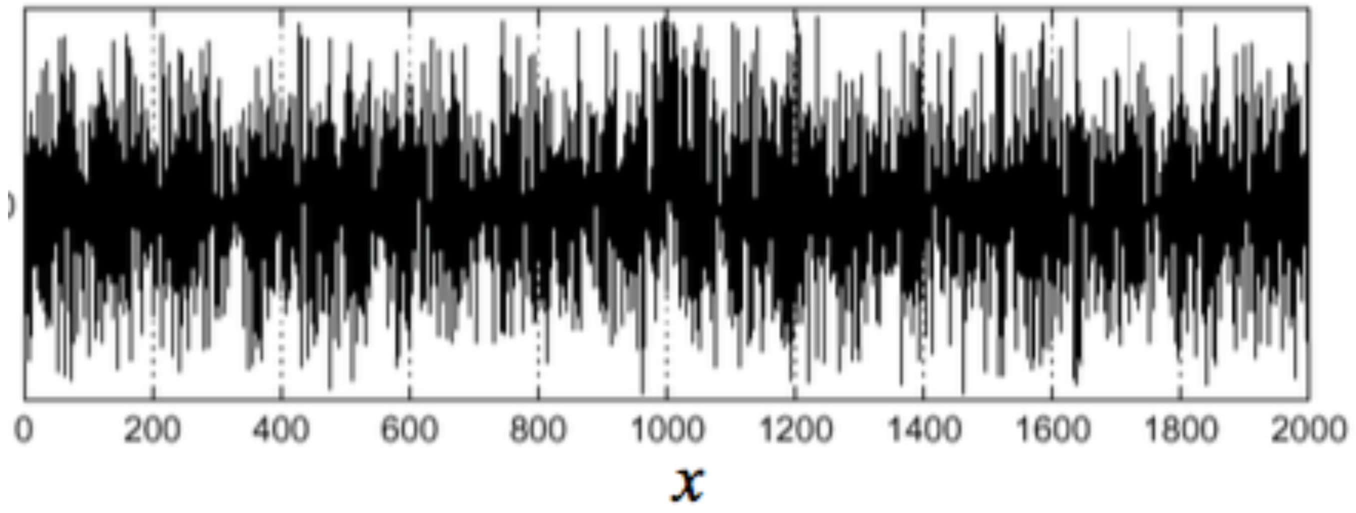
- Two steps:
  1. Differentiation: take derivatives along *x* and *y* directions to find locations with high gradients
  2. Smoothing: smooth the image to reduce noise prior to differentiation process

