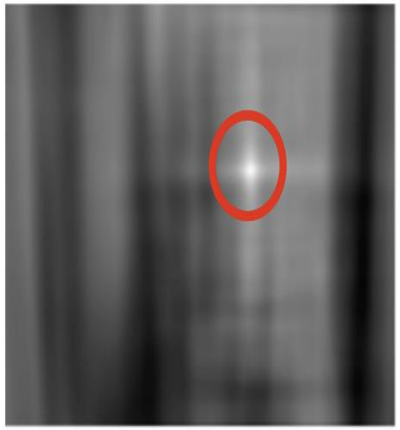
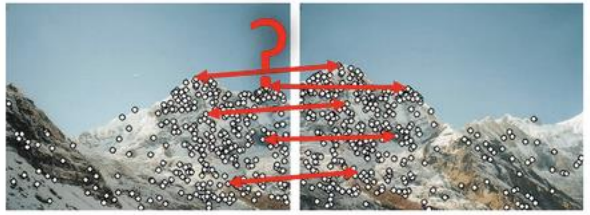
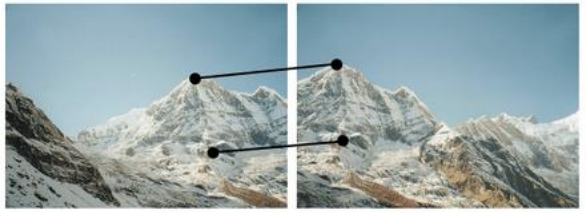
Announcement

2/11/2025 Tuesday

- Final project
  - Details out
  - Keep milestones in mind

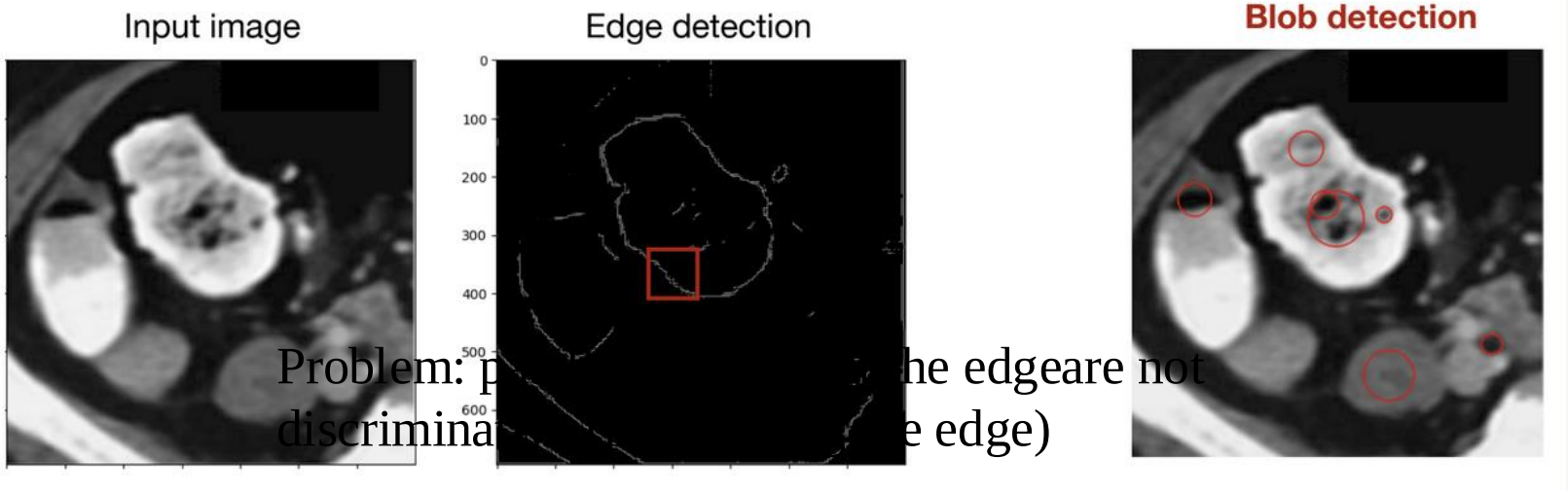
- Image transformation

Method 2: feature-based approach

	1. What to match	2. How to match (NCC)	3. Match → transformation
Intensity-based			Translation $T: \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x - x_2 + x_1 \\ y - y_2 + y_1 \end{bmatrix}$
Feature-based			
	Find key points	Many matches	Find inliers and $\{(x, y), (x', y')\}_K \rightarrow T$

What are good regions to match?

- Edge detection
  - Detects high-gradient boundaries
    - may change significantly under transformations
- Blob detection
  - Blobs: Homogeneous regions that stand out from the background
    - Shape; Size; Brightness
  - Detects keypoints that are scale- and rotation-invariant
    - More robust/reliable



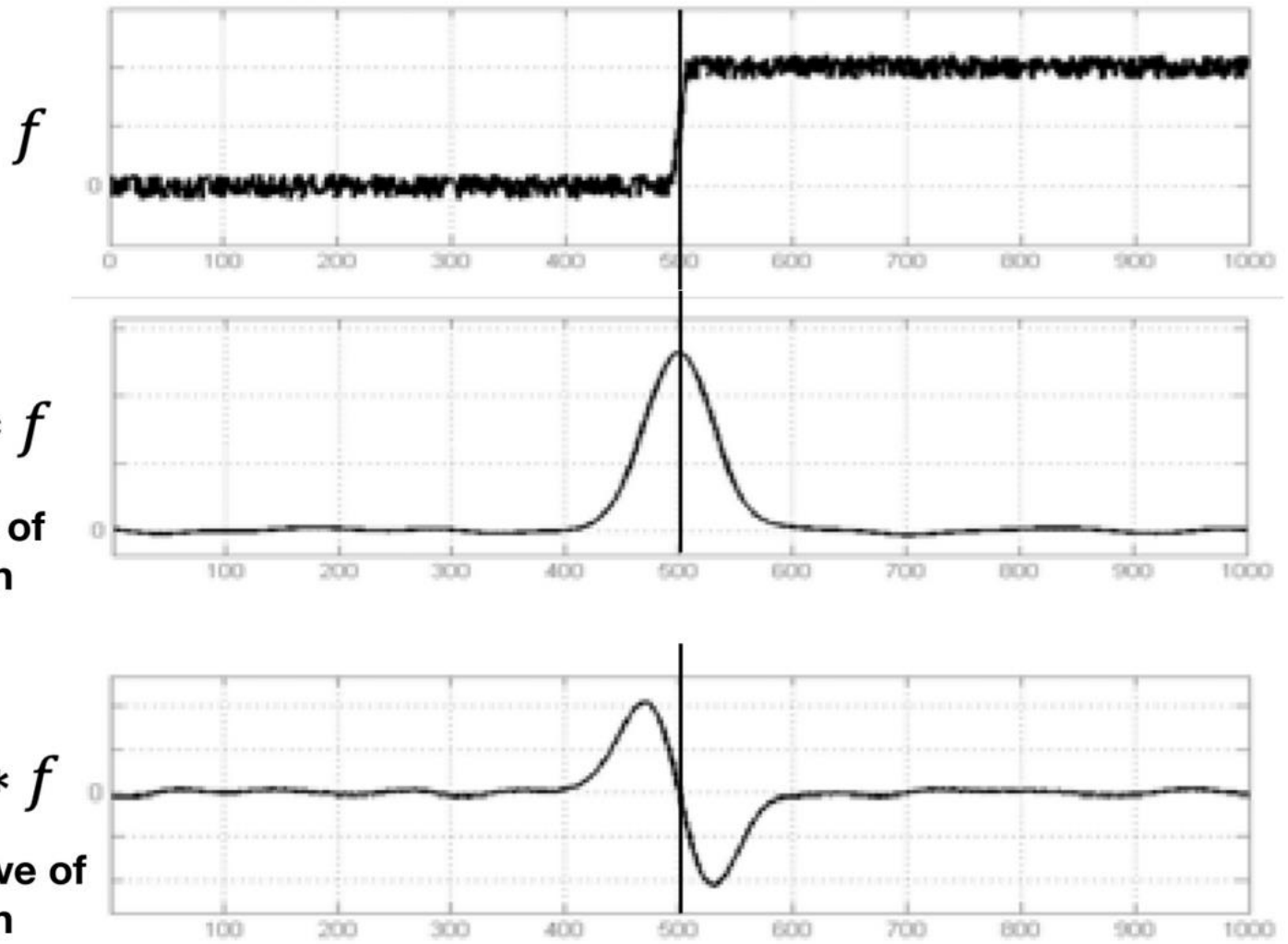
Problem: p he edgeare not discrimina e edge)

