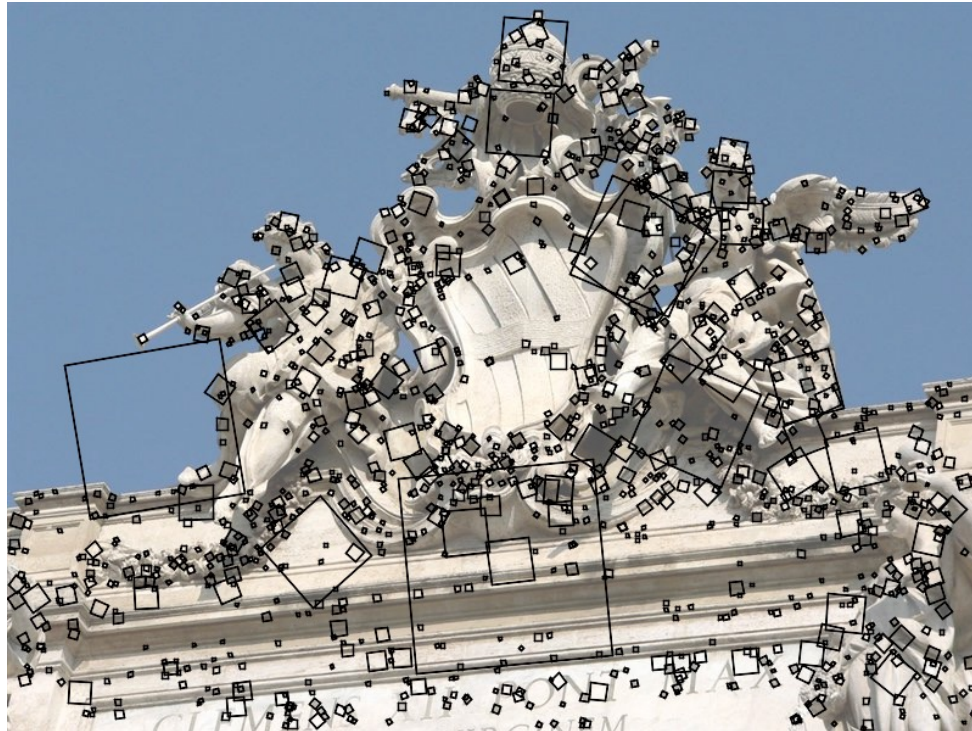


SIFT keypoint detection



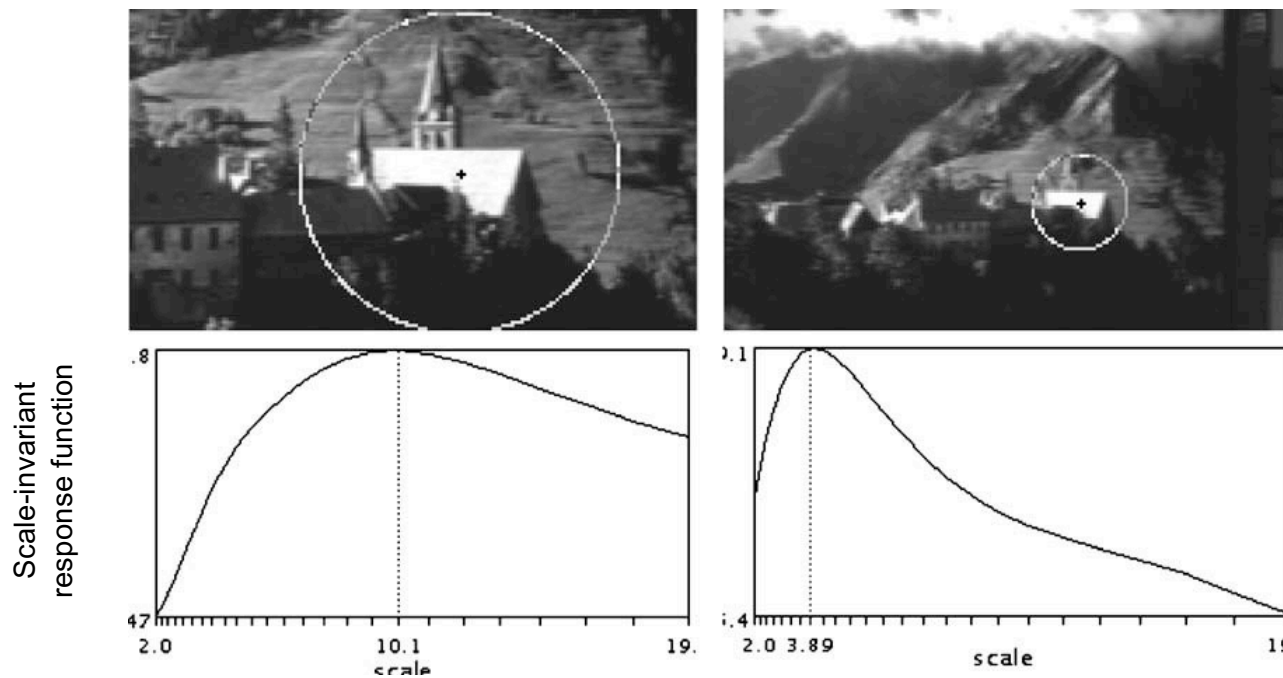
D. Lowe, [Distinctive image features from scale-invariant keypoints](#),
IJCV 60 (2), pp. 91-110, 2004

Overview

- Multiscale blob detection: Key idea
- Laplacian of Gaussian filter
- Multiscale blob detection algorithm
- SIFT detector
- SIFT descriptors

Keypoint detection with scale selection

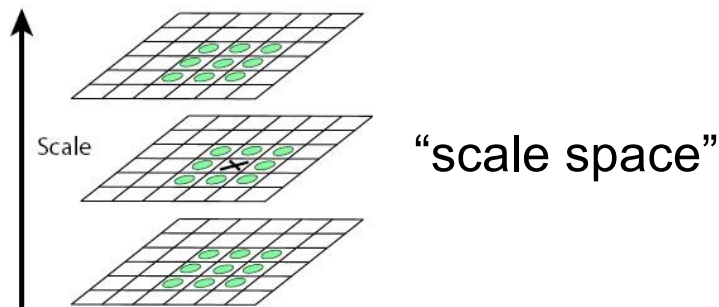
- We want to extract keypoints with *characteristic scales* that are *equivariant* (or *covariant*) w.r.t. to scaling of the image



K. Mikolajczyk and C. Schmid. [Indexing based on scale invariant interest points](#). ICCV 2001
T. Lindeberg, [Feature detection with automatic scale selection](#), *IJCV* 30(2), pp. 77-116, 1998

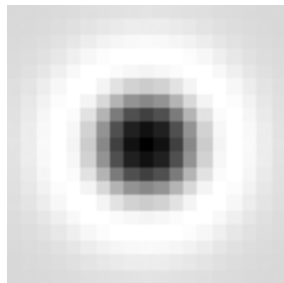
Keypoint detection with scale selection

- We want to extract keypoints with *characteristic scales* that are *equivariant* (or *covariant*) w.r.t. to scaling of the image
- Approach: compute a *scale-invariant* response function over neighborhoods centered at each location (x, y) and a range of scales (σ) , find *scale-space locations* (x, y, σ) where this function reaches a local maximum