1D case: differentiation without smoothing

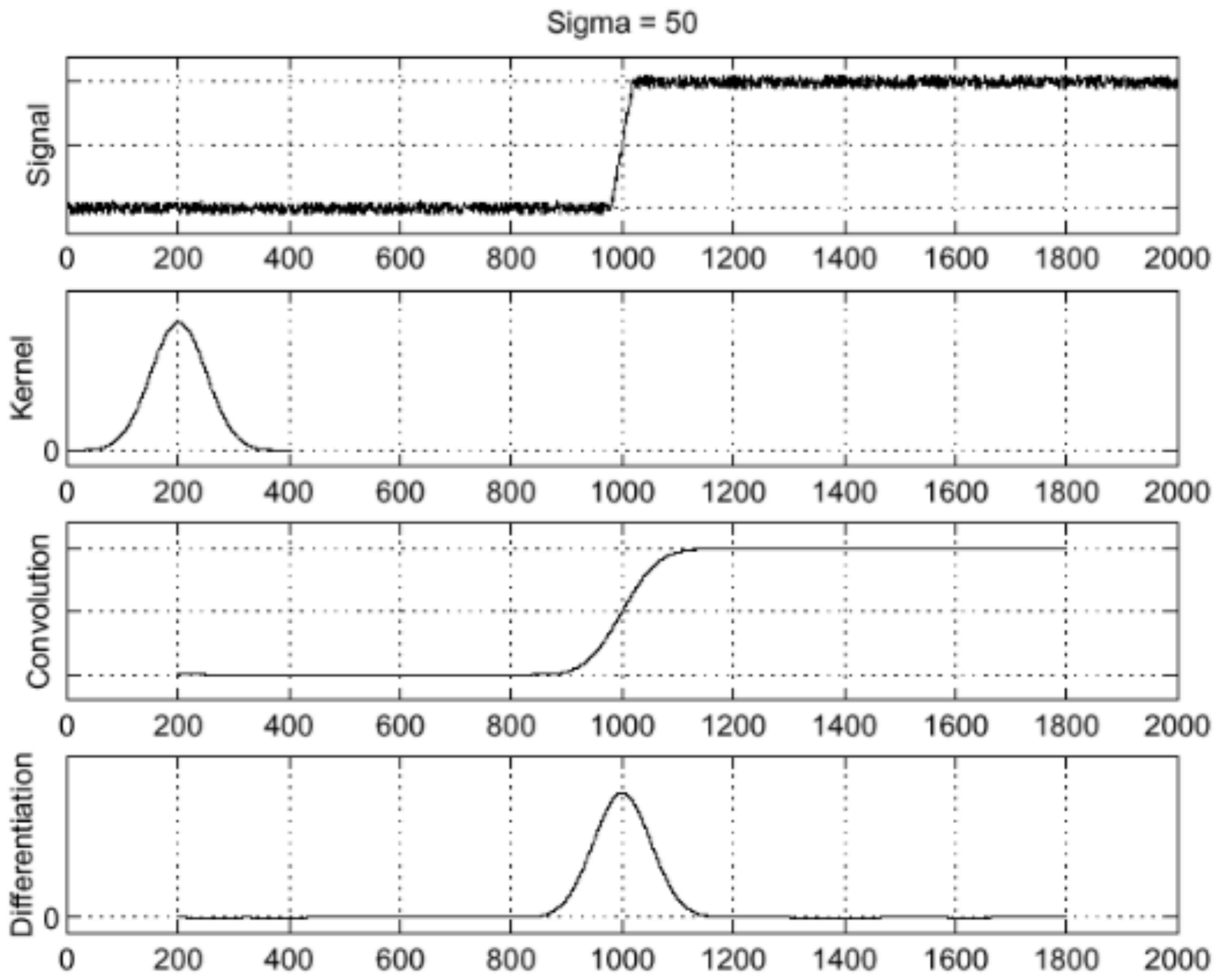
$I(x)$



$$\frac{dI(x)}{dx}$$



1D case: differentiation with smoothing



$$I(x)$$

$$g_\sigma(x)$$

$$s(x) = g_\sigma(x) * I(x)$$

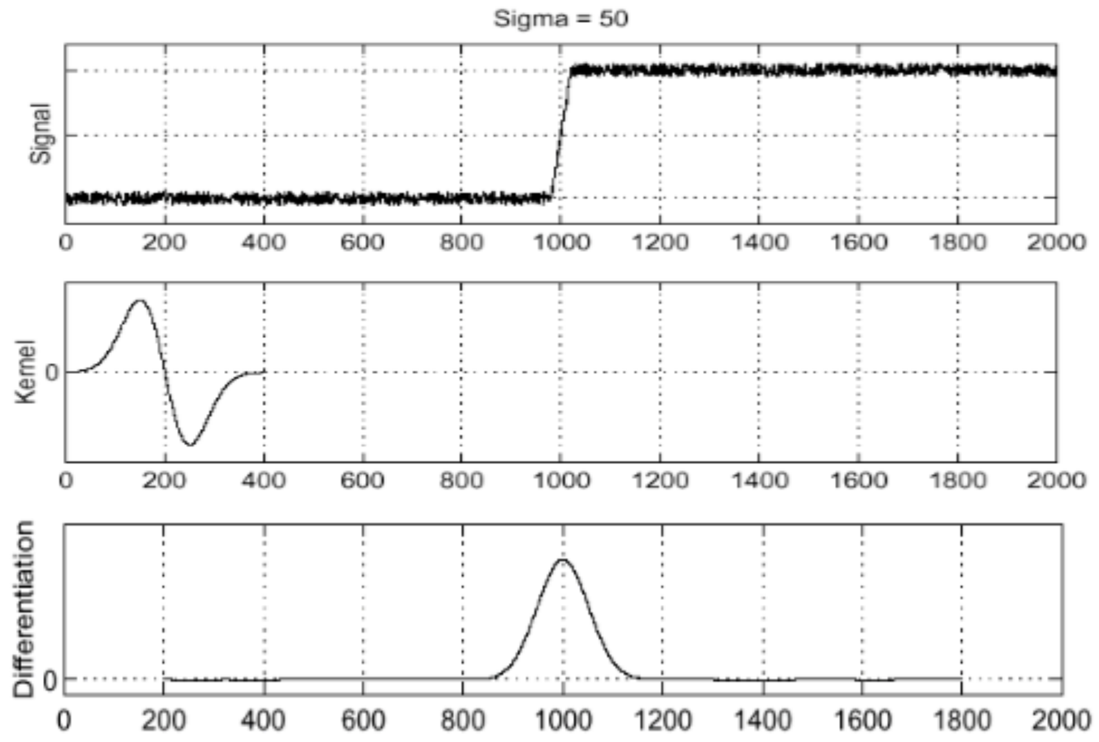
$$s'(x) = \frac{d}{dx} * s(x)$$

Edges occur at maxima or minima of $s'(x)$

A better implementation

- Convolution theorem:

$$s'(x) = \frac{d}{dx} * (g_\sigma(x) * I(x)) = \underbrace{\left(\frac{d}{dx} * g_\sigma(x)\right)}_{g'_\sigma(x)} * I(x)$$