Derivative of Gaussian

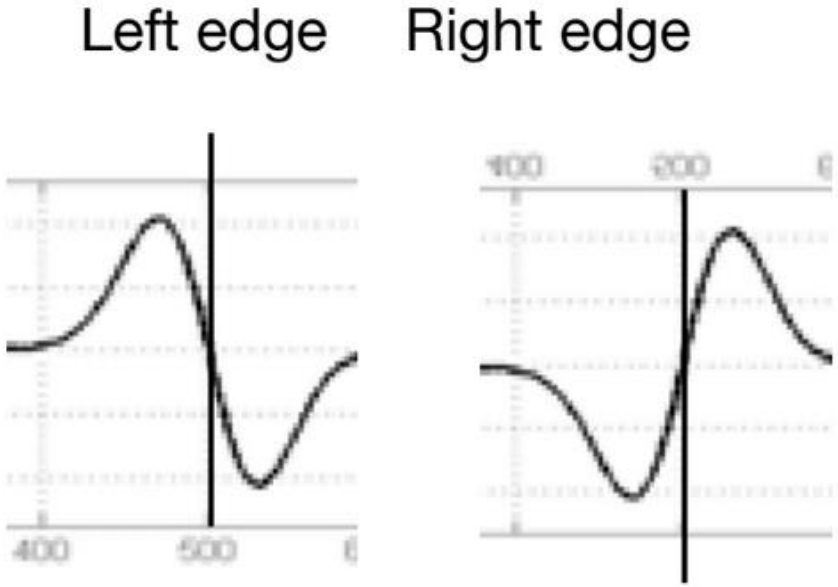
$$L(x, y, n_\sigma) = (G_\sigma * f)(x, y)$$

$$\nabla L = \left(\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y}\right)$$

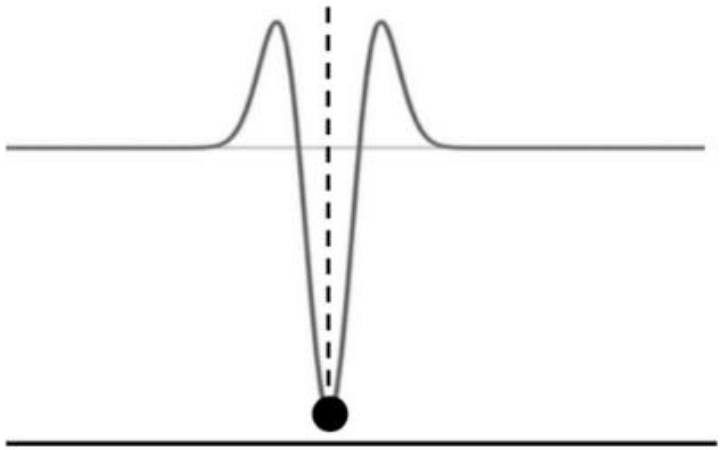
$$\nabla^2 L = \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}$$



2nd Derivative of Gaussian at a blob



If we pick the “right” sigma, two valleys overlap ➔ blob detection



2D blob detector

- Normalized Laplacian of Gaussian (NLoG) is used as the 2D equivalent for blob detection

Laplacian	Gaussian	LoG	NLoG
$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$			
n_σ	$\nabla^2 n_\sigma$	$\sigma^2 \nabla^2 n_\sigma$	

Laplacian filter (sigma=0)