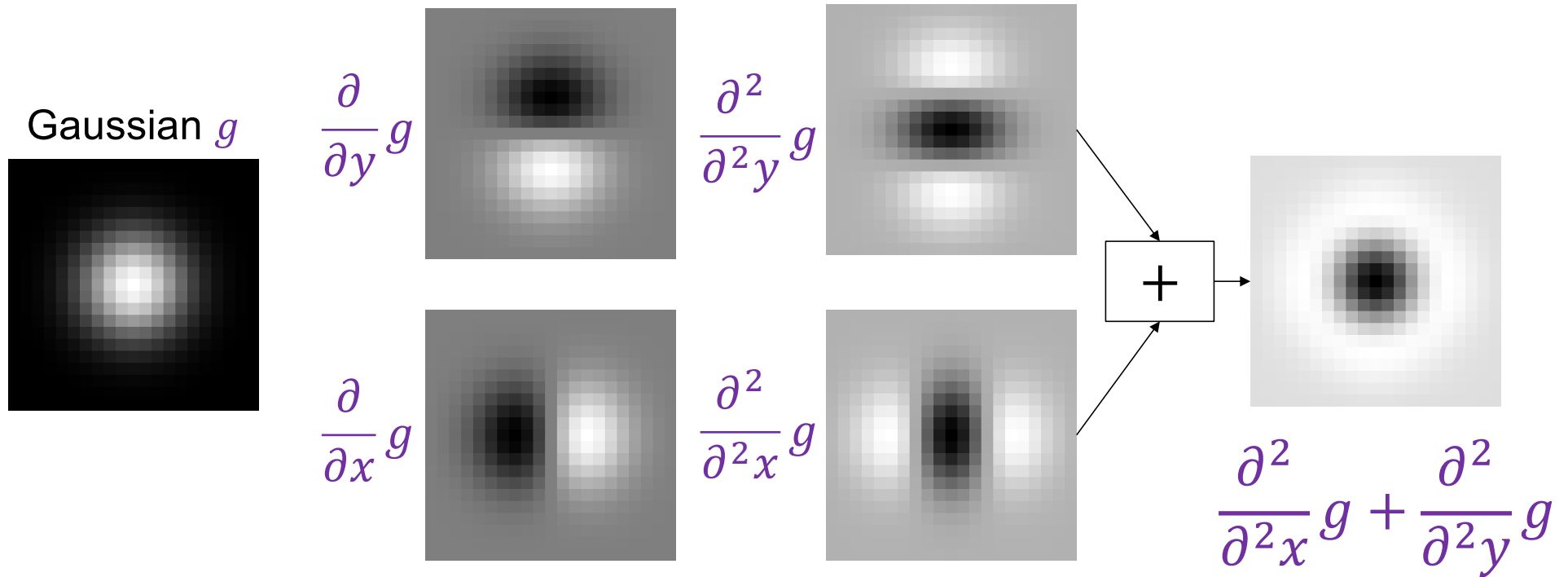
Keypoint detection with scale selection

- We want to extract keypoints with *characteristic scales* that are *equivariant* (or *covariant*) w.r.t. to scaling of the image
- Approach: compute a *scale-invariant* response function over neighborhoods centered at each location (x, y) and a range of scales (σ) , find *scale-space locations* (x, y, σ) where this function reaches a local maximum
- A particularly convenient response function is given by the *scale-normalized Laplacian of Gaussian (LoG) filter*:



$$\nabla_{\text{norm}}^2 = \sigma^2 \left(\frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial y^2} g \right)$$

Laplacian of Gaussian

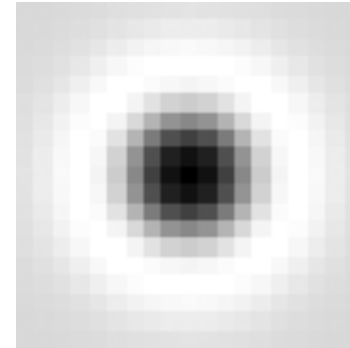
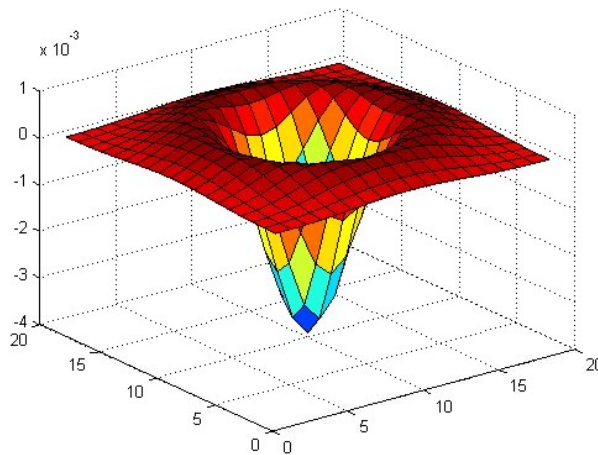


Source: [J. Johnson and D. Fouhey](#)

Is this filter separable?

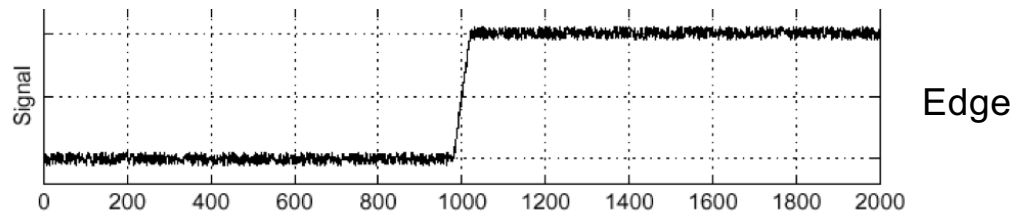
Scale-normalized Laplacian

- You need to multiply the LoG by σ^2 to make responses comparable across scales

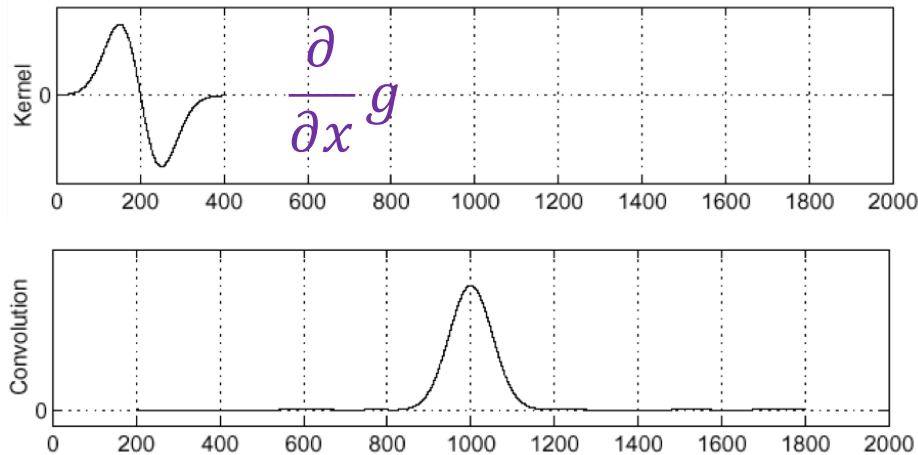


$$\nabla_{\text{norm}}^2 = \sigma^2 \left(\frac{\partial^2}{\partial x^2} g + \frac{\partial^2}{\partial y^2} g \right)$$