$I(x)$

$g'_\sigma(x)$

$$s'(x) = g'_\sigma(x) * I(x)$$

Edge detection in 2D

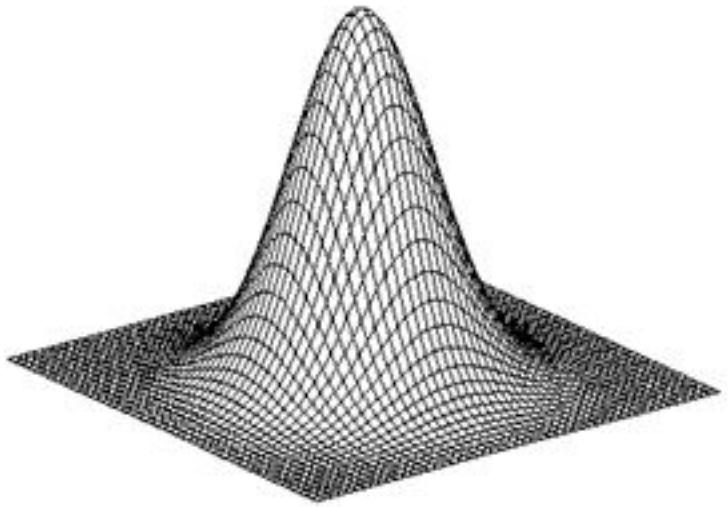
1. Find the gradient of smoothed image in both directions

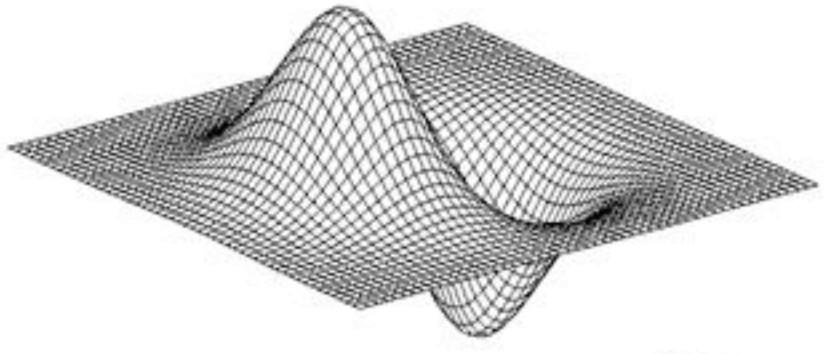
$$\nabla S := \begin{bmatrix} \frac{\partial}{\partial x} * (G_\sigma * I) \\ \frac{\partial}{\partial y} * (G_\sigma * I) \end{bmatrix} = \begin{bmatrix} \left(\frac{\partial}{\partial x} * G_\sigma\right) * I \\ \left(\frac{\partial}{\partial y} * G_\sigma\right) * I \end{bmatrix} = \begin{bmatrix} G_{\sigma,x} * I \\ G_{\sigma,y} * I \end{bmatrix} := \begin{bmatrix} S_x \\ S_y \end{bmatrix}$$

E.g., Sobel

2. Compute the magnitude $|\nabla S| = \sqrt{S_x^2 + S_y^2}$ and discard pixels below a certain threshold
3. Non-maxima suppression: identify local maxima of $|\nabla S|$