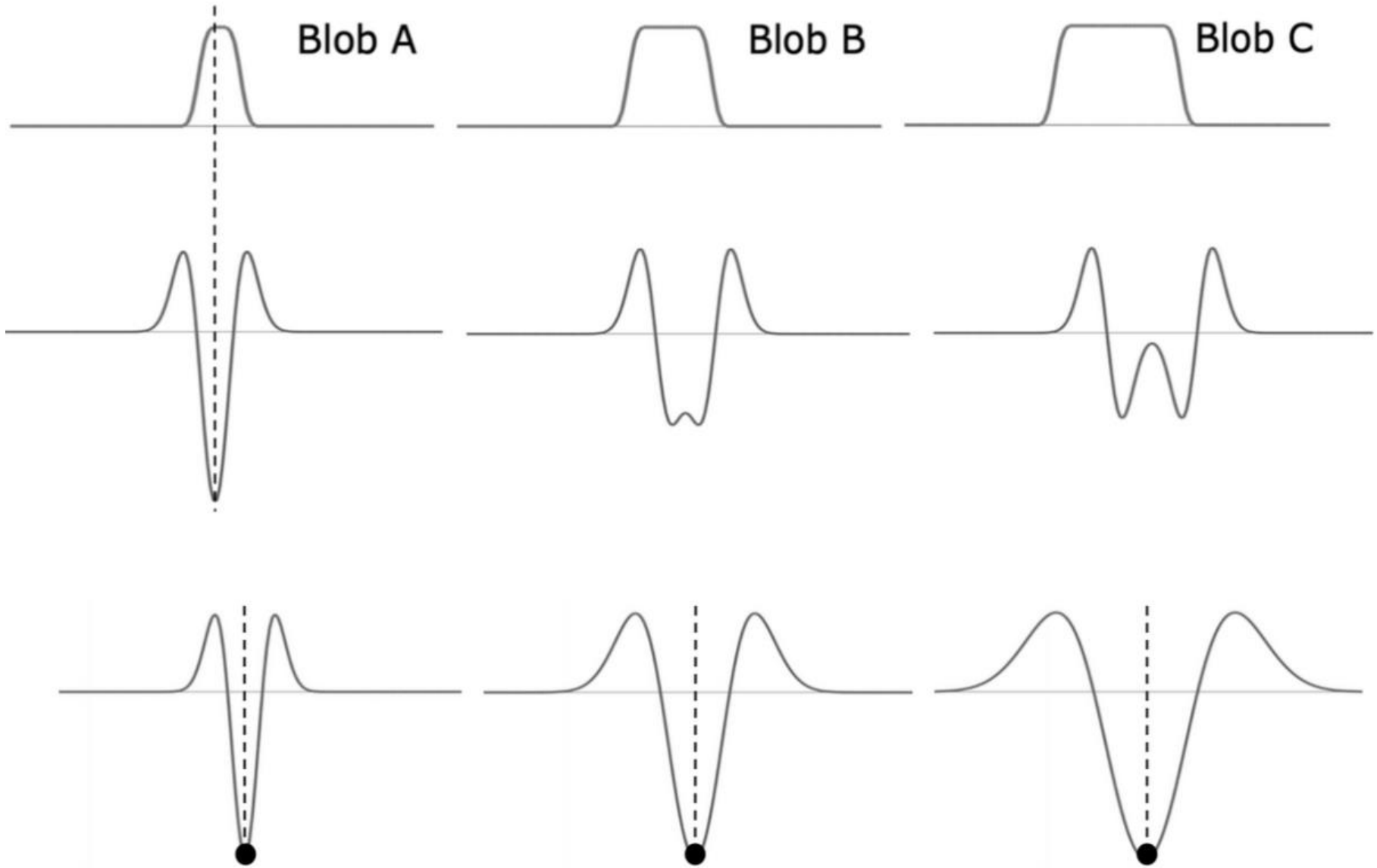
$$\circ \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & -1 \\ 0 & 1 & 0 \end{bmatrix} =$$

Need different sigma for different scales of the blob

$f(x)$

$\sigma^2 \frac{\partial^2 n_\sigma}{\partial x^2} * f(x)$

Bigger sigma



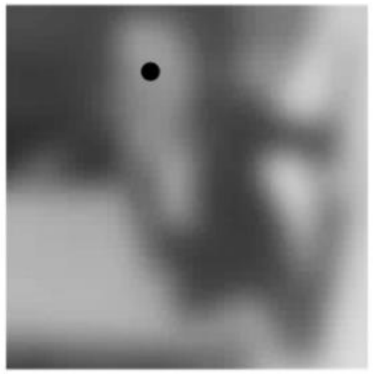
Creating scale-space

$S(x, y, \sigma_0)$ $S(x, y, \sigma_1)$ $S(x, y, \sigma_2)$ $S(x, y, \sigma_3)$

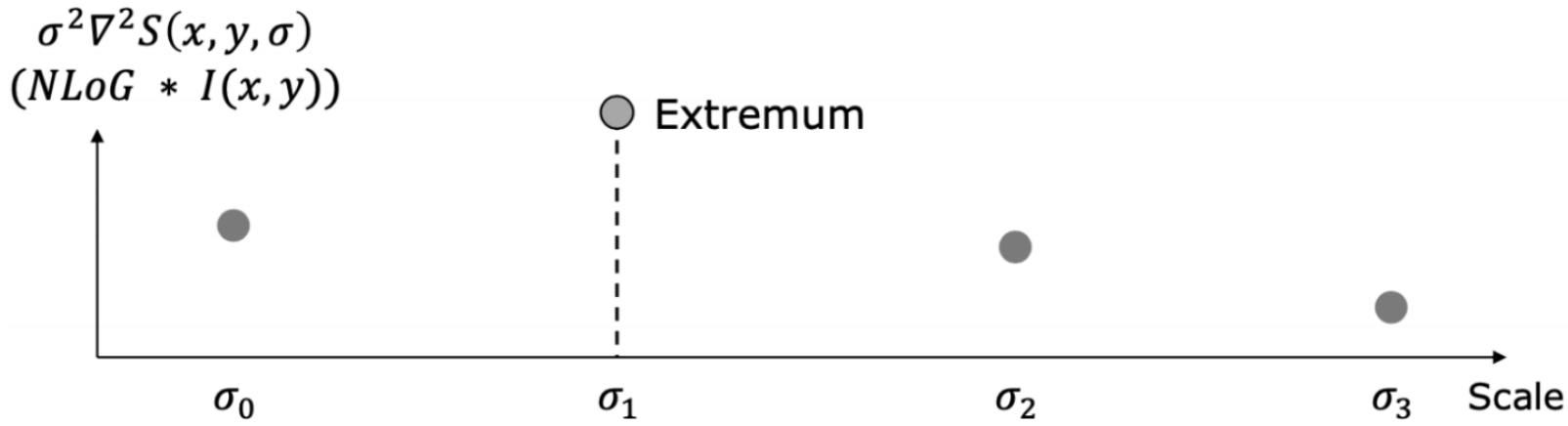
Increasing σ, higher scale, lower resolution

Blob detection using local extrema



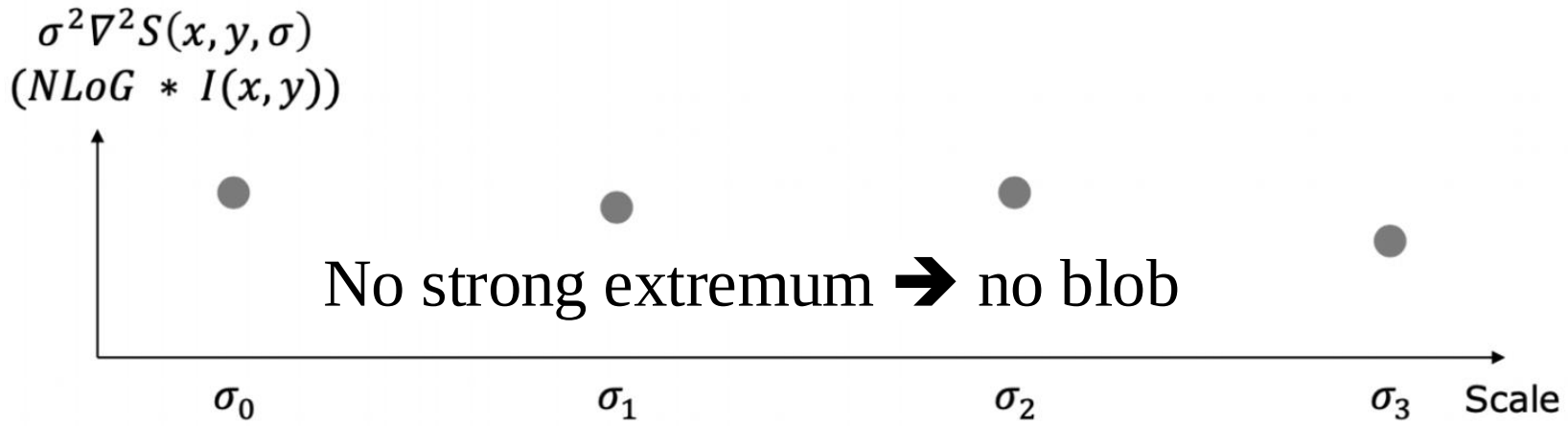
$S(x, y, \sigma_0)$ $S(x, y, \sigma_1)$ $S(x, y, \sigma_2)$ $S(x, y, \sigma_3)$