Edge detection with LoG

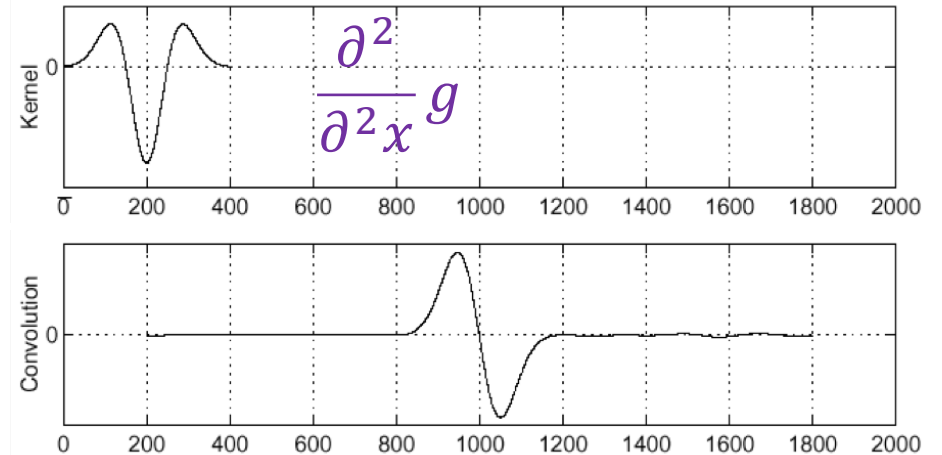


Derivative of Gaussian



Edge = *local maximum* of derivative

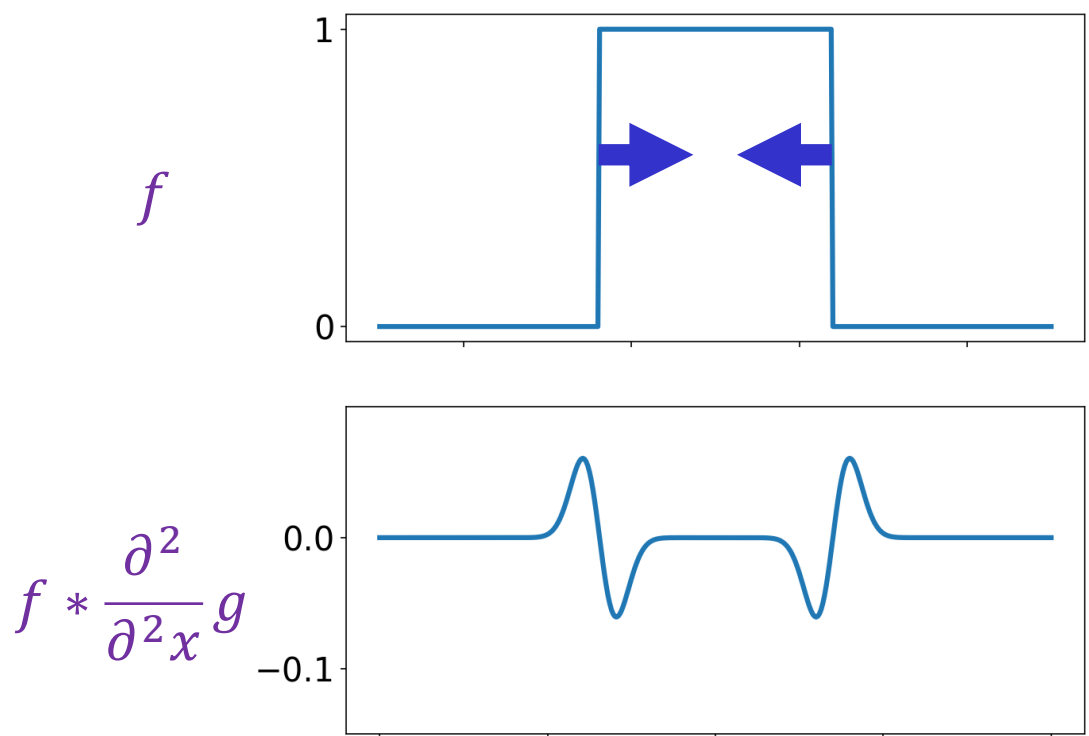
Laplacian of Gaussian



Edge = *zero-crossing* of Laplacian

Blob detection with LoG

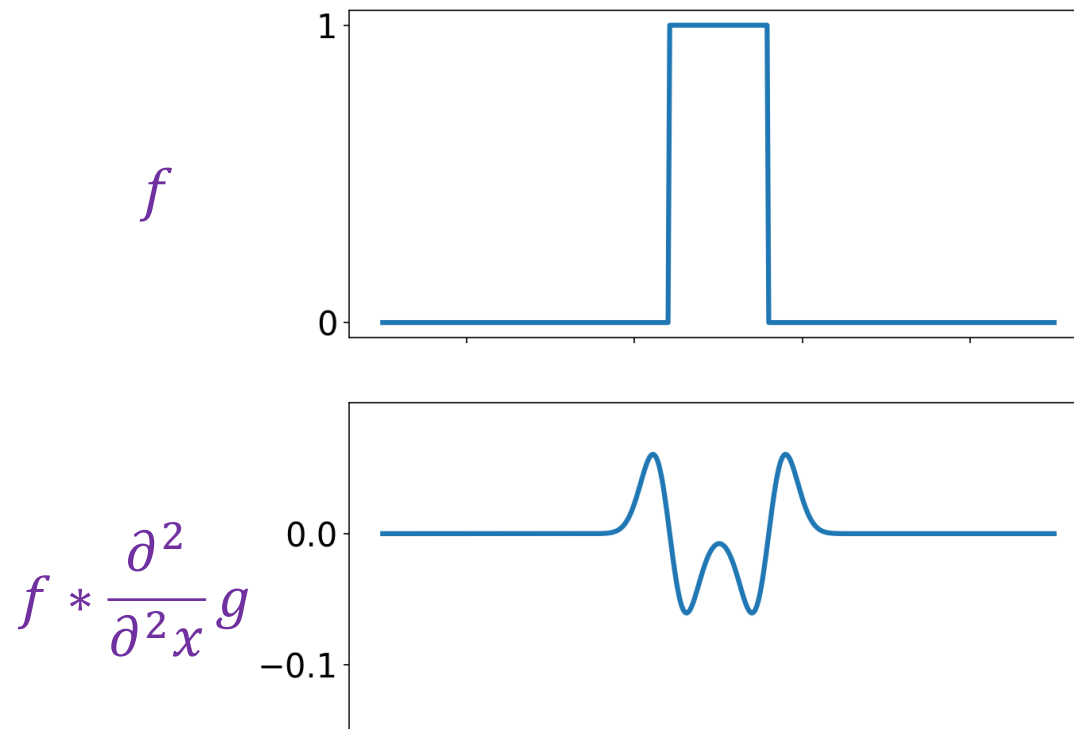
- Let's convolve a 1D "blob" with the Laplacian:



Source: [J. Johnson and D. Fouhey](#)

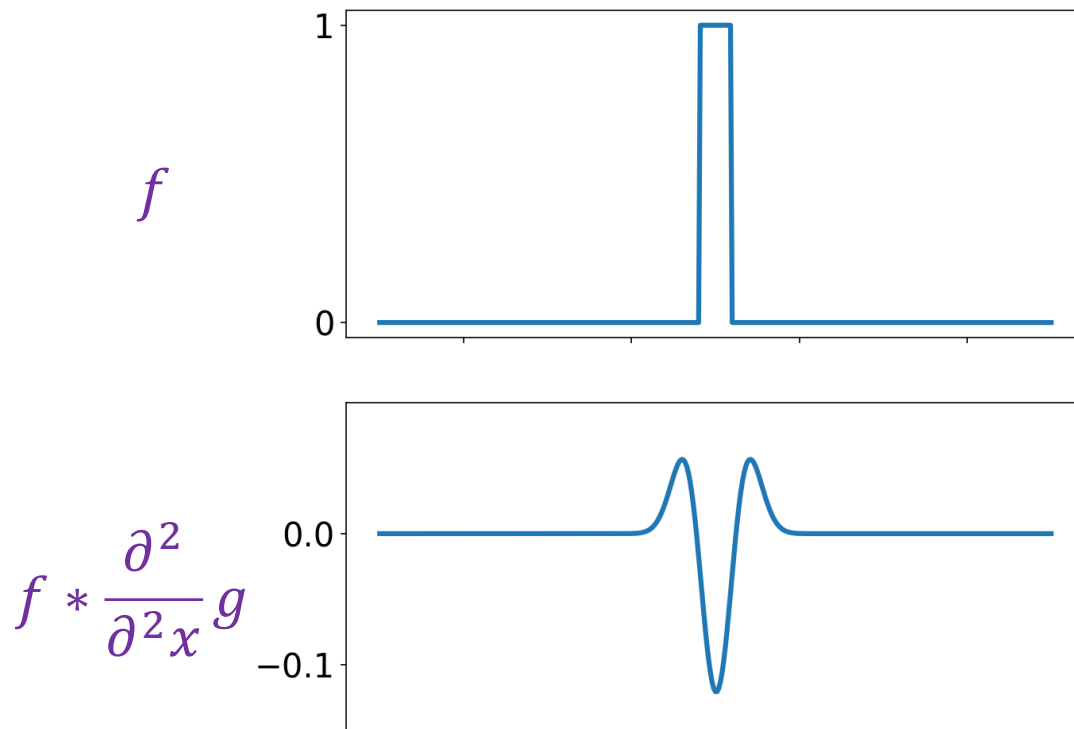
Blob detection with LoG

- Let's convolve a 1D "blob" with the Laplacian:



Blob detection with LoG

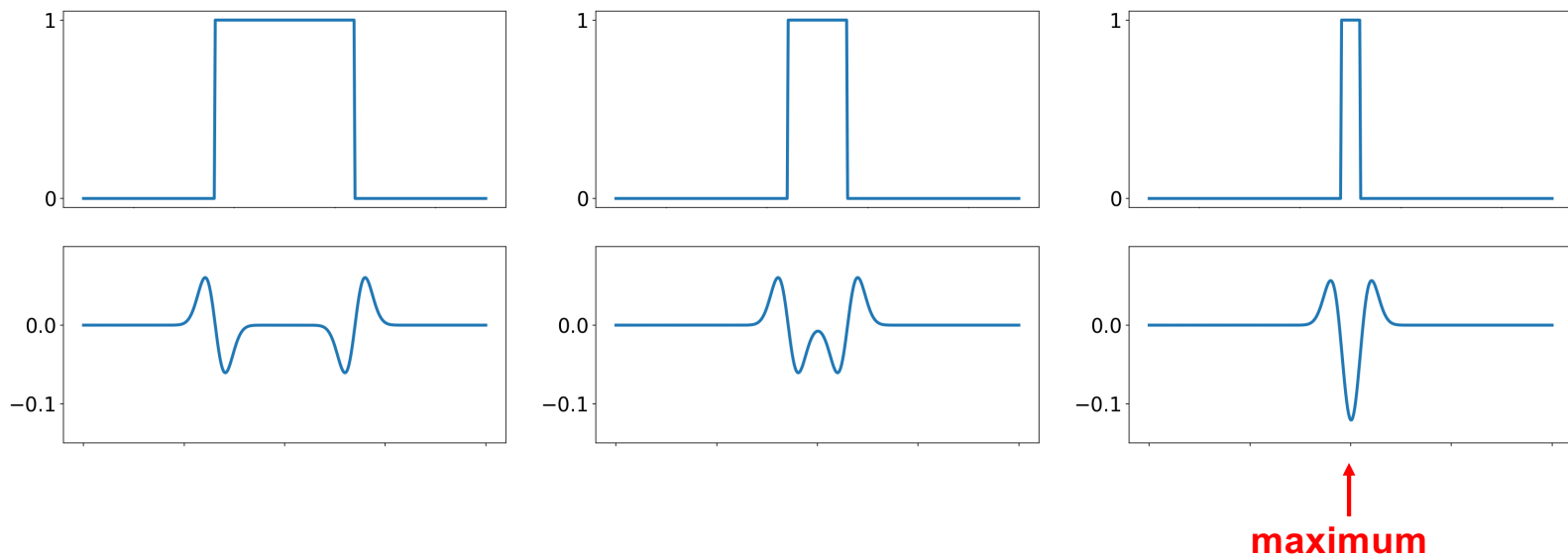
- Let's convolve a 1D "blob" with the Laplacian:



Source: [J. Johnson and D. Fouhey](#)

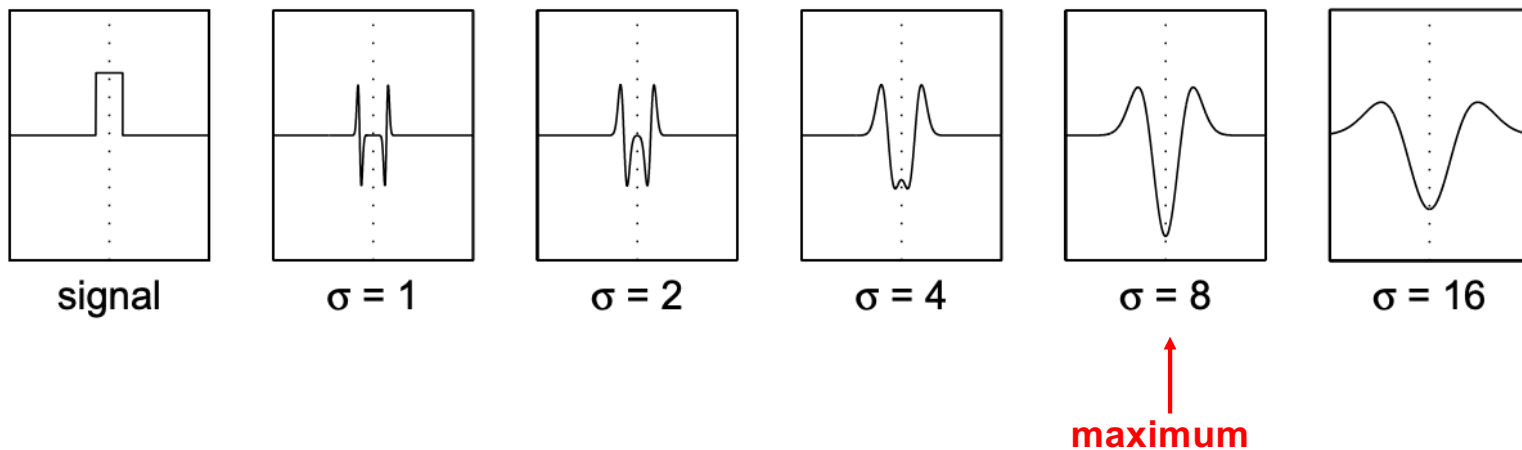
Spatial selection

- The magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is “matched” to the scale of the blob



Scale selection

- We can find the *characteristic scale* of the blob by convolving it with *scale-normalized* Laplacians at several scales (σ) and looking for the maximum response

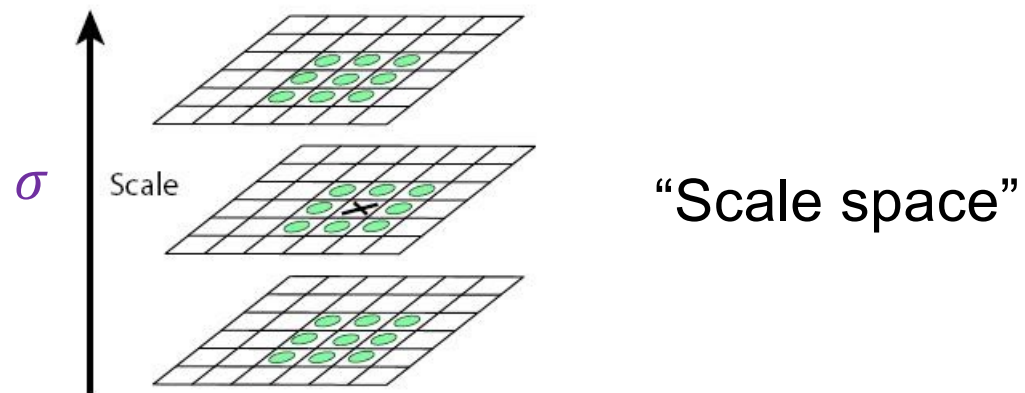


Overview

- Multiscale blob detection: Key idea
- Laplacian of Gaussian filter
- **Multiscale blob detection algorithm**

2D multiscale blob detection

1. Convolve image with *scale-normalized Laplacian of Gaussian* at several scales (values of σ)
2. Find maxima of squared Laplacian response in space *and* across scales