Derivative of Gaussian filter

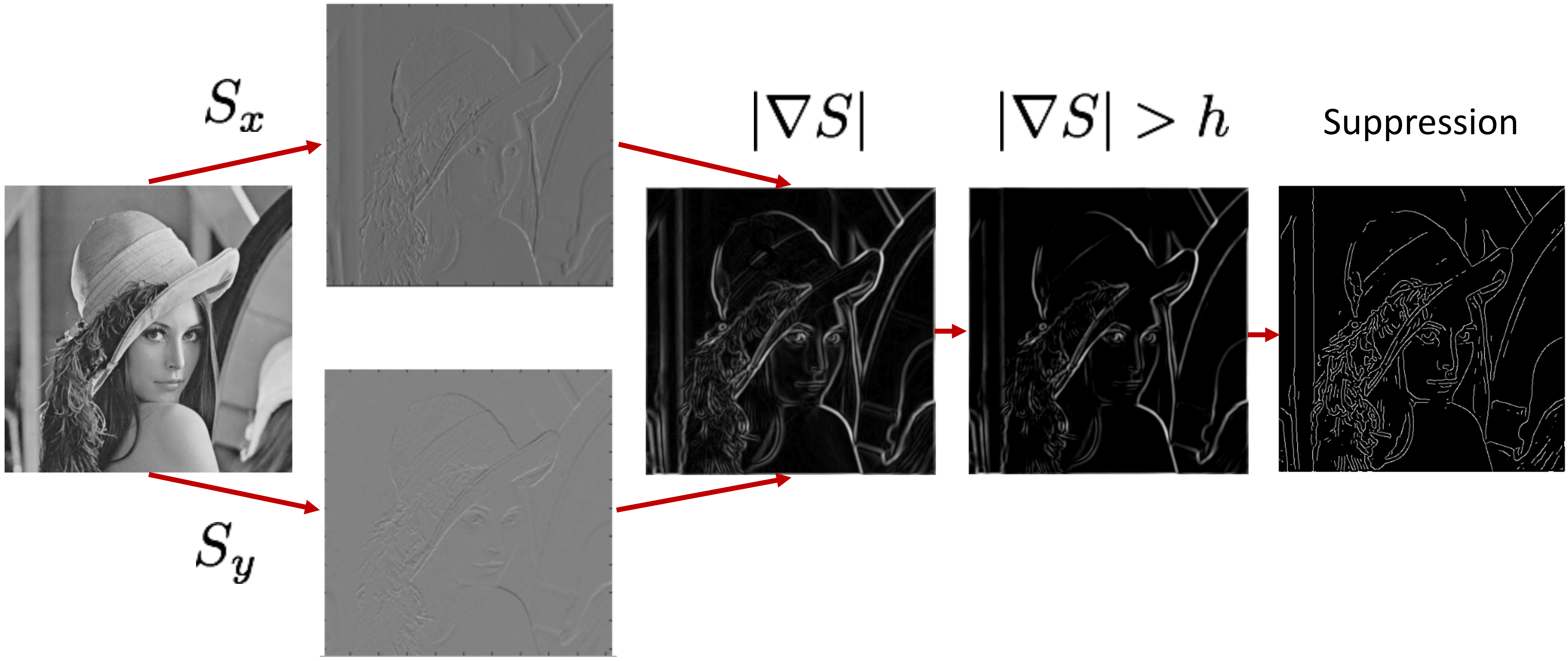




$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$\frac{\partial G_\sigma(x,y)}{\partial x}$$

Canny edge detector



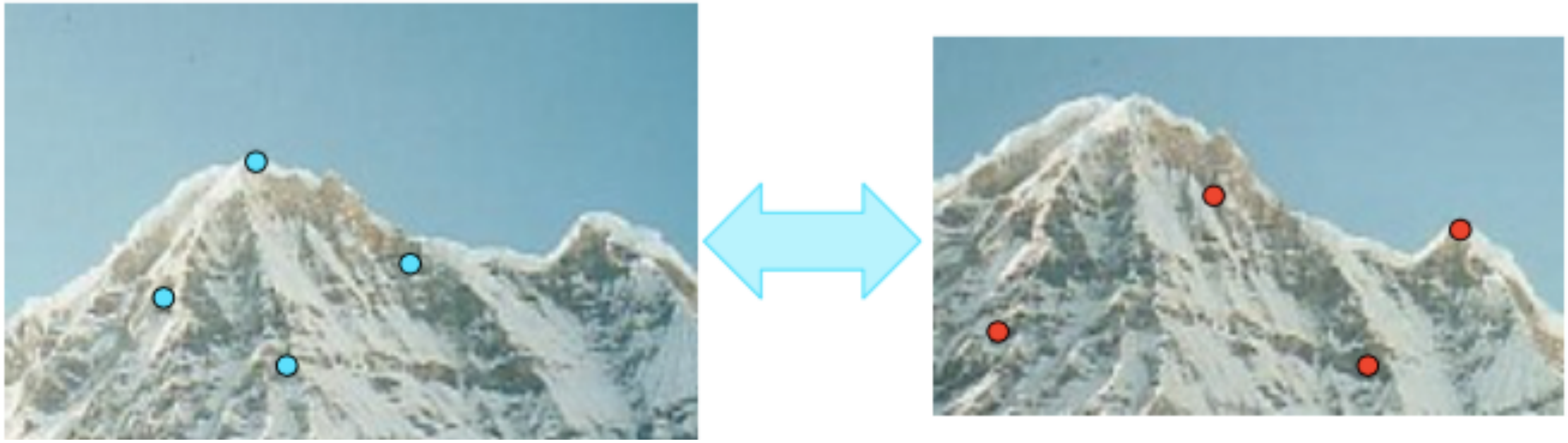
Corner detectors

Key Criteria for “good” corner detectors

1. Repeatability: same feature can be found in multiple images despite geometric and photometric transformations

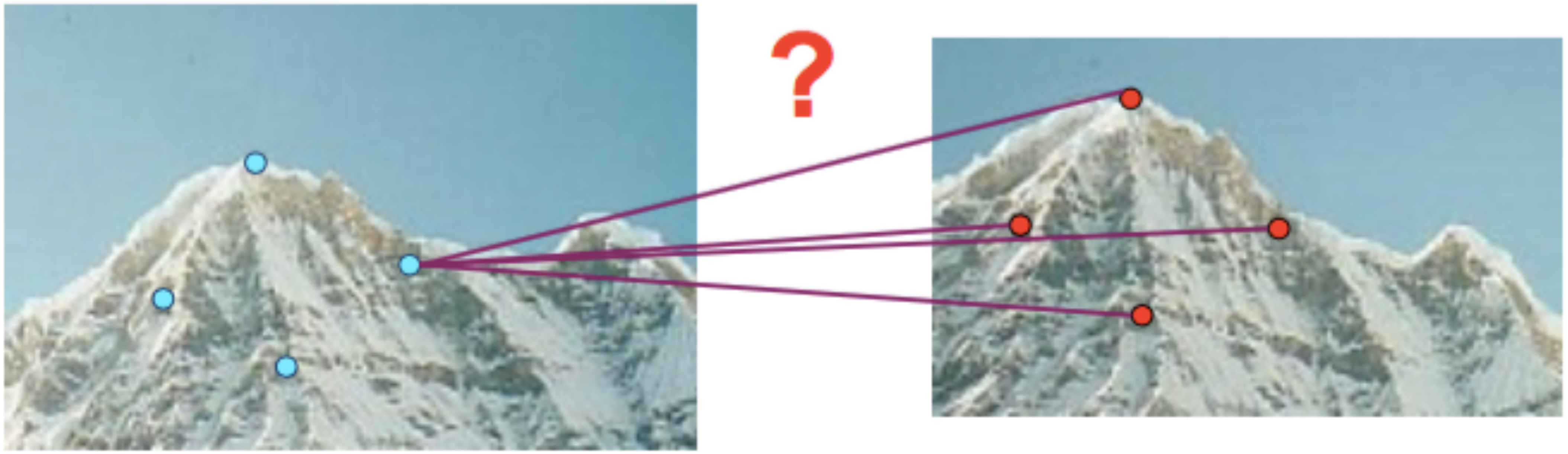
2. Distinctiveness: information carried by the patch surrounding the feature should be as distinctive as possible