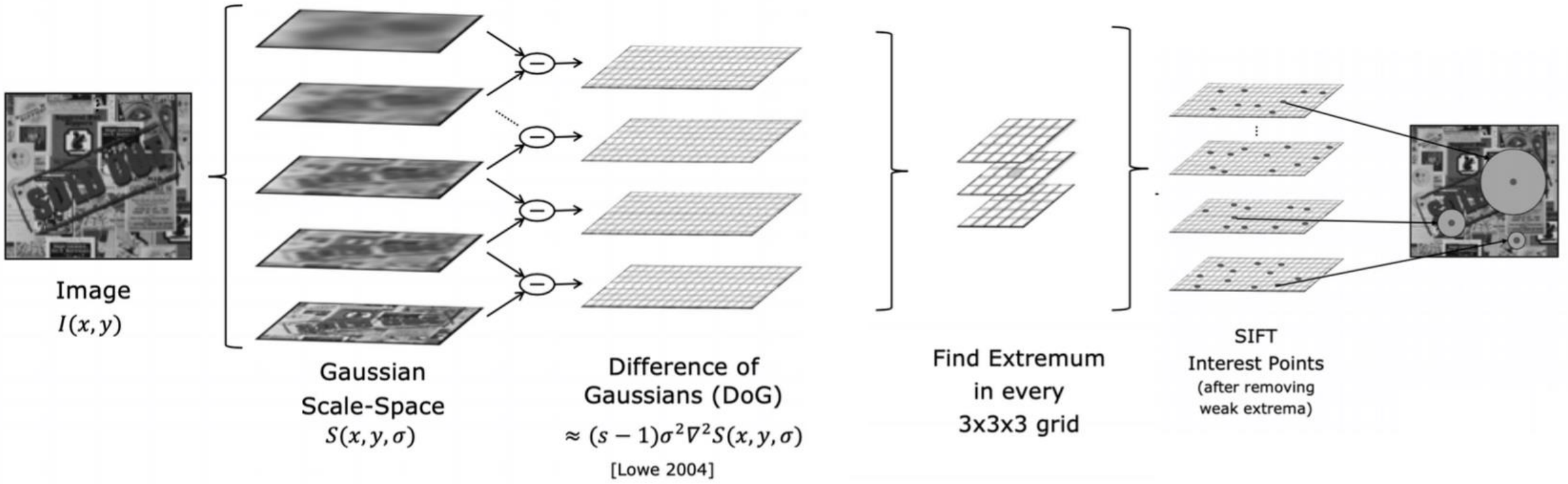
Characteristic scale σ^*

Blob detection using local extrema

$S(x, y, \sigma_0)$ $S(x, y, \sigma_1)$ $S(x, y, \sigma_2)$ $S(x, y, \sigma_3)$

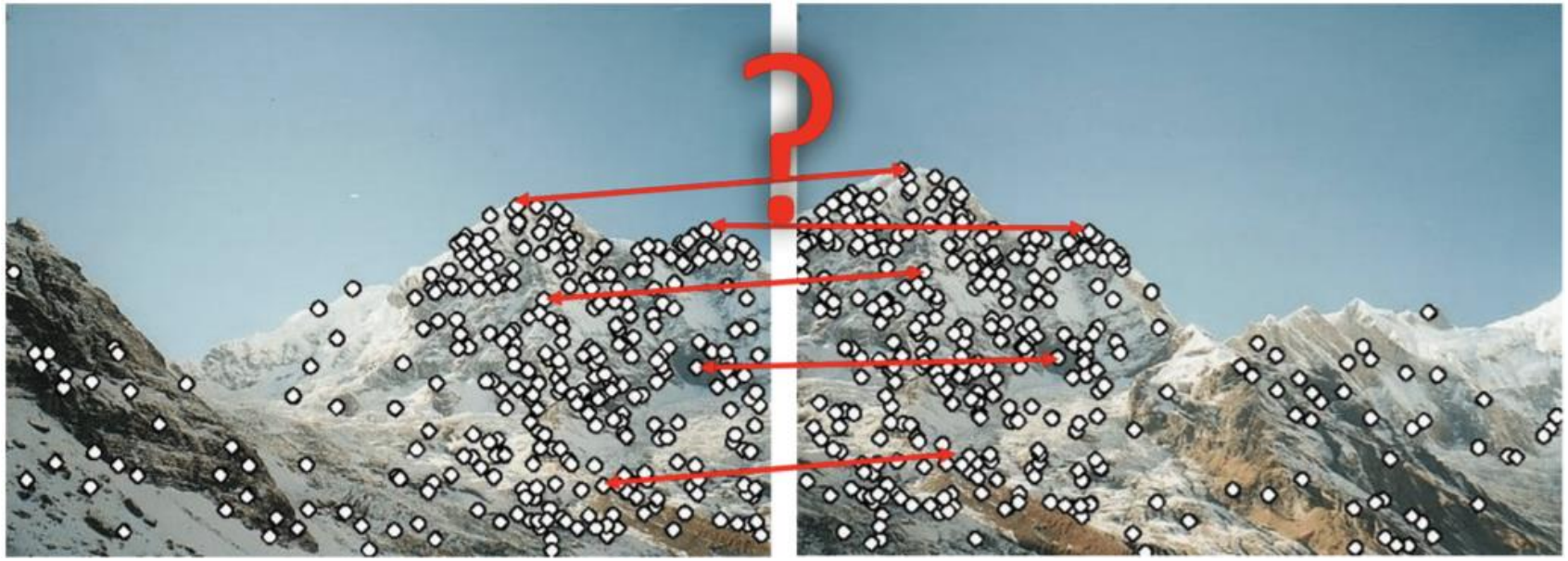


Scale-invariant feature transformation (SIFT-detector)