2D multiscale blob detection: Example



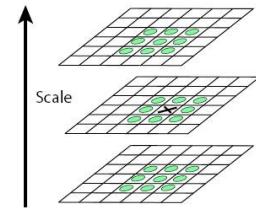
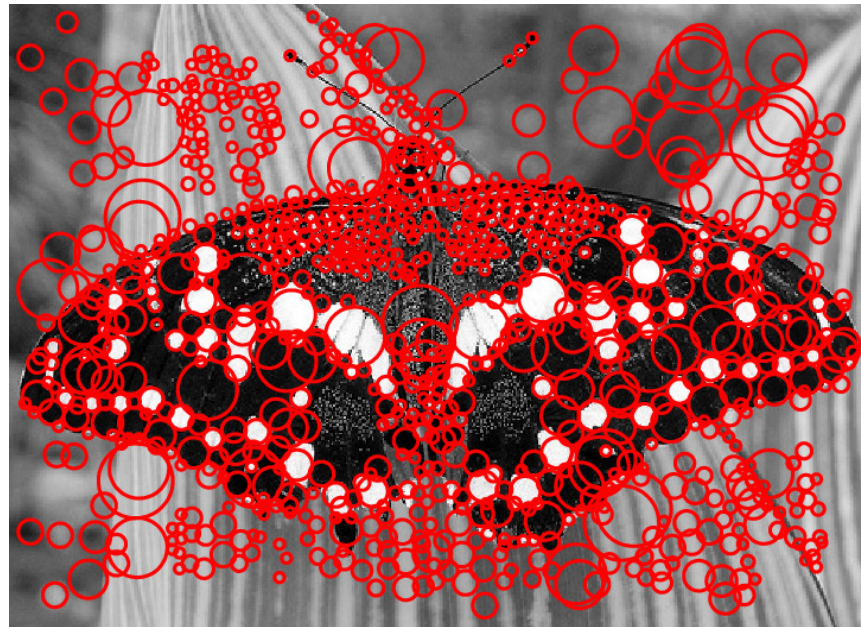
2D multiscale blob detection: Example



sigma = 11.9912

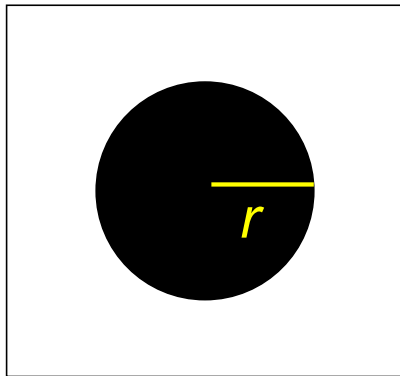
2D multiscale blob detection: Example

After scale-space NMS

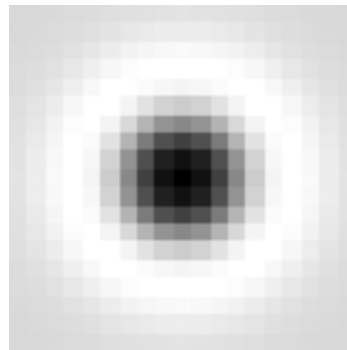


Technical detail 1: Drawing circles

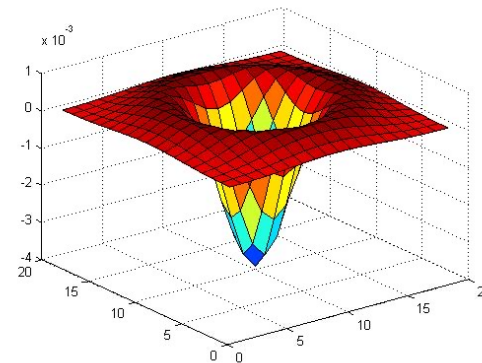
- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r ?



image

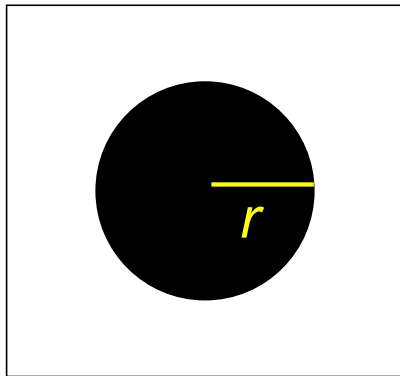


Laplacian

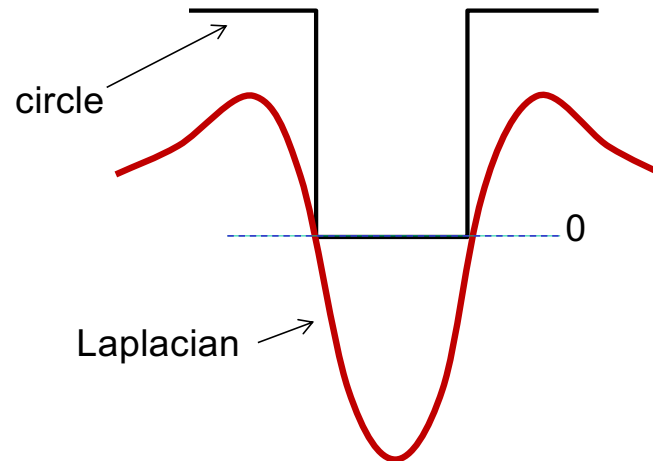


Technical detail 1: Drawing circles

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r ?
 - To get maximum response, the zeros of the Laplacian have to be aligned with the circle
 - Up to scale, the Laplacian is given by $(x^2 + y^2 - 2\sigma^2)e^{-(x^2+y^2)/2\sigma^2}$, so the maximum response occurs at $\sigma = r/\sqrt{2}$
- Therefore, for display purposes, you should multiply the characteristic scales of detected keypoints by $\sqrt{2}$

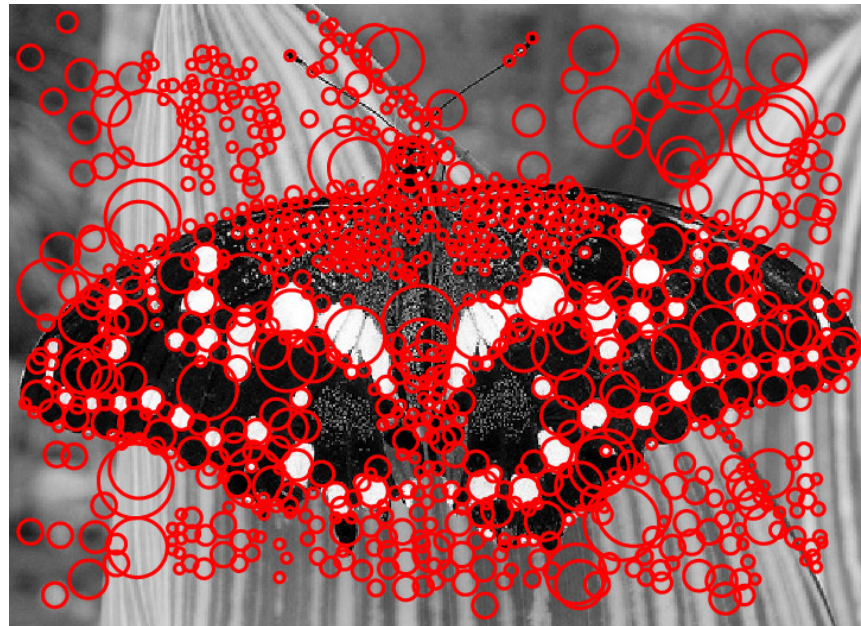


image



Technical detail 2: Eliminating edge responses

- Laplacian has strong response along edges



Technical detail 2: Eliminating edge responses

- Laplacian has strong response along edges

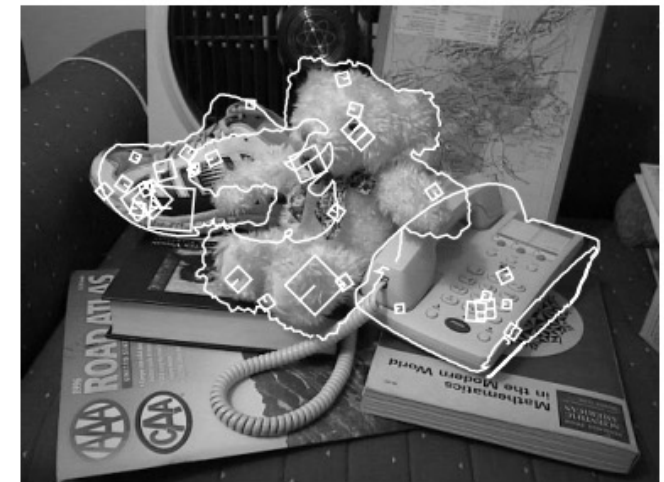


- Solution: filter based on Harris response function over neighborhoods containing the “blobs”

Overview

- Multiscale blob detection: Key idea
- Laplacian of Gaussian filter
- Multiscale blob detection algorithm
- SIFT detector
- SIFT descriptors