Repeatability



Without repeatability, matching is impossible

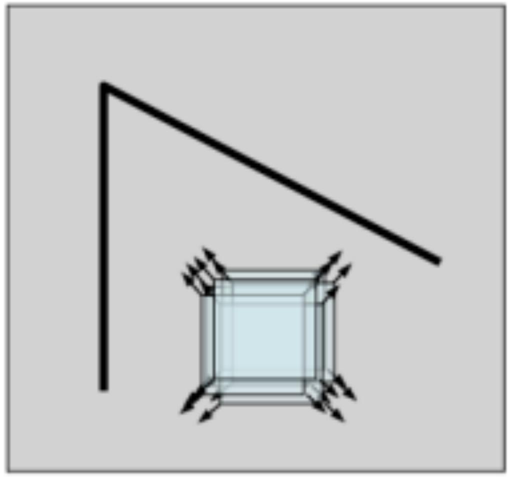
Distinctiveness



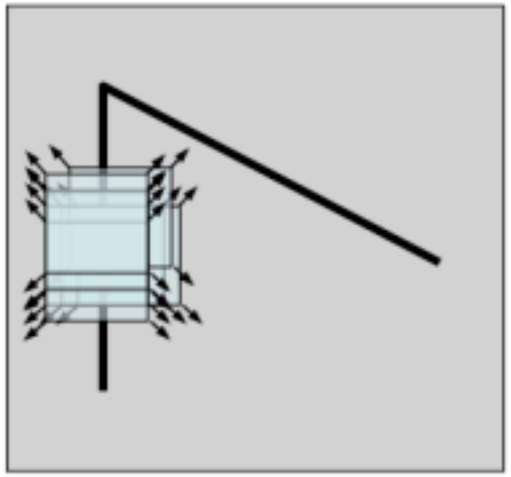
Without distinctiveness, it is not possible to establish reliable correspondences; distinctiveness is key for having a useful descriptor

Finding corners

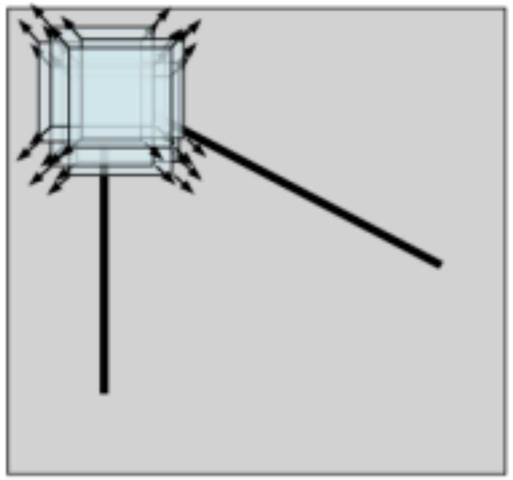
- Corner: intersection of two or more edges
- Geometric intuition for corner detection: explore how intensity changes as we shift a window