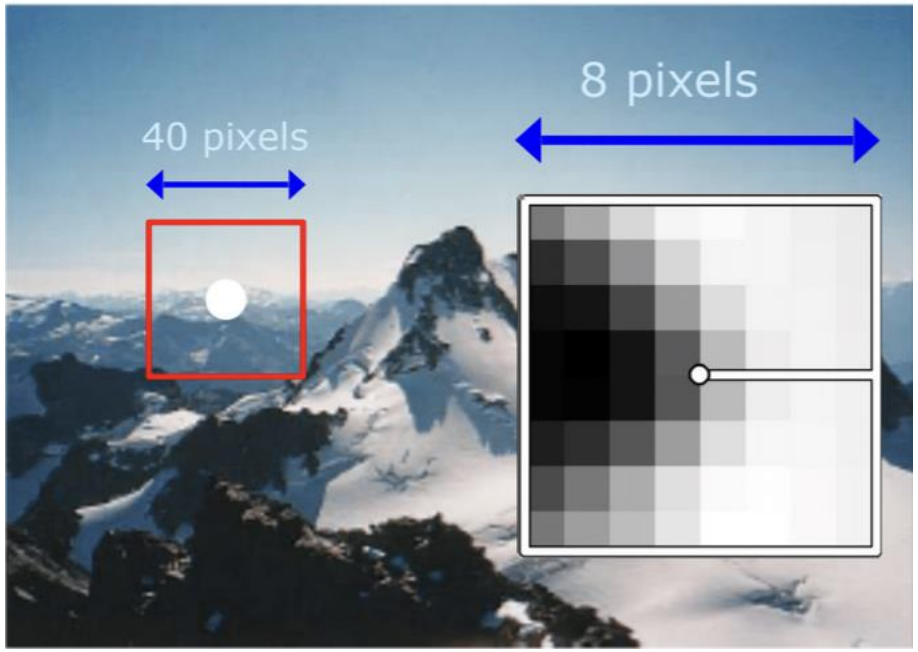
Feature descriptors

- We know how to detect good points [(x,y): position]
- Next question: How to describe them? (Feature vector: appearance)



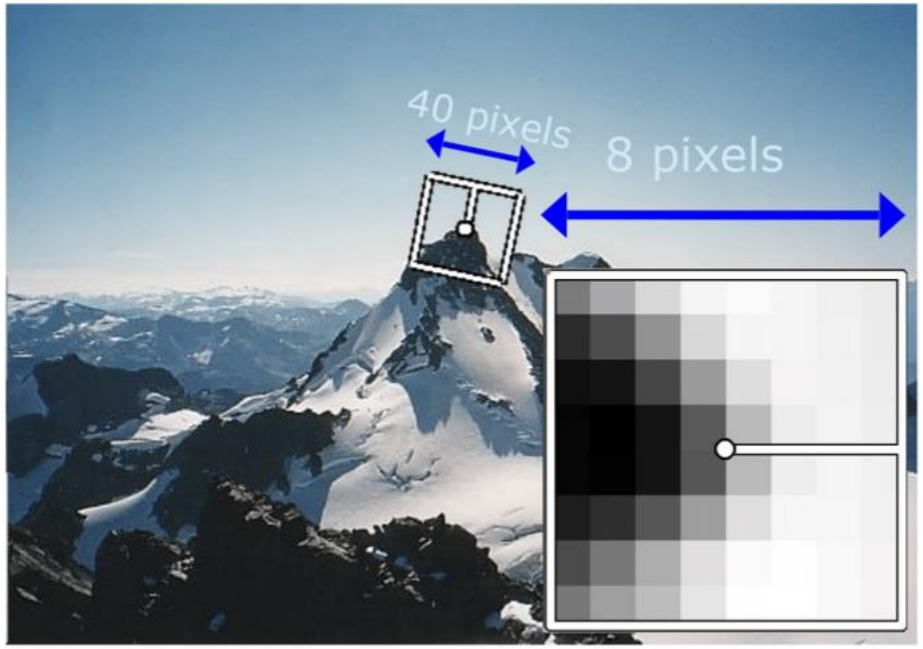
Brightness invariance

- Normalized image patch
  - Take 40x40 window around key point
  - Downsample to 8x8
  - Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window

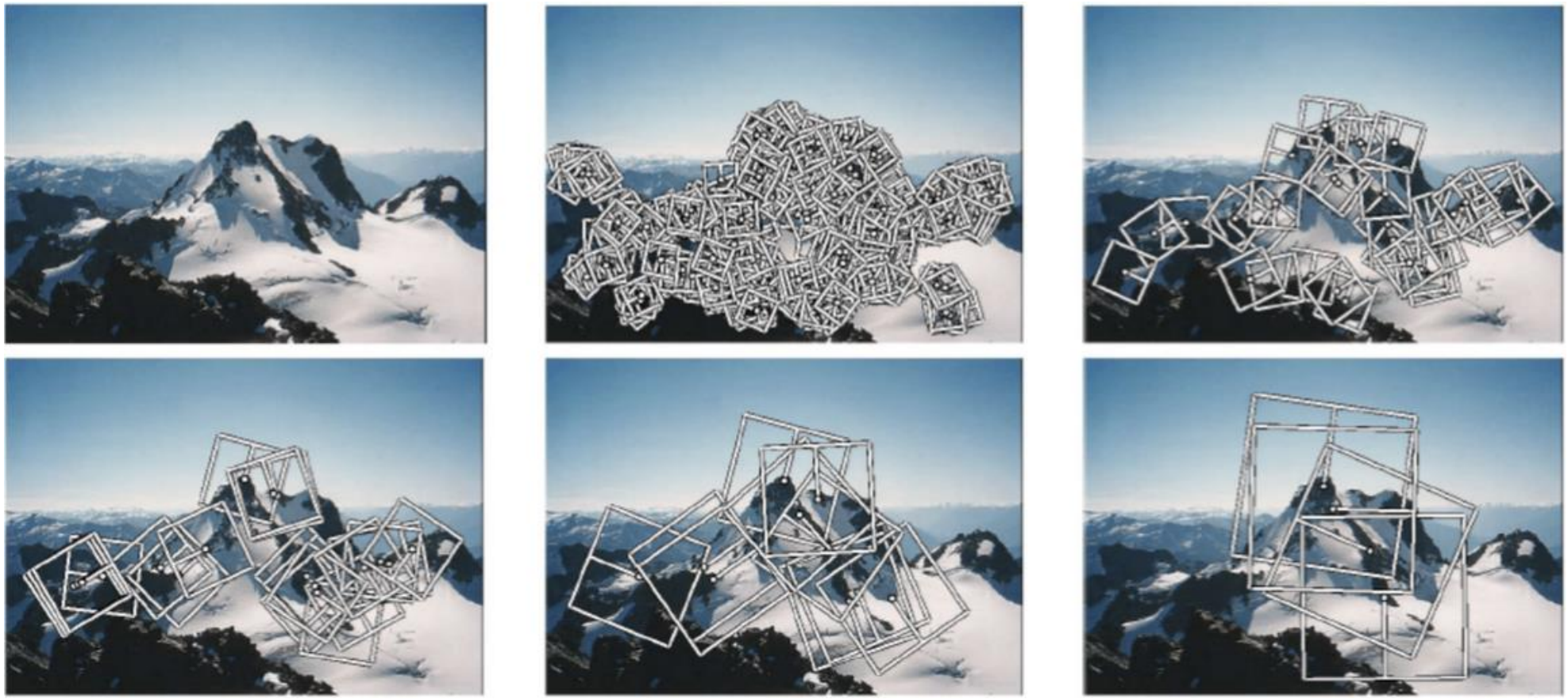


Orientation invariance

- We want invariance to rotation, lighting, and tiny spatial shifts
  - Take 40x40 window around key point
  - Find dominant orientation
  - Rotate the patch to horizontal orientation
  - Downsample to 8x8
  - Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