SIFT: Scale-invariant feature transform



D. Lowe. [Object recognition from local scale-invariant features](#). ICCV 1999

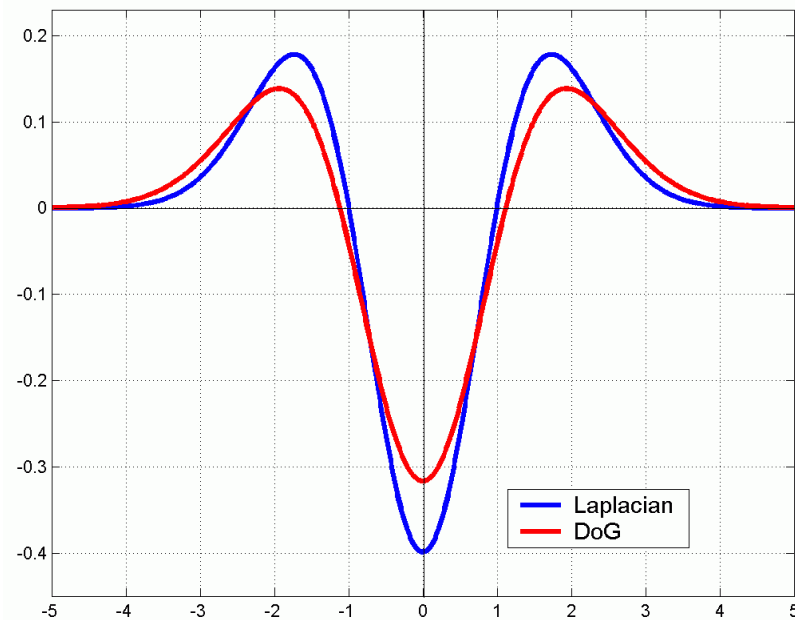
D. Lowe. [Distinctive image features from scale-invariant keypoints](#). *IJCV* 60 (2), pp. 91-110, 2004

SIFT detector: Efficient implementation

- Approximate LoG with a *difference of Gaussians* (DoG)

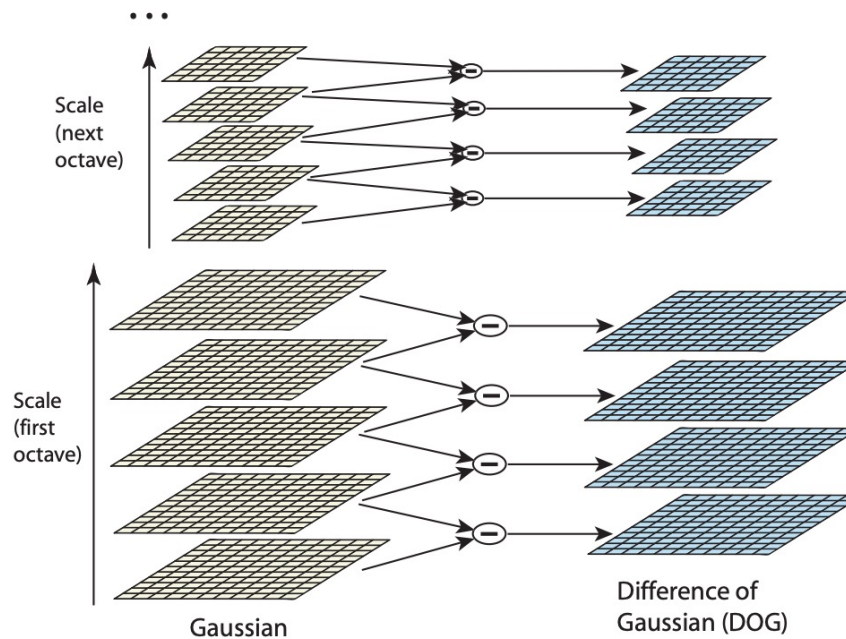
- Laplacian: $\sigma^2(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma))$

- DoG: $G(x, y, k\sigma) - G(x, y, \sigma)$



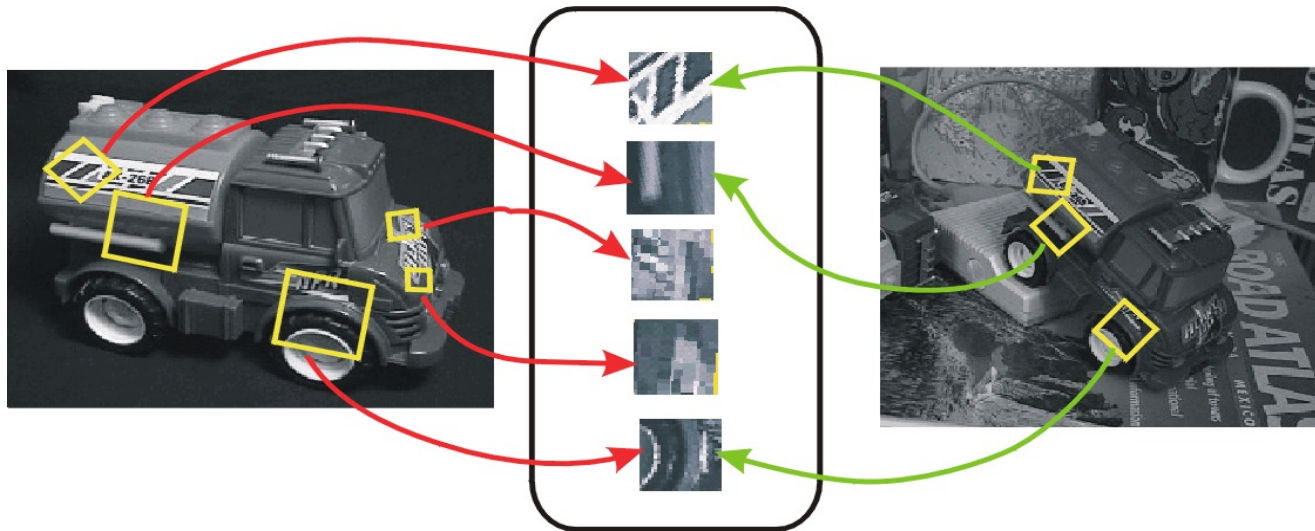
SIFT detector: Efficient implementation

- Approximate LoG with a *difference of Gaussians* (DoG)
 - Laplacian: $\sigma^2(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma))$
 - DoG: $G(x, y, k\sigma) - G(x, y, \sigma)$
- Compute DoG via an *image pyramid*:



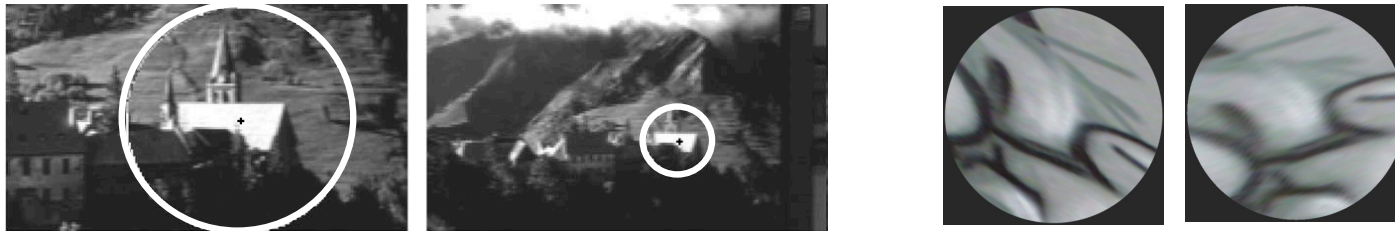
SIFT for matching

- The main goal of SIFT is to enable image matching in the presence of significant transformations
 - To recognize the same keypoint in multiple images, we need to match appearance descriptors or “signatures” in their neighborhoods
 - Descriptors that are *locally* invariant w.r.t. **scale** and **rotation** can handle a wide range of *global* transformations