Flat: no changes in any direction

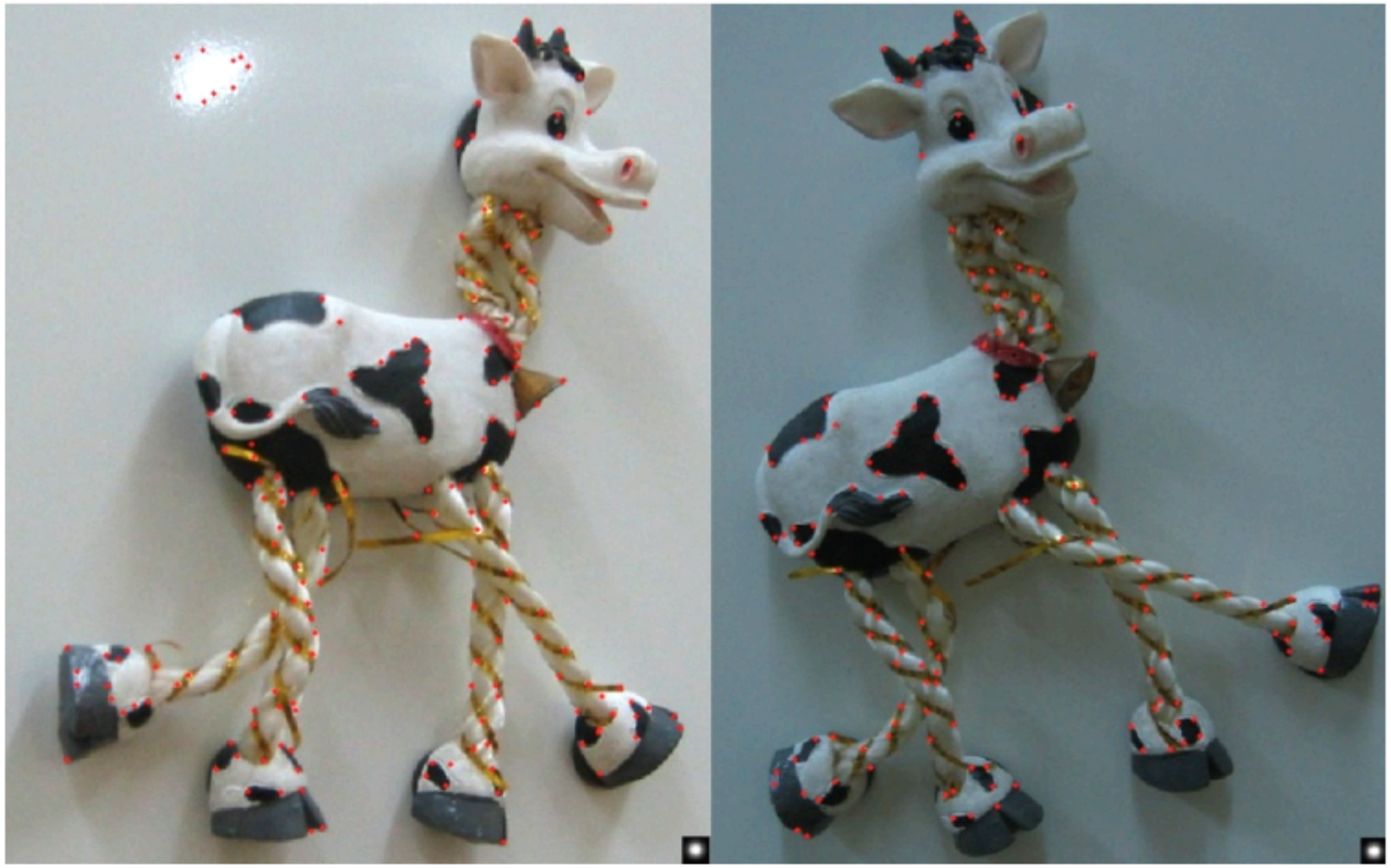


Edge: no change along the edge direction



Corner: changes in all directions

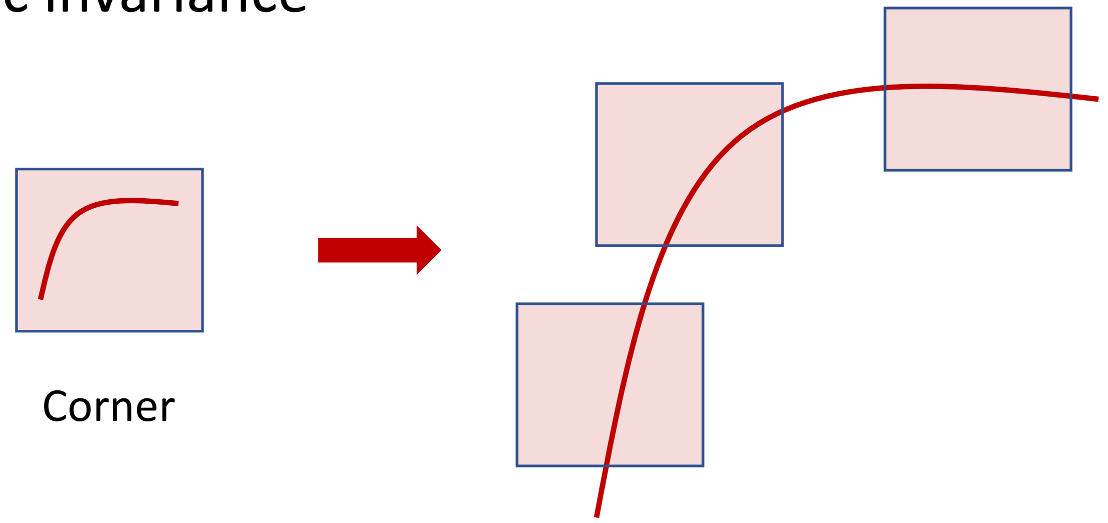
Harris detector: example



Properties of Harris detectors

- Widely used
- Detection is invariant to
  - Rotation -> geometric invariance
  - Linear intensity changes -> photometric invariance