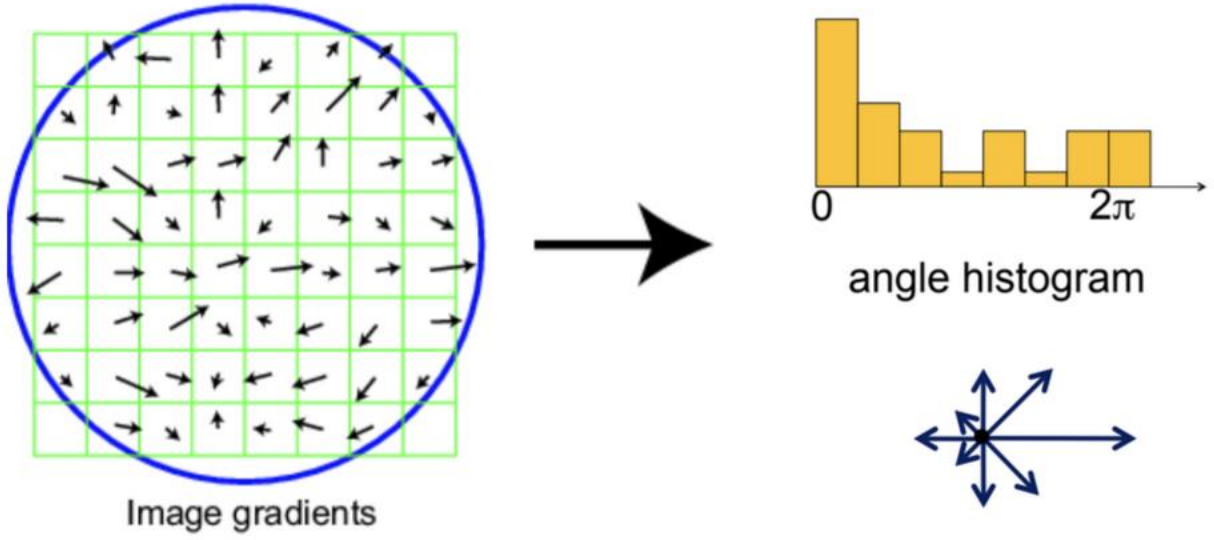
Scale invariance



Multi-scale Oriented Patches (MOPs) extracted at five pyramid levels from one of the Matier Images. The boxes show the feature orientation and the region from which the descriptor vector is sampled

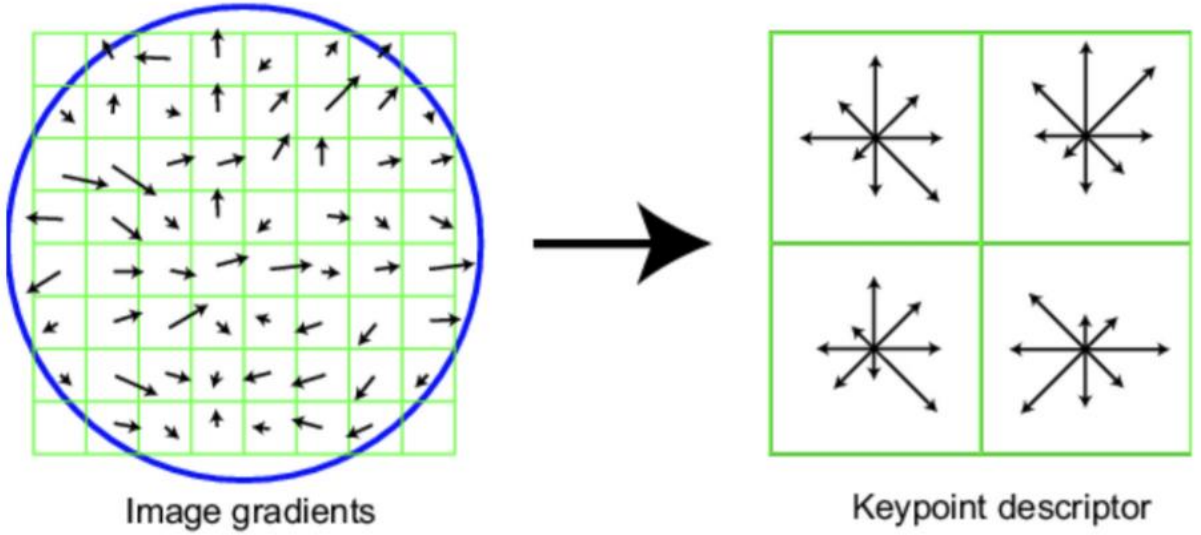
Color invariance

- Gradients capture intensity changes rather than absolute color values.
  - Take 16x16 square window around detected feature
  - Compute edge orientation for each pixel
  - Create histogram of edge orientations



SIFT-descriptor

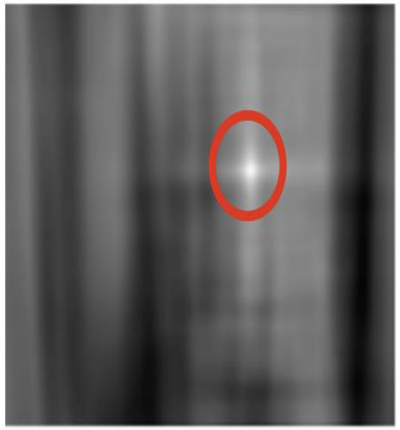
- Create the descriptor:
  - Rotation invariance: rotate by “dominant” orientation
  - Spatial invariance: spatial pool to 2x2
  - Compute an orientation histogram for each cell
  - (4 x 4) cells x 8 orientations = 128 dimensional descriptor