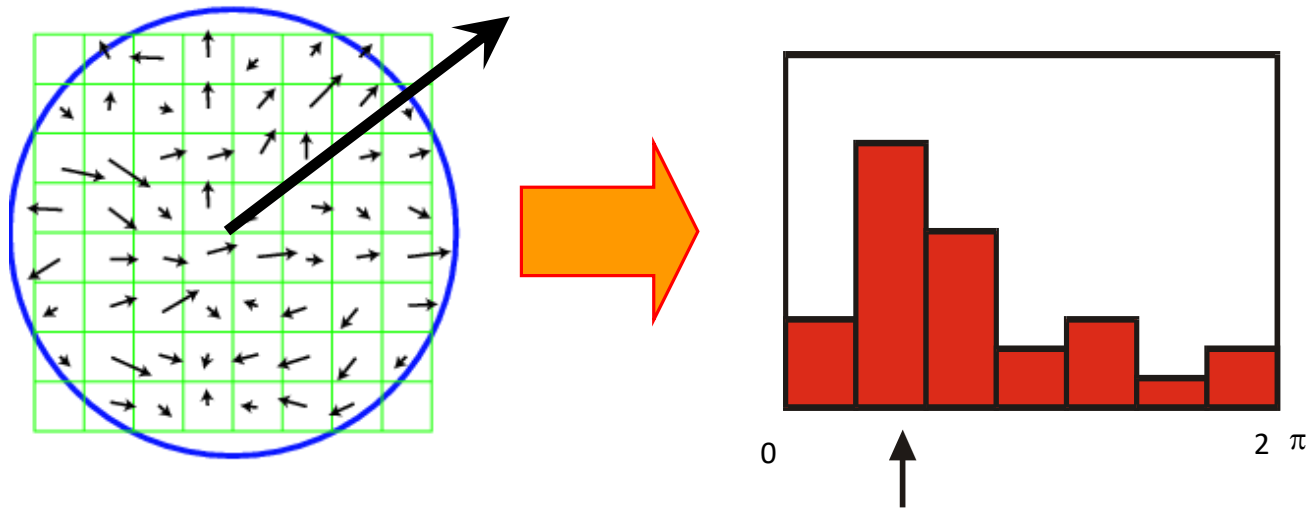
SIFT for matching

- SIFT detector returns a characteristic scale that can be normalized out, but no characteristic orientation



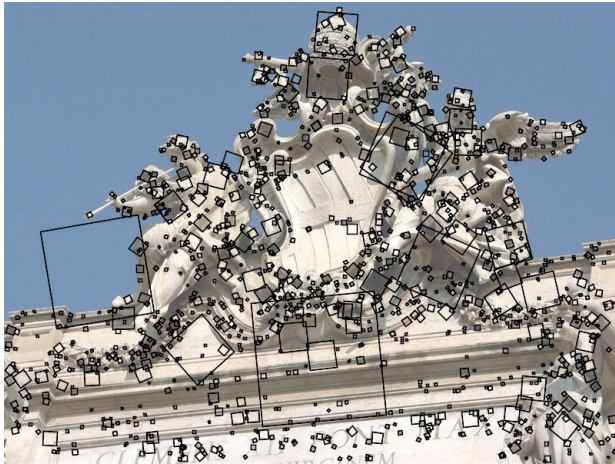
SIFT for matching

- SIFT detector returns a characteristic scale that can be normalized out, but no characteristic orientation
- Hack for finding a reference orientation:
 - Create histogram of local gradient directions in the patch
 - Assign reference orientation at peak of smoothed histogram



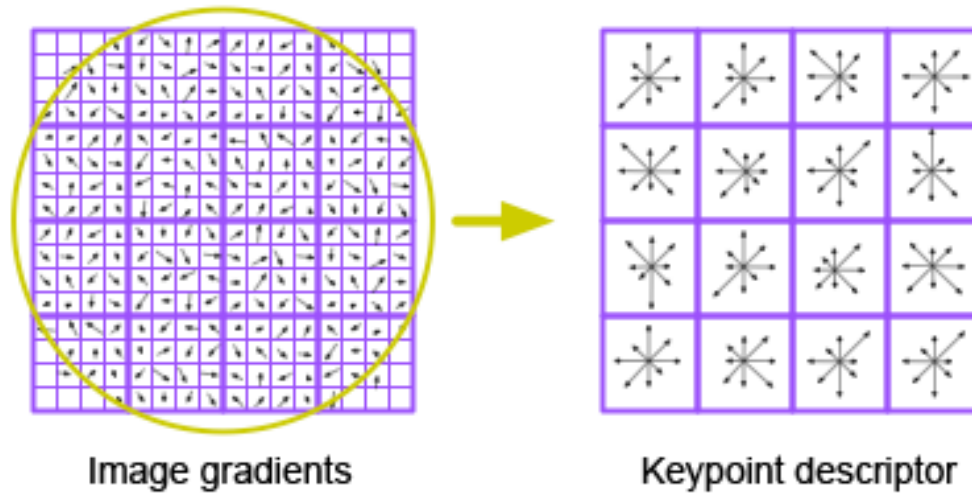
SIFT detector: Example outputs

- Detected keypoints with characteristic scales and orientations:



SIFT descriptors

- Inspiration: complex neurons in the primary visual cortex

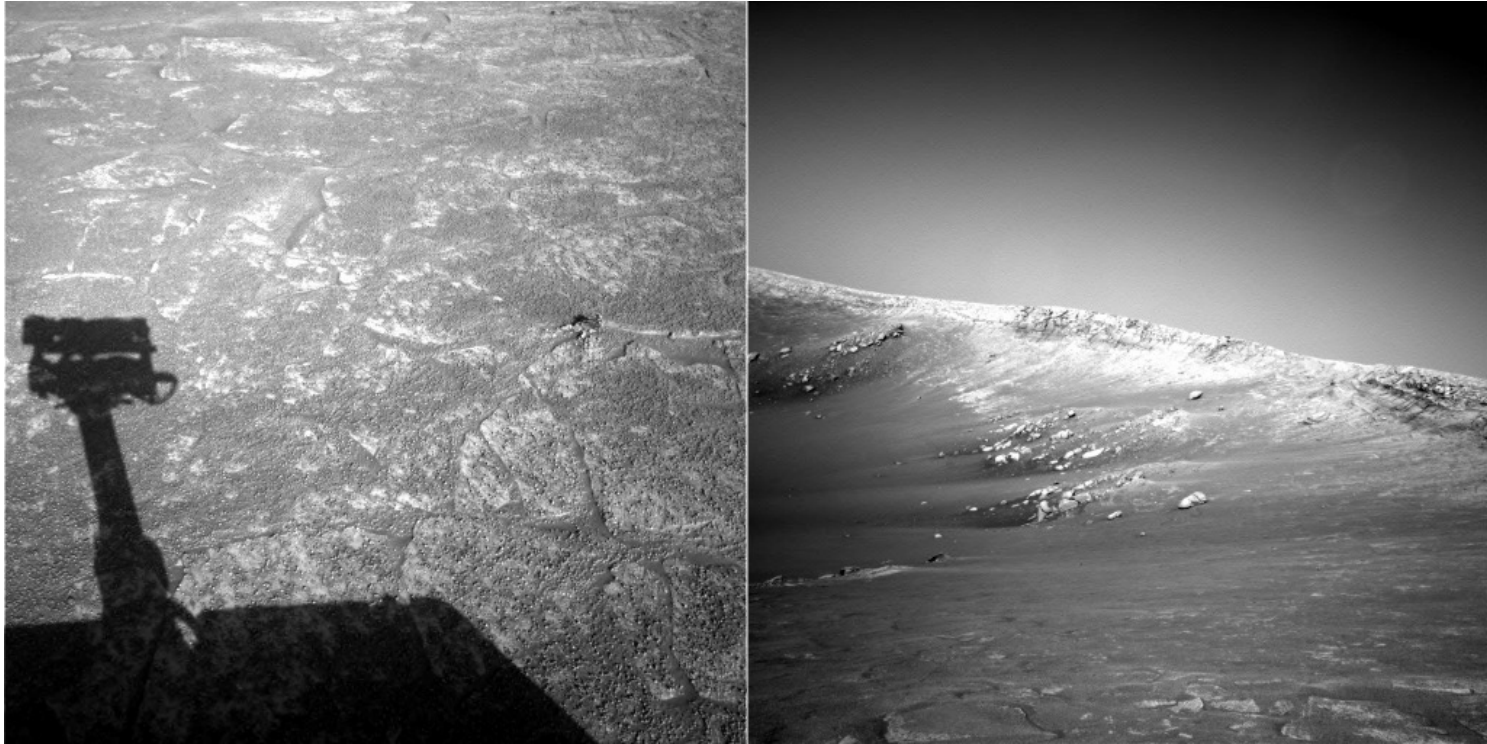


SIFT for matching

- Extraordinarily robust detection and description technique
 - Can handle changes in viewpoint
 - Up to about 60 degree out-of-plane rotation
 - Can handle significant changes in illumination
 - Sometimes even day vs. night
 - Fast and efficient—can run in real time
 - Lots of code available