- Detection is not invariant to
  - Scale changes
  - Geometric affine changes



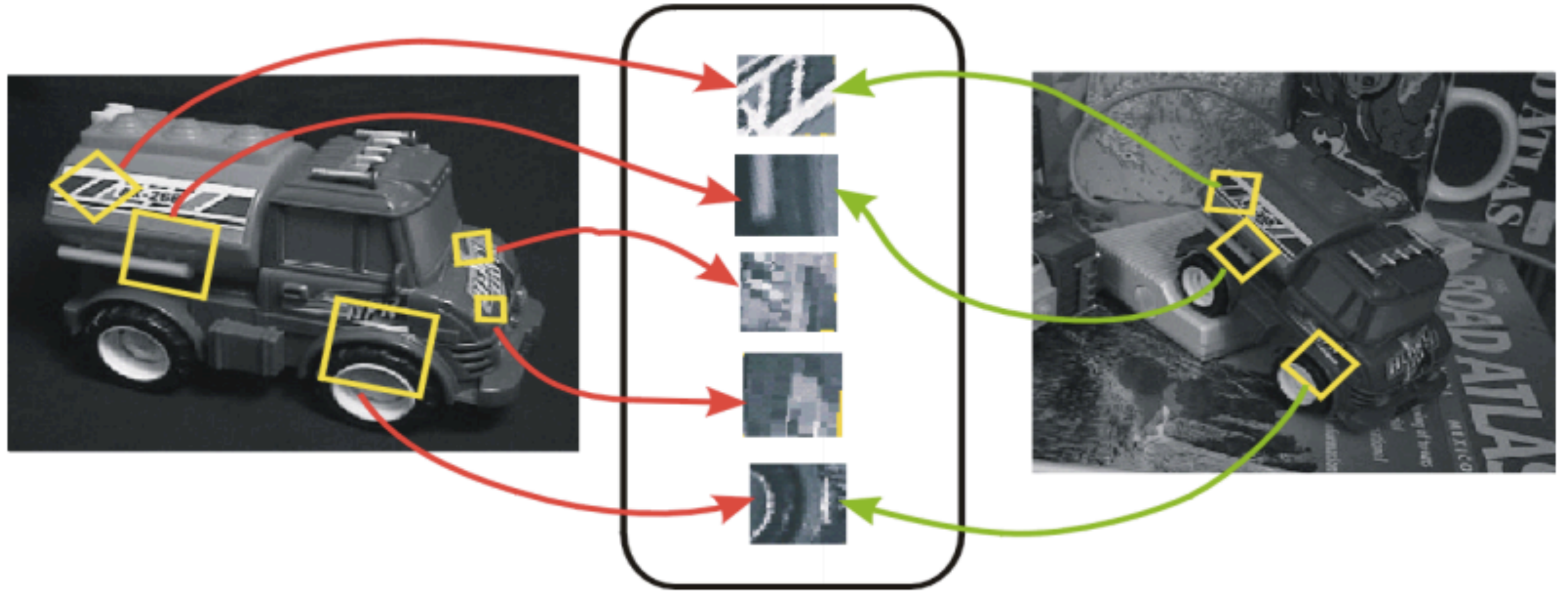
All points classified as edges!

➡ Scale-invariant detection, such as

1. Harris-Laplacian
2. SIFT

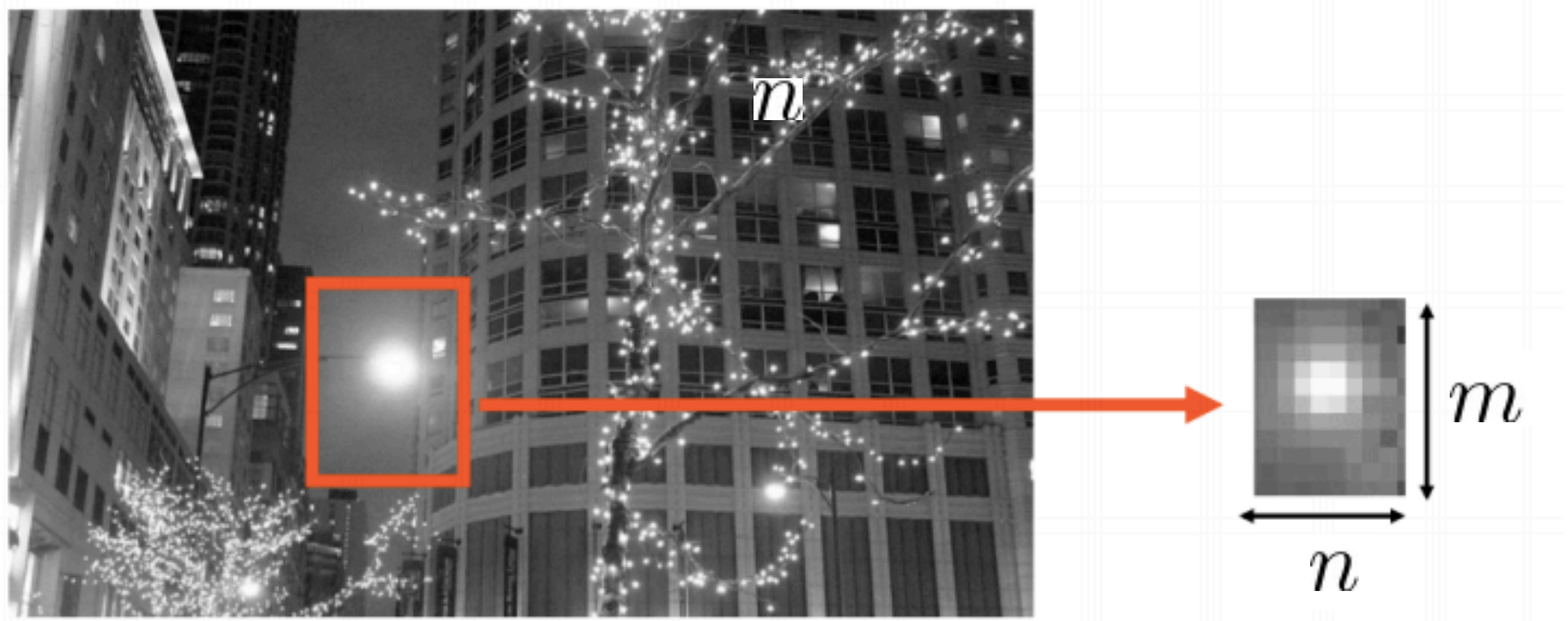
Descriptors

- Goal: *describe* keypoints so that we can compare them across images or use them for object detection or matching
- Desired properties:
    - Invariance with respect to pose, scale, illumination, etc.
    - Distinctiviness