Method 2: feature-based approach

1. What to match
2. How to match (NCC)
3. Match → transformation

Intensity-based

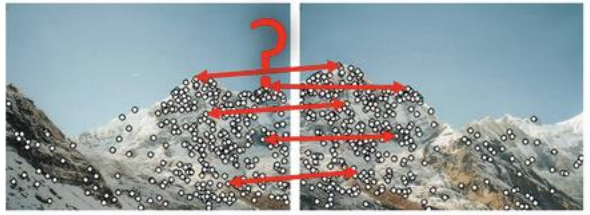


Translation

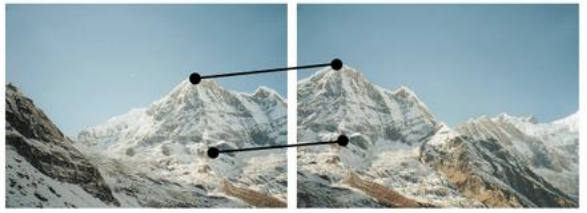
$$T: \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x - x_2 + x_1 \\ y - y_2 + y_1 \end{bmatrix}$$

Feature-based

Find key points



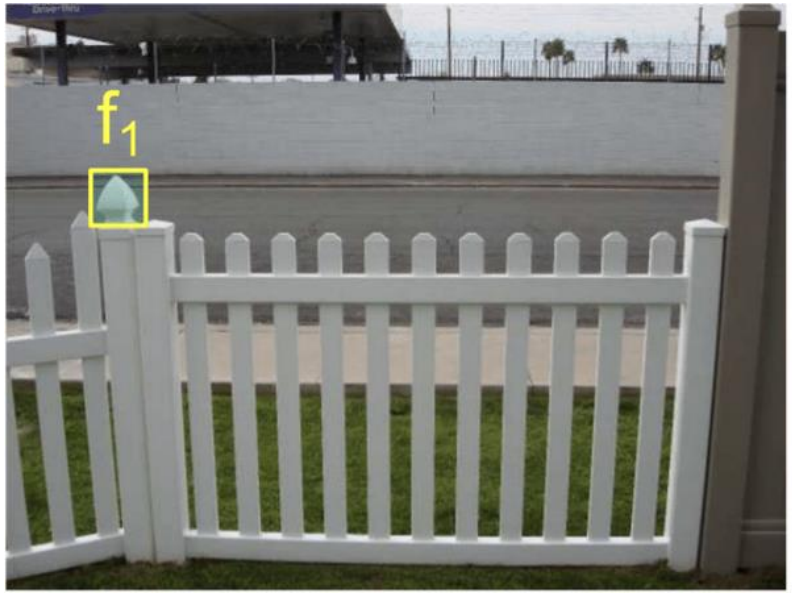
Many matches

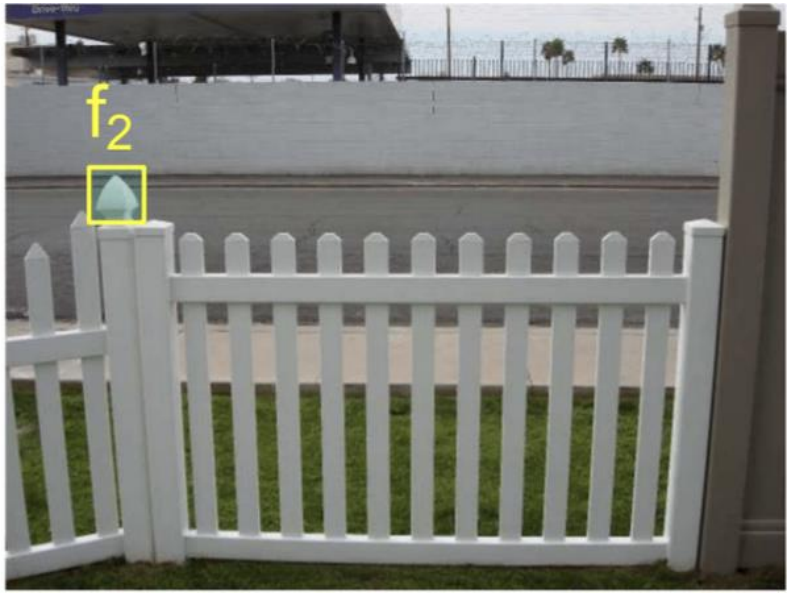


Find inliers and $\{(x, y), (x', y')\}_K \rightarrow T$

How can we tell if two features match?

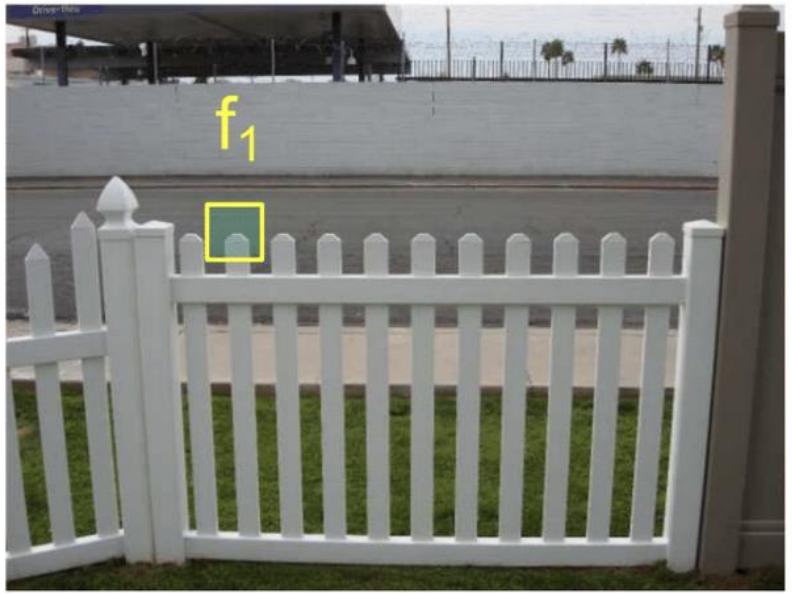
- Simple approach: are they the nearest neighbor in L2 distance, $\|f_1 - f_2\|$?
- Problem?

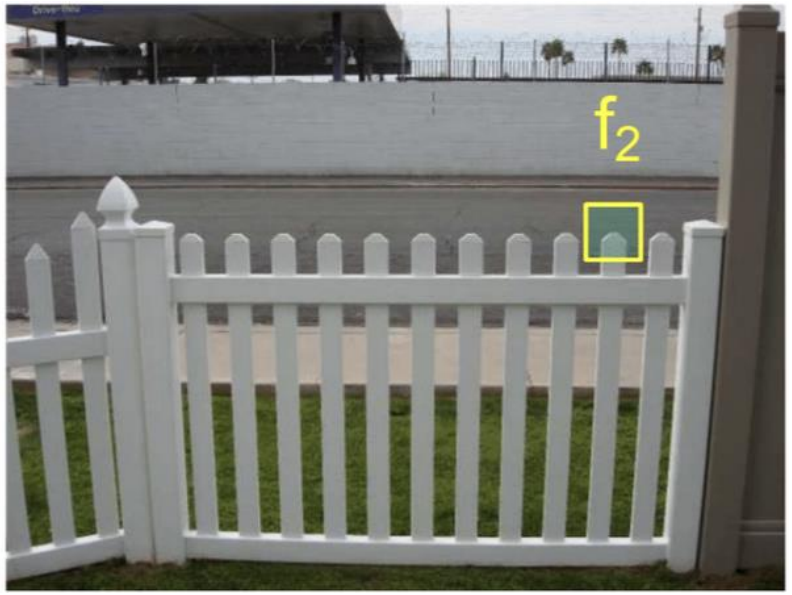




How can we tell if two features match?