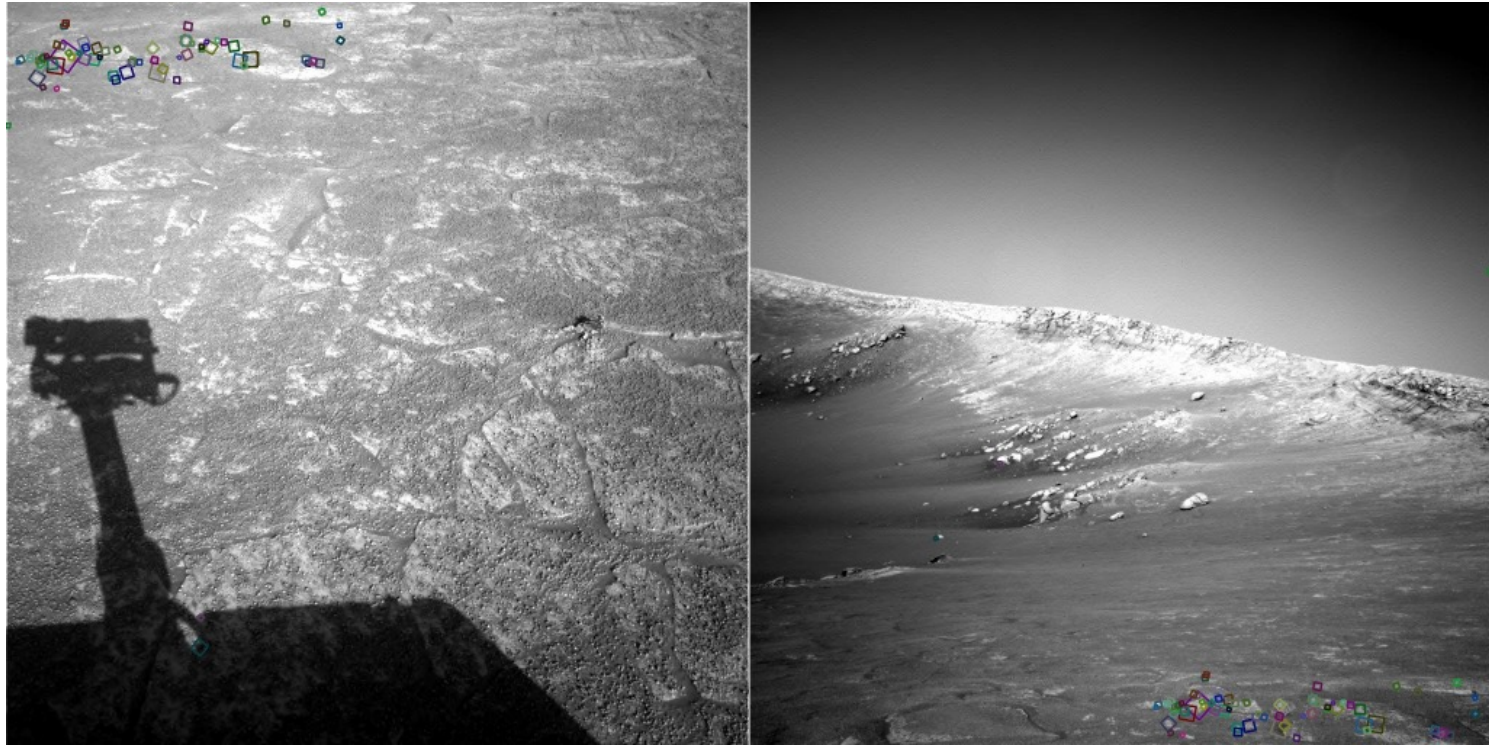


A hard matching problem



NASA Mars Rover images

Answer below (look for tiny colored squares...)



NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Local invariance beyond SIFT

- Can we make local descriptors invariant w.r.t. viewpoint changes?

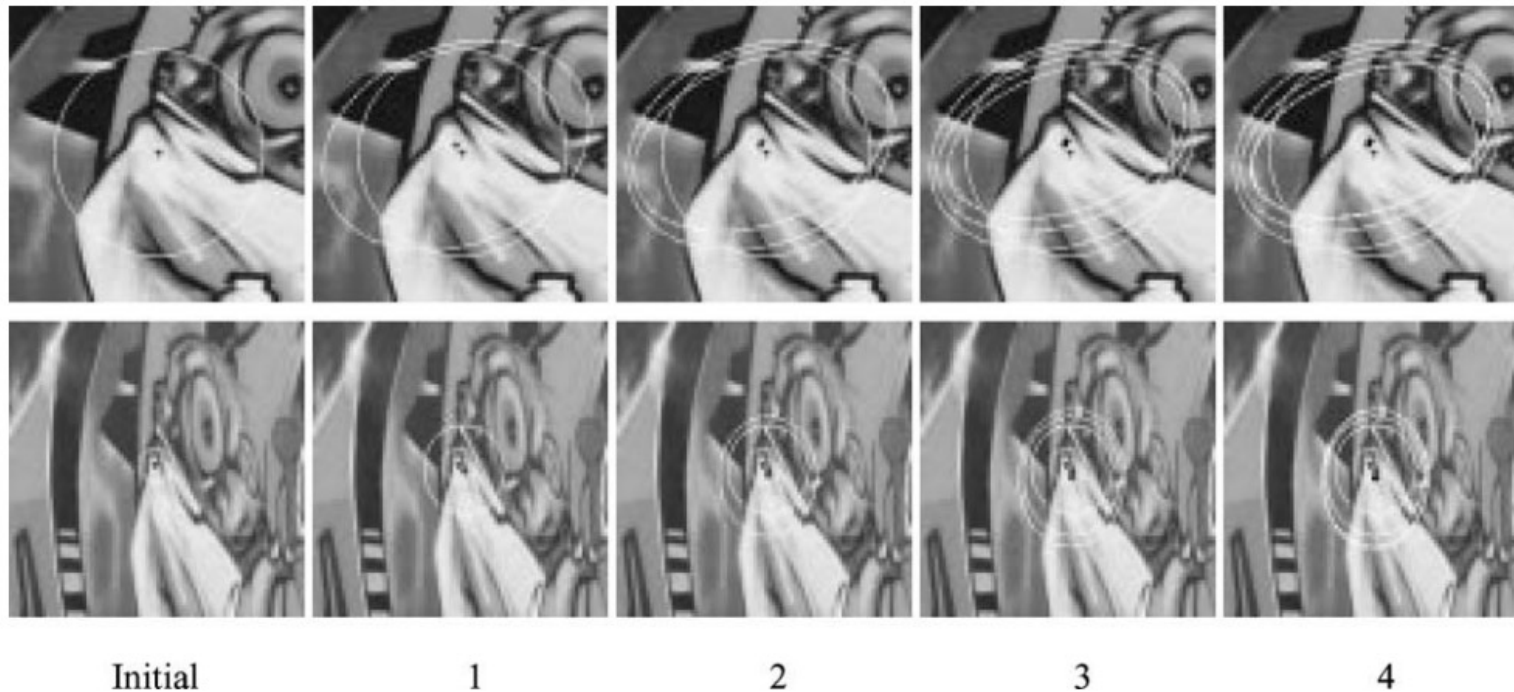


Local invariance beyond SIFT

- Can we make local descriptors invariant w.r.t. viewpoint changes?
- *Affine transformations* approximate viewpoint changes for roughly planar objects and roughly orthographic cameras



Affine adaptation



- More cumbersome in practice and not as successful as SIFT