Simplest descriptor

- Naïve descriptor: associate with a given keypoint an $n \times m$ window of pixel intensities centered at that keypoint
- Window can be normalized to make it invariant to illumination



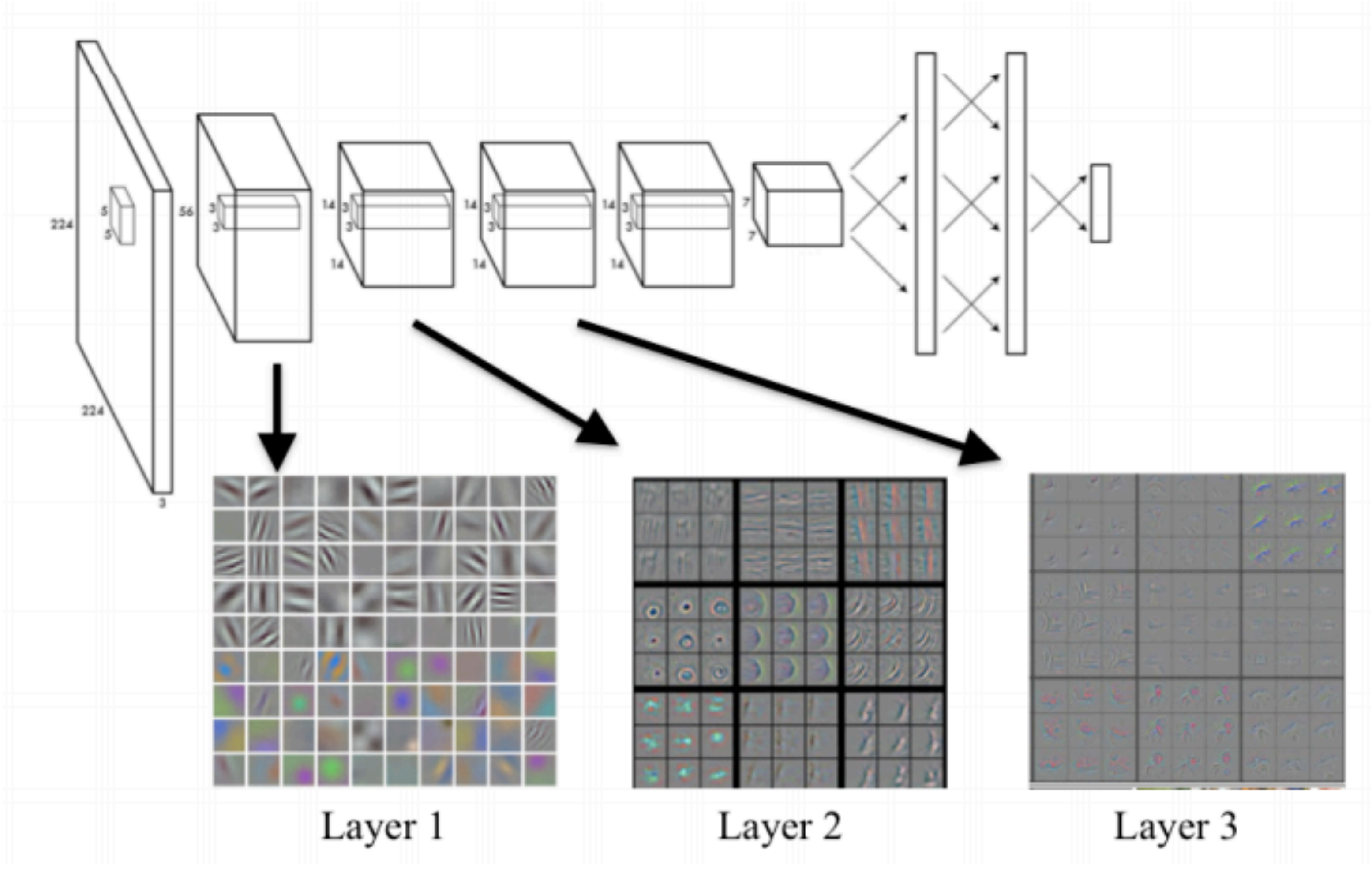
Main drawbacks

1. Sensitive to pose
2. Sensitive to scale
3. Poorly distinctive

Popular detectors / descriptors

- SIFT
  - Invariant to rotation and scale, but computationally demanding
  - SIFT descriptor is a 128-dimensional vector!
- SURF
- FAST
- BRIEF
- ORB
- BRISK

A different paradigm: using CNNs to detect and describe features



Next time