Scale Invariant Feature Transform (SIFT)

Outline

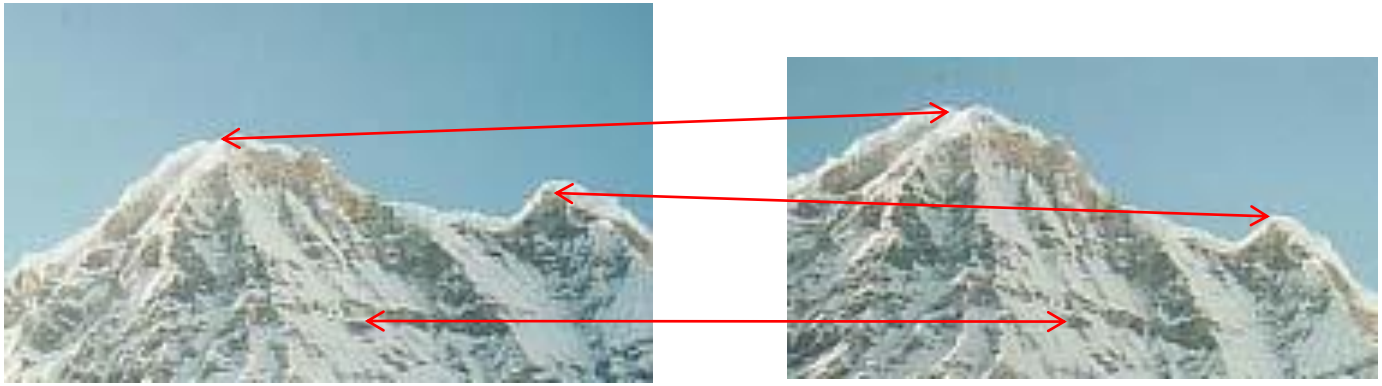
- SIFT Background
- SIFT Extraction
- Application in Content Based Image Search
- Conclusion

SIFT Background

- **Scale-invariant feature transform**
  - **SIFT:** to detect and describe local features in an images.
  - Proposed by *David Lowe* in ICCV1999.
  - Refined in IJCV 2004.
  - Cited more than 65953 times till now(March 2022).
  - Wildly used in image search, object recognition, video tracking, gesture recognition, *etc*.

Why SIFT is so popular?

An instance of object matching



Why SIFT is so popular?

- Desired property of SIFT
  - Invariant to scale change
  - Invariant to rotation change
  - Invariant to illumination change
  - Robust to addition of noise
  - Robust to substantial range of affine transformation
  - Robust to 3D view point
  - Highly distinctive for discrimination

Claimed Advantages of SIFT

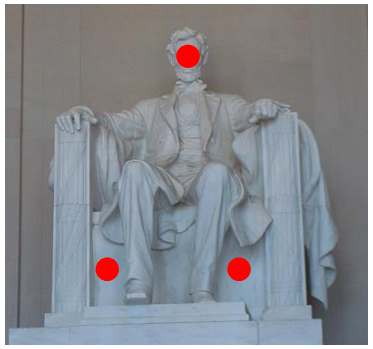
- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance

Major stages of SIFT computation

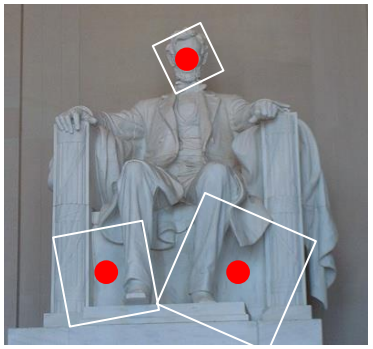
- Scale-space extrema detection
- Keypoint localization
- Orientation assignment
- Keypoint descriptor

How to extract SIFT

Test image

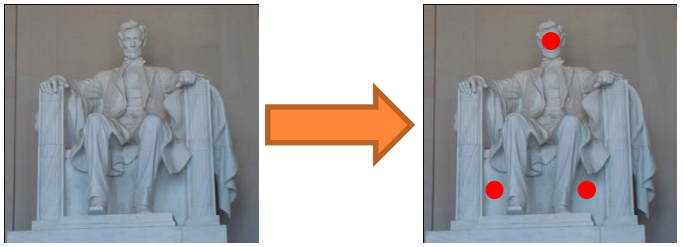


Detector: where are the local features?

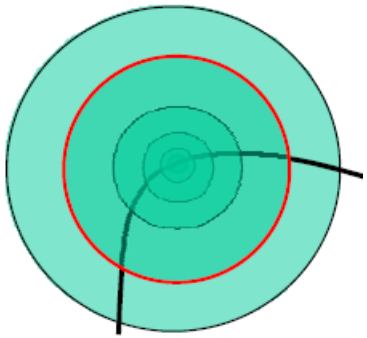


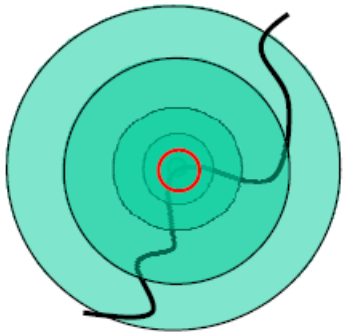
Descriptor: how to describe them?

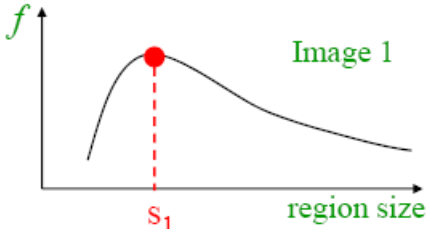
SIFT Detector

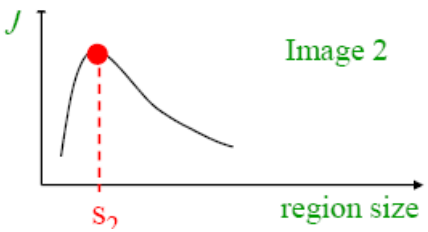


- Desired properties for detector
  - **Position**: Repeatable across different changes
  - **Scale**: automatic scale estimation
- **Intuition:** Find scale that gives local maxima of some function *f* in both position and scale.



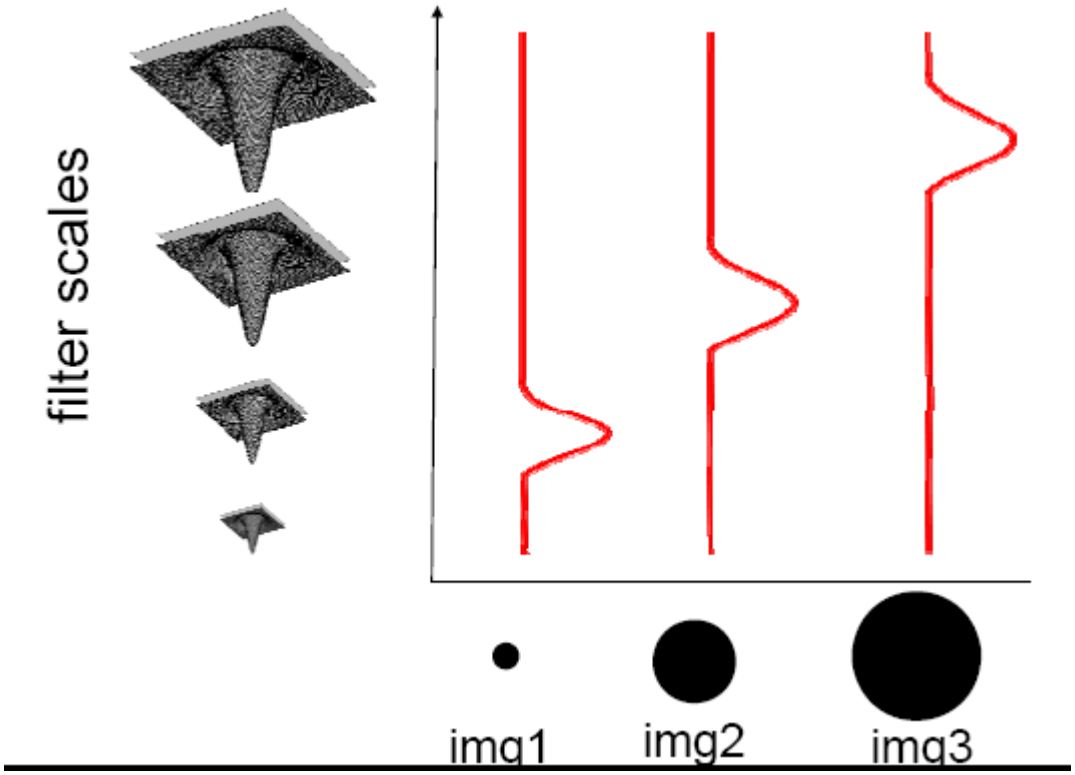






What can be the "signature" function *F*?

Laplacian-of-Gaussian = "***blob***" detector



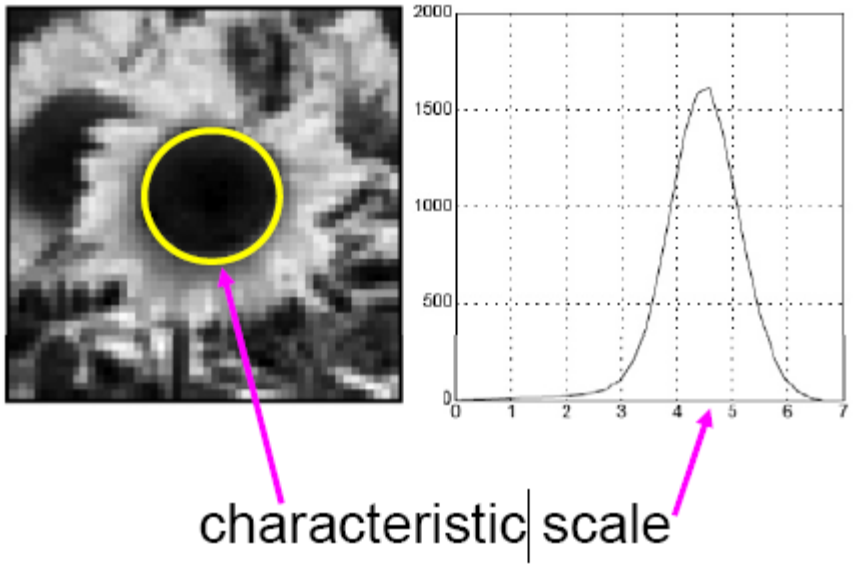
$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

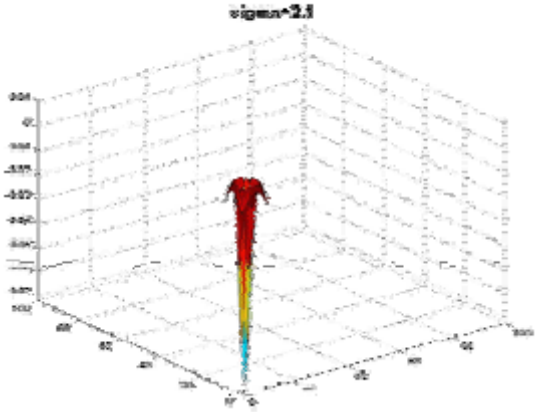
$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -16 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

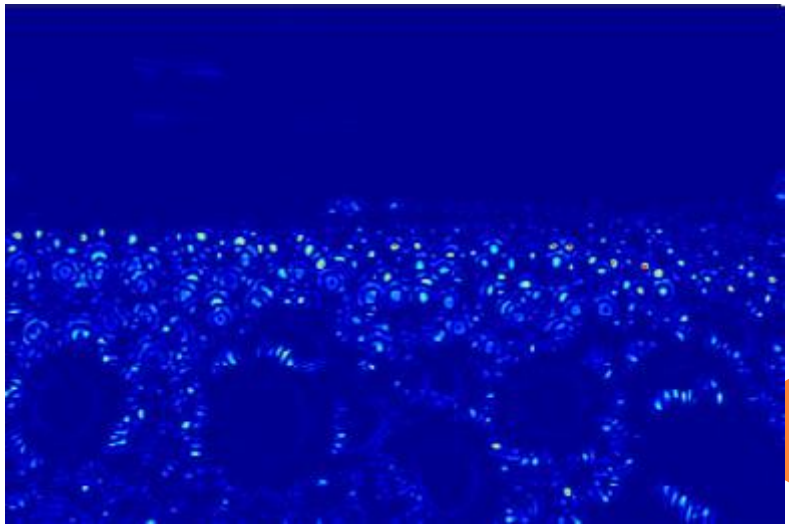
2-D LoG
by a 5 × 5 convolution kernel

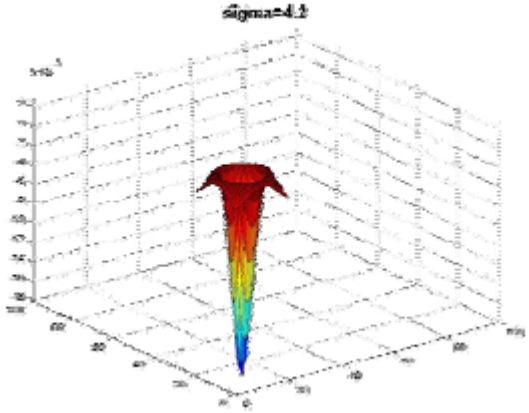
At a given point in the image:

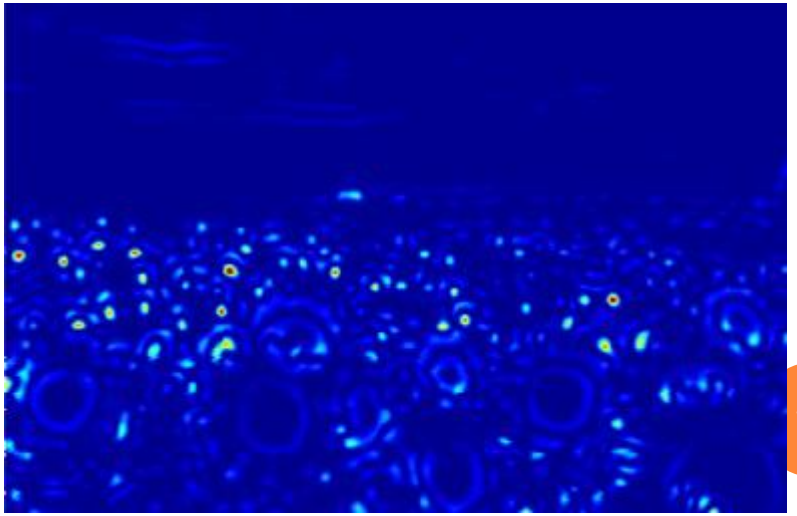
- We define the *characteristic scale* as the scale that produces peak of Laplacian response

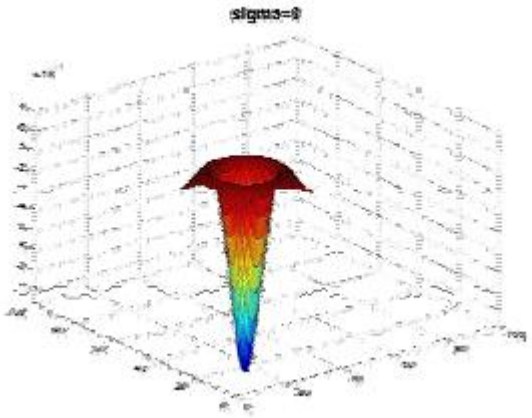


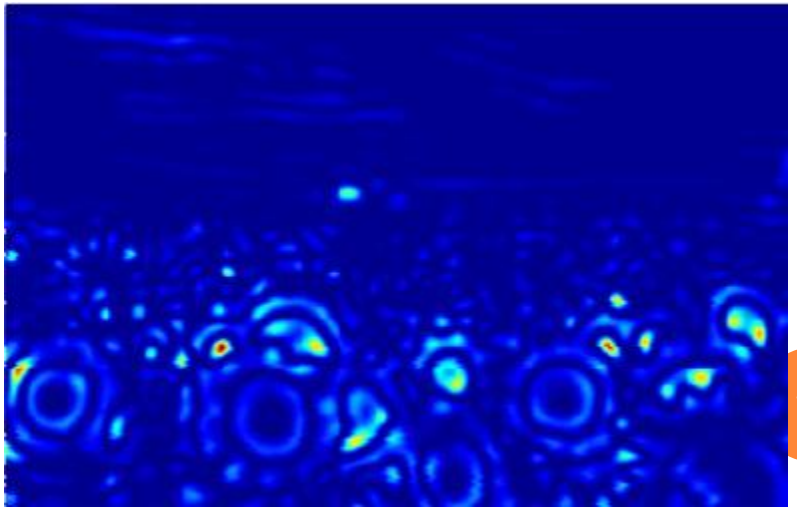

sigma=2.1

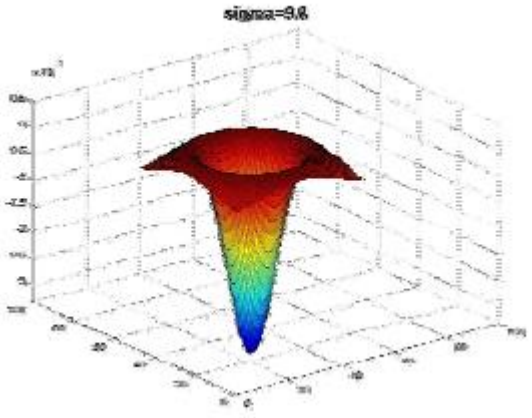


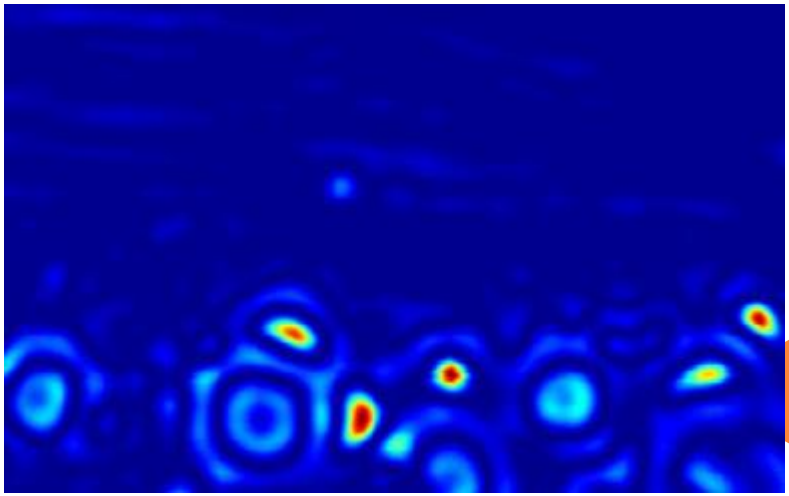

Sigma=4.1

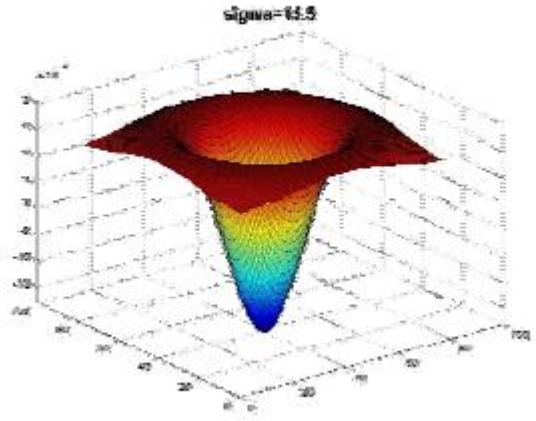


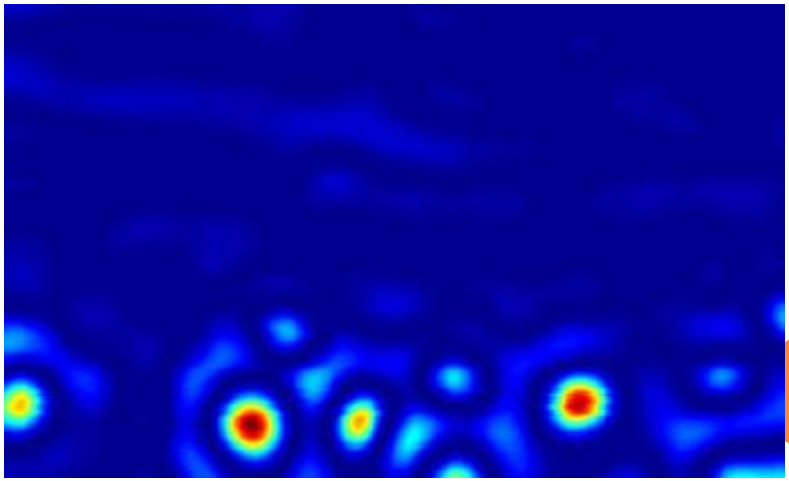

sigma=9

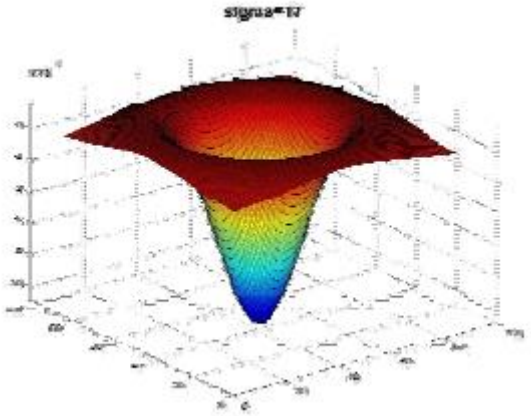


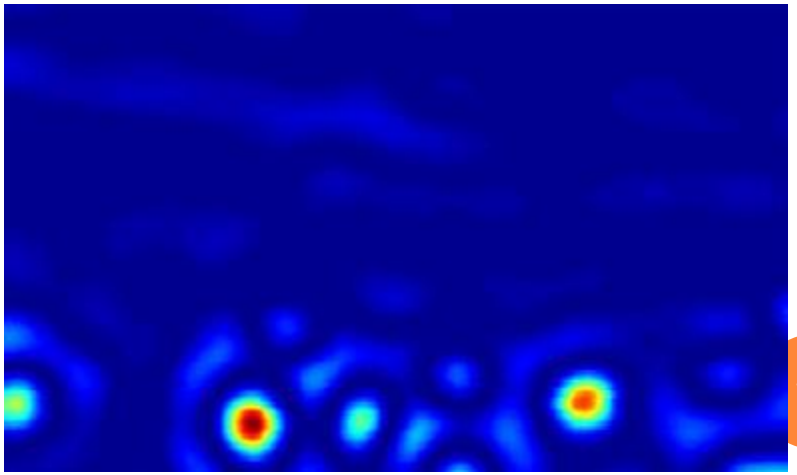

sigma=9.6




sigma=65.5







LoG vs. DoG

$$\nabla^2 G_\sigma(x,y) = \left(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G_\sigma(x,y)$$

$$\underbrace{G(x,y,k\sigma) - G(x,y,\sigma)}_{\text{DoG}} \approx (k-1)\sigma \frac{\partial G}{\partial \sigma} = (k-1)\sigma^2 \underbrace{\nabla^2 G}_{\text{LoG}}$$

Technical detail

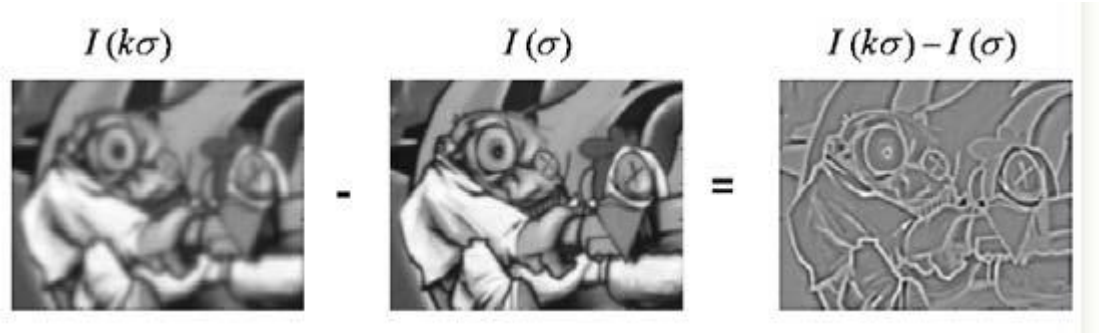
- We can approximate the Laplacian with a difference of Gaussians; more efficient to implement.

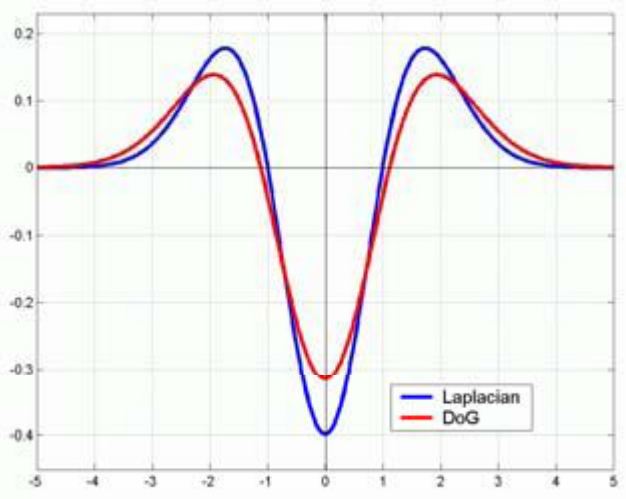
$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

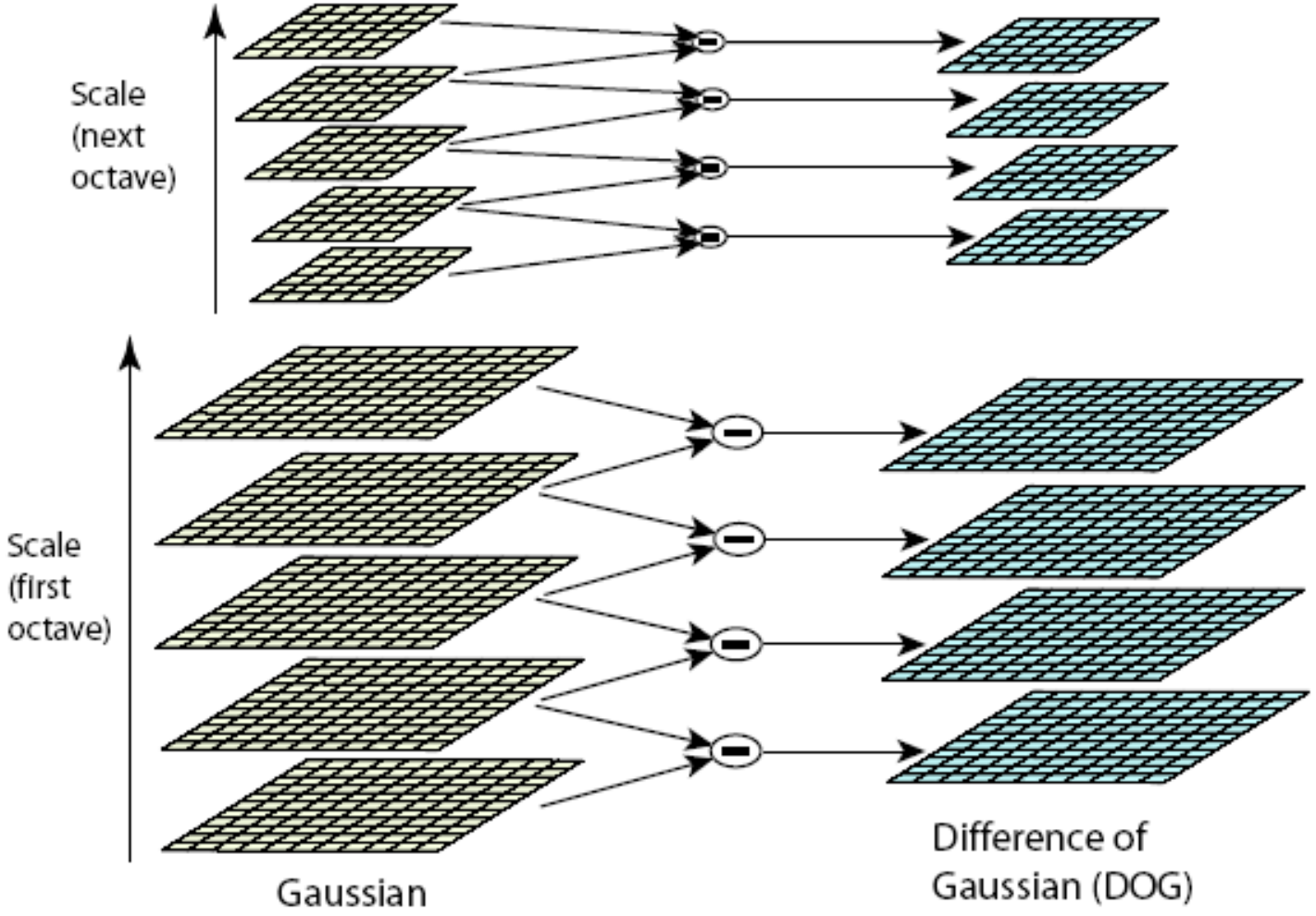
(Laplacian)

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

(Difference of Gaussians)







For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor 2, and the process repeated.

Lowe's Pyramid Scheme

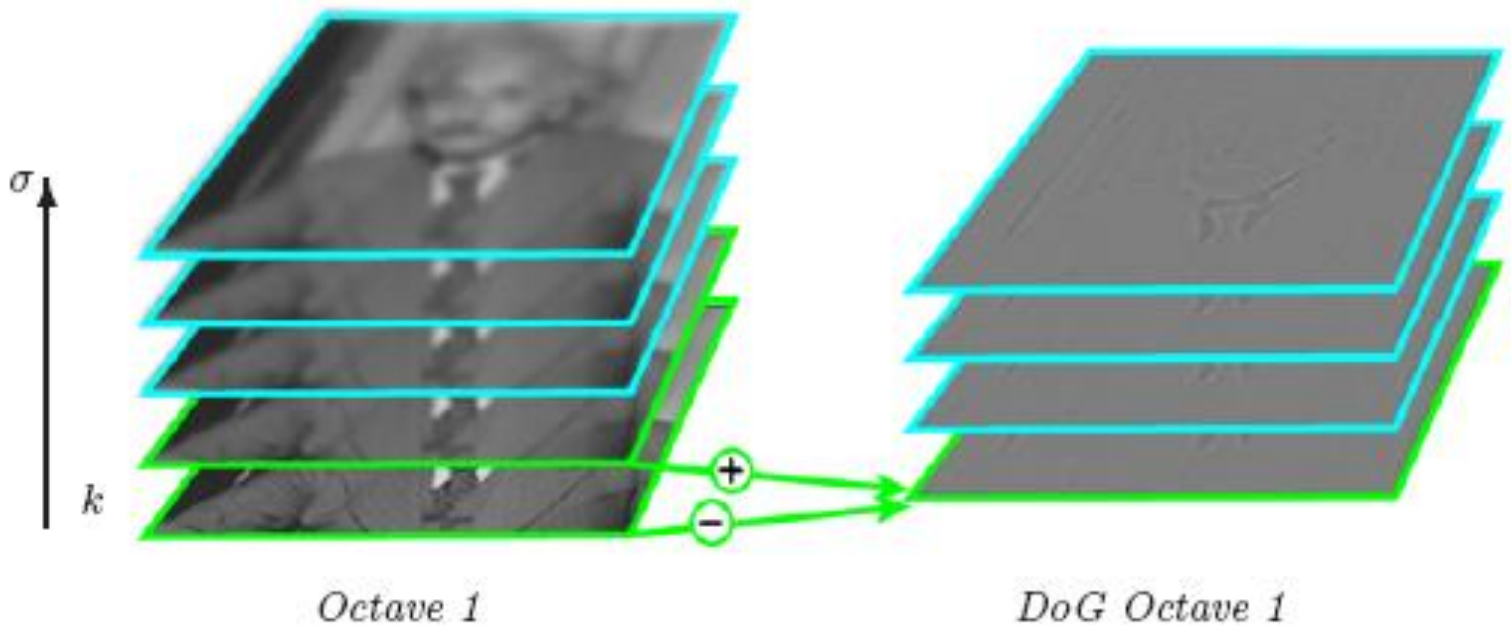
- Scale space is separated into octaves:
  - Octave 1 uses scale σ
  - Octave 2 uses scale 2σ
  - etc.

- In each octave, the initial image is repeatedly convolved with Gaussians to produce a set of scale space images.

- Adjacent Gaussians are subtracted to produce the DOG

- After each octave, the Gaussian image is down-sampled by a factor of 2 to produce an image ¼ the size to start the next level.

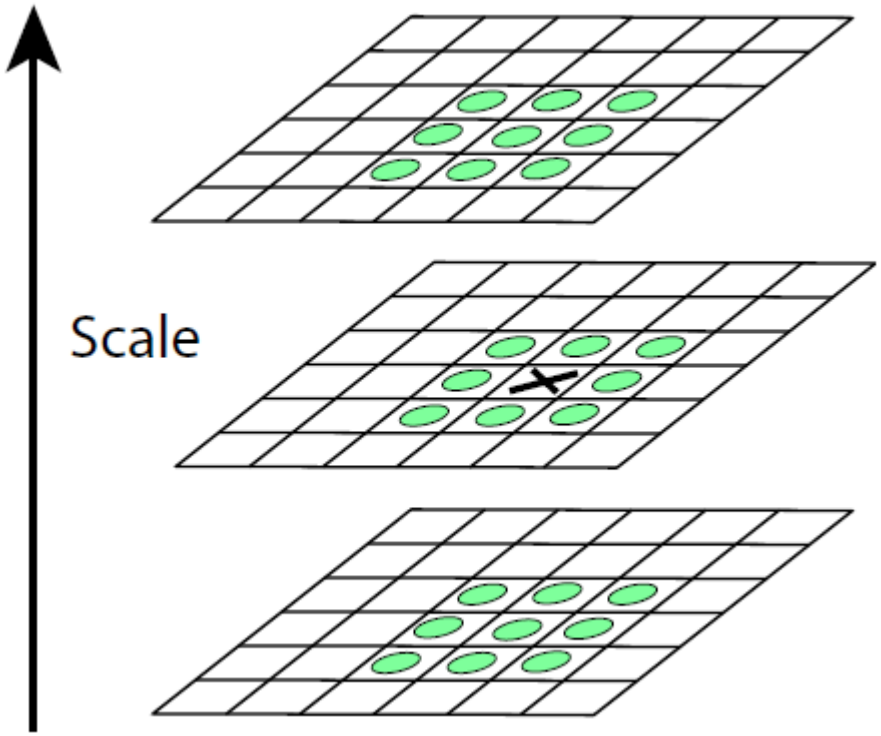
DoG Image Pyramid



$$\sigma_0 \rightarrow 2\sigma_0$$

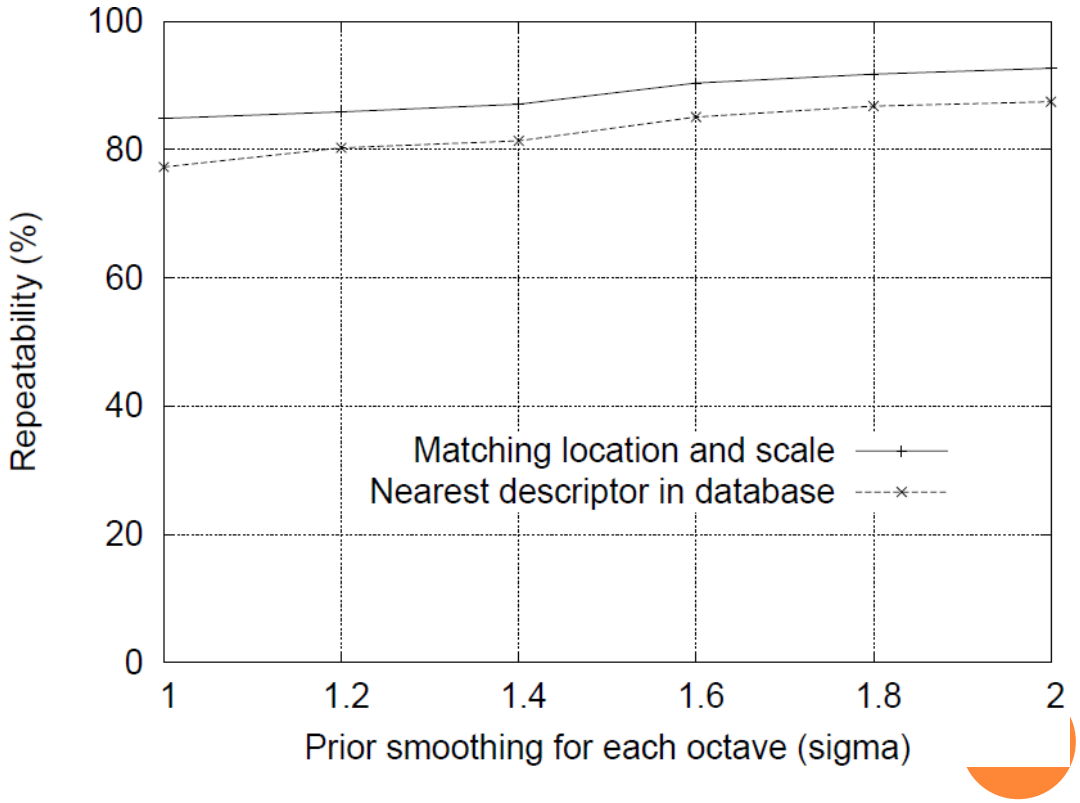
Local Extrema Detection

- Maxima and minima
- Compare x with its 26 neighbors at 3 scales



Frequency of Sampling in Domain

- Trade-off between sampling frequency and rate of detection
- Sigma=1.6



Eliminating Edge Responses

- Motivation
  - DoG aims to detect “blob”.
  - DoG function have a strong response along edges.
  - Remove such key points by Hessian Matrix analysis
- Hessian matrix
  - Formulation

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Eliminating Edge Responses

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

α: *larger eigenvalue*
β *: smaller eigenvalue*

$$\alpha = r\beta$$

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}.$$

$r = 10$ *for this experimants*

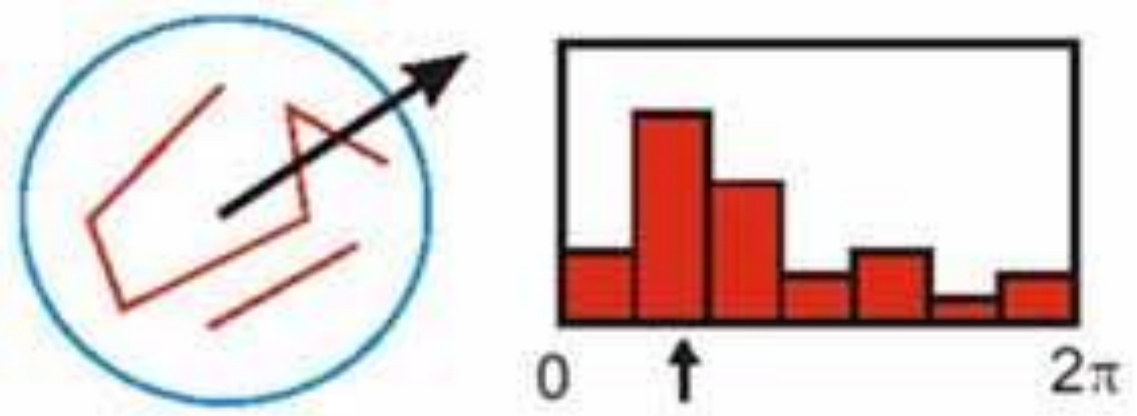
Orientation

Gradient and angle:

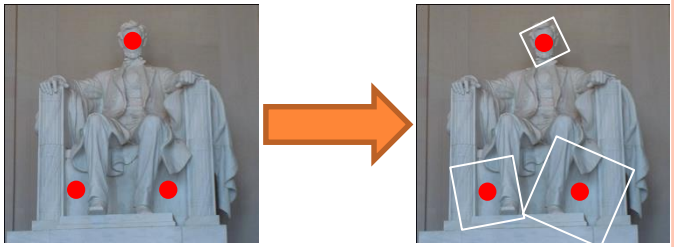
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = a\tan 2((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

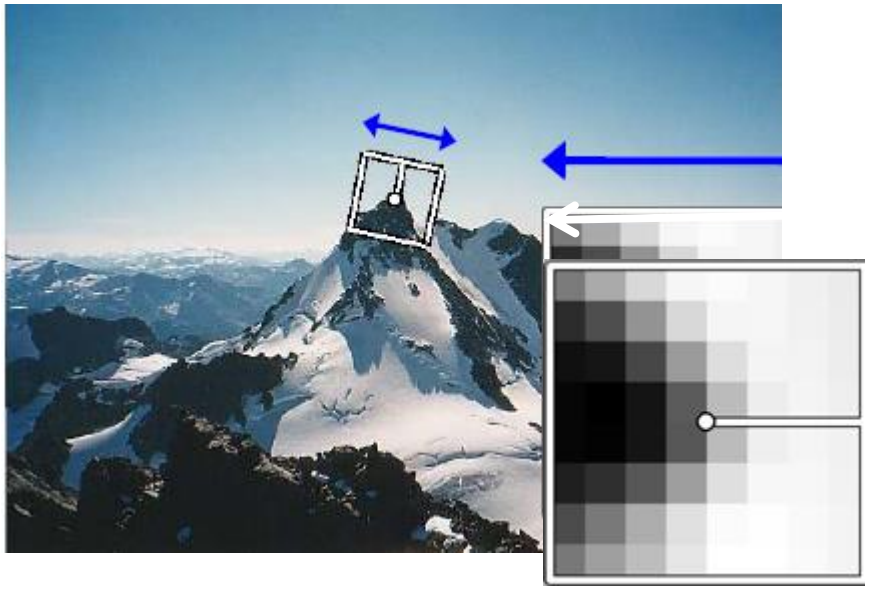
Orientation selection



SIFT Descriptor

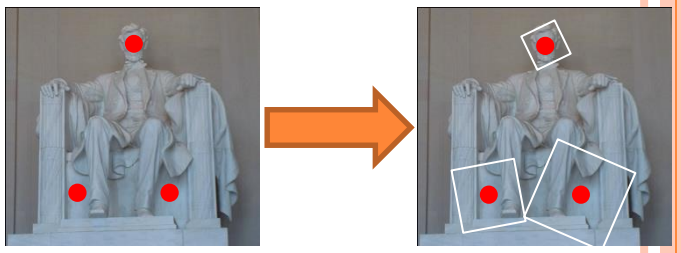


- Making descriptor rotation invariant

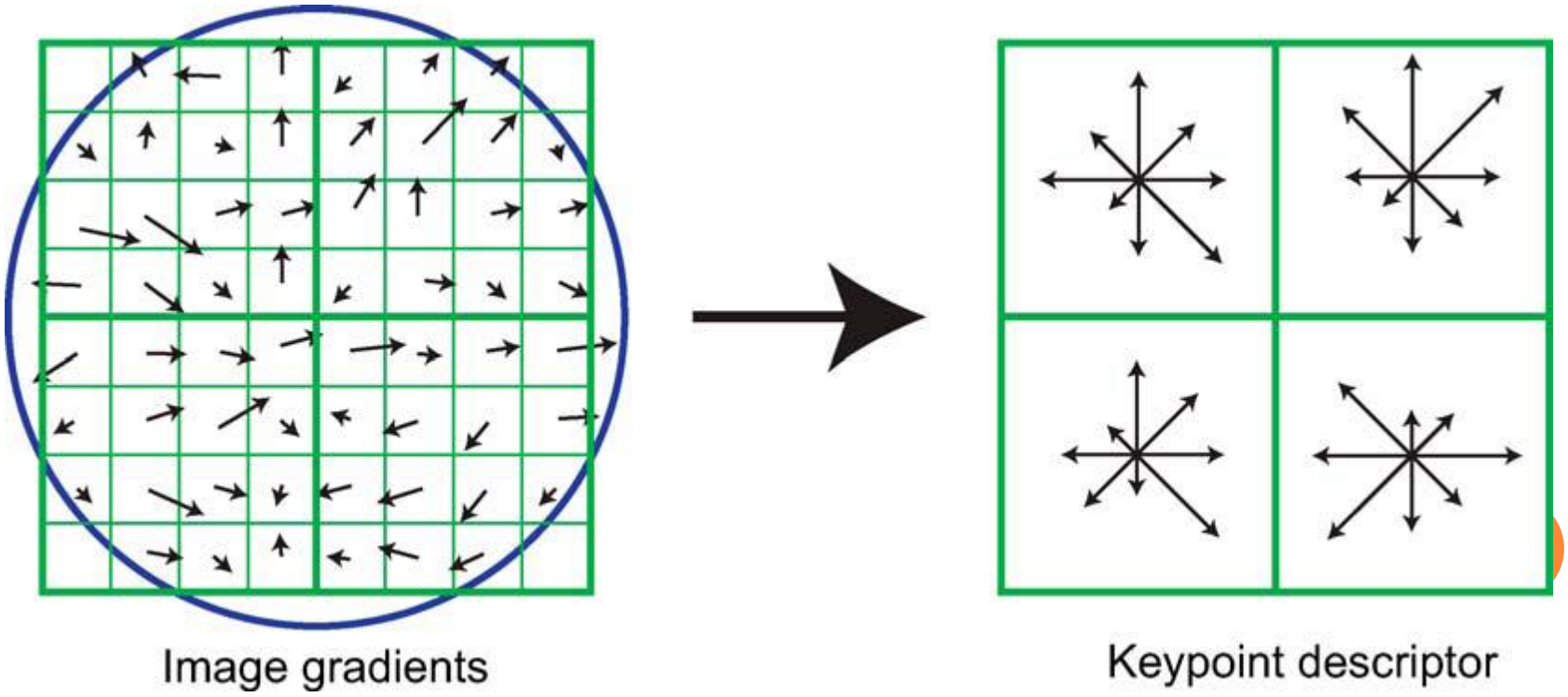


- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation.

SIFT Descriptor



- Use histograms to bin pixels within sub-patches according to their orientation.



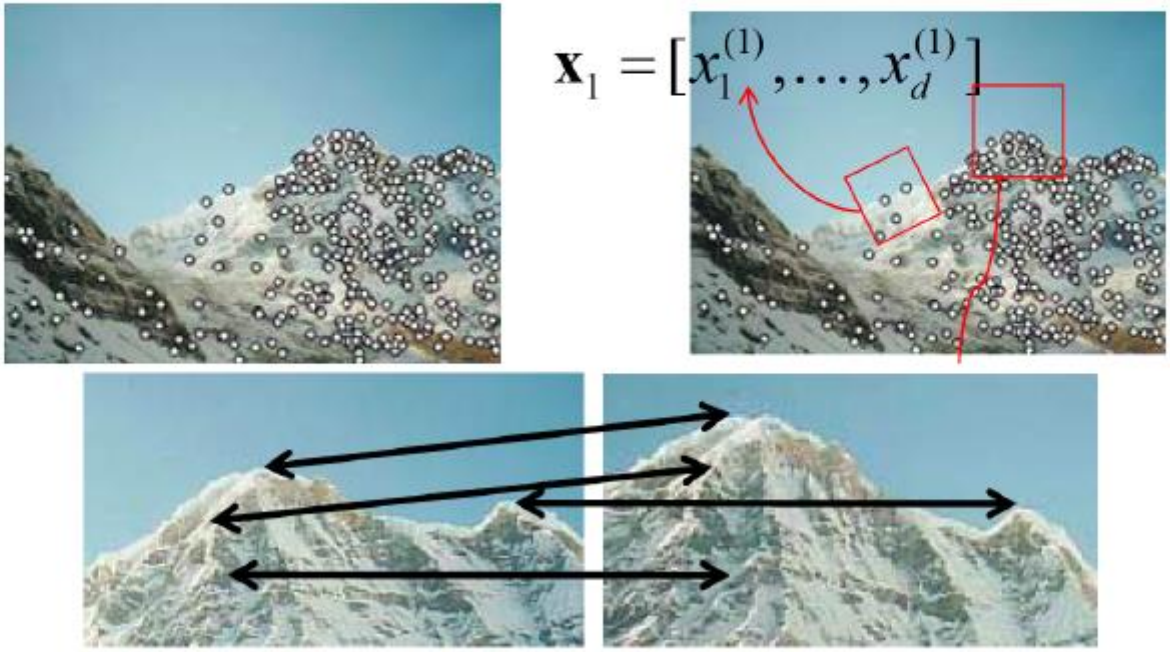
Summary of SIFT Feature

- Descriptor: 128-D
  - 4 by 4 patches, each with 8-D gradient angle histogram:

    $4 \times 4 \times 8 = 128$
  - Normalized to reduce the effects of illumination change.
- Position: (x, y)
  - Where the feature is located at.
- Scale
  - Control the region size for descriptor extraction.
- Orientation
  - To achieve rotation-invariant descriptor.