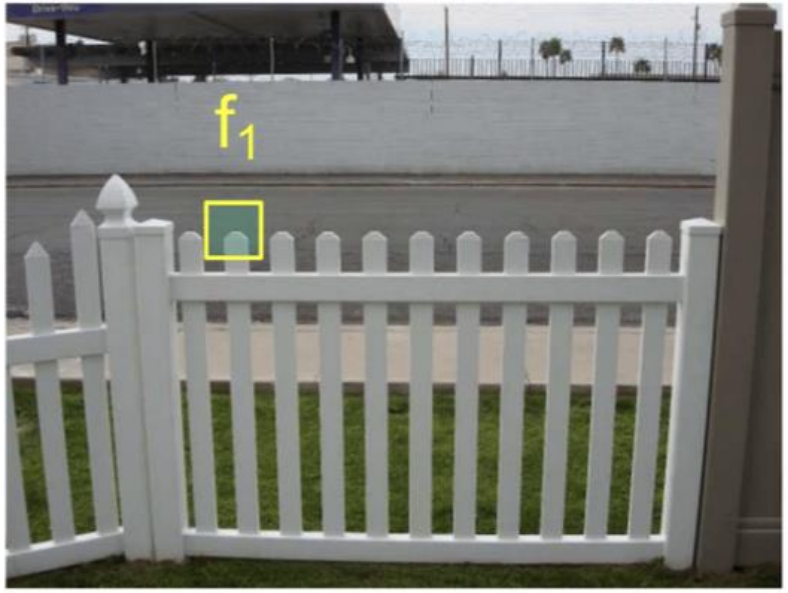
- Simple approach: are they the nearest neighbor in L2 distance, $\|f_1 - f_2\|$?
- Problem: can give good scores to ambiguous (incorrect) matches

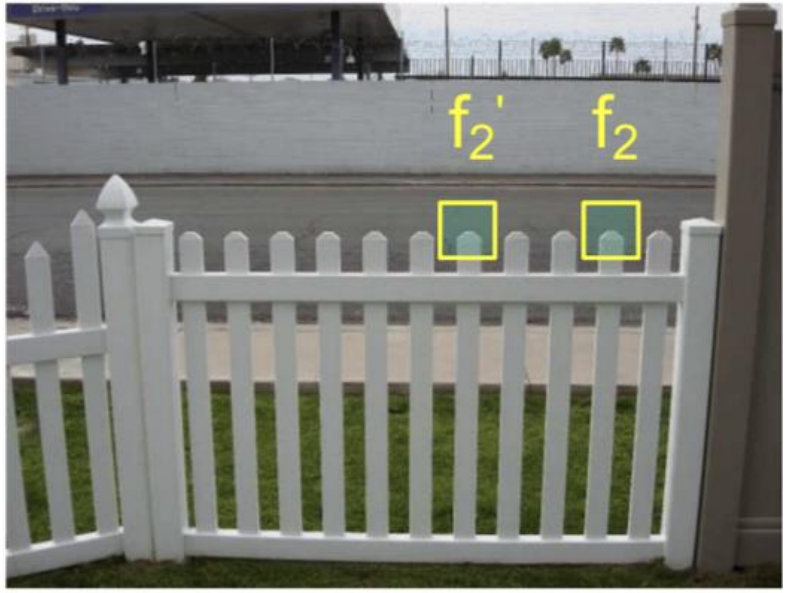




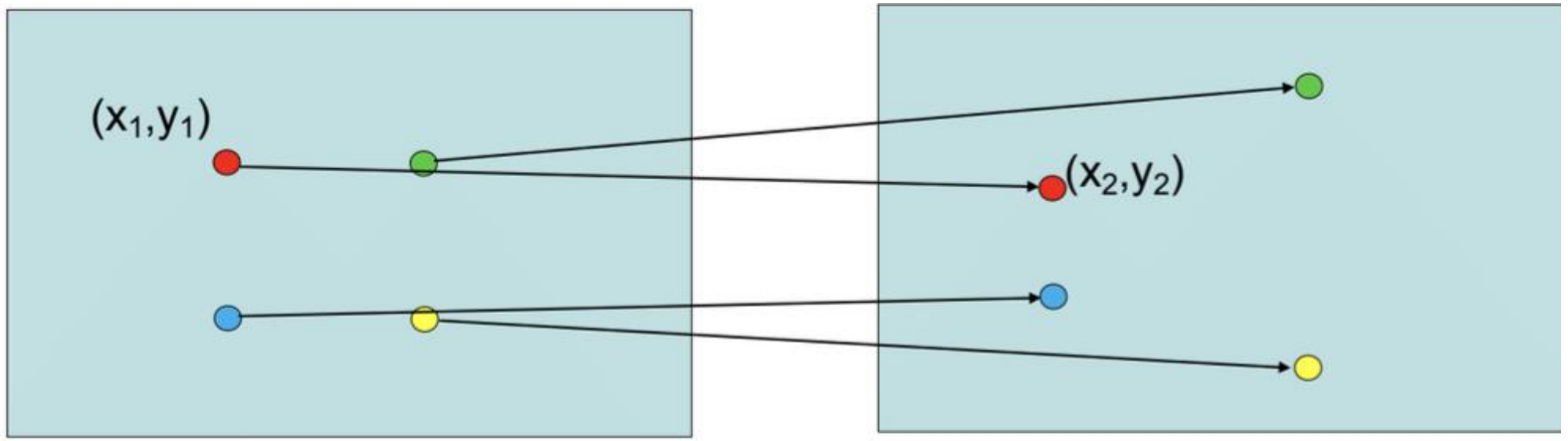
How can we tell if two features match?

- Throw away matches that fail tests:
  - Rotation test: best match should be “better” with a big margin
    - Ratio distance: $\|f_1 - f_2\| / \|f_1 - f_2'\|$
    - f_2 is best Sum of Squared Differences (SSD) match to f_1 in I_2
    - f_2' is 2nd best SSD match to f_1 in I_2
  - Forward-backward consistency: f_1 should also be nearest neighbor of f_2





Matches → Transformation matrix



$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \times \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

(Affine: rotation/scale/translation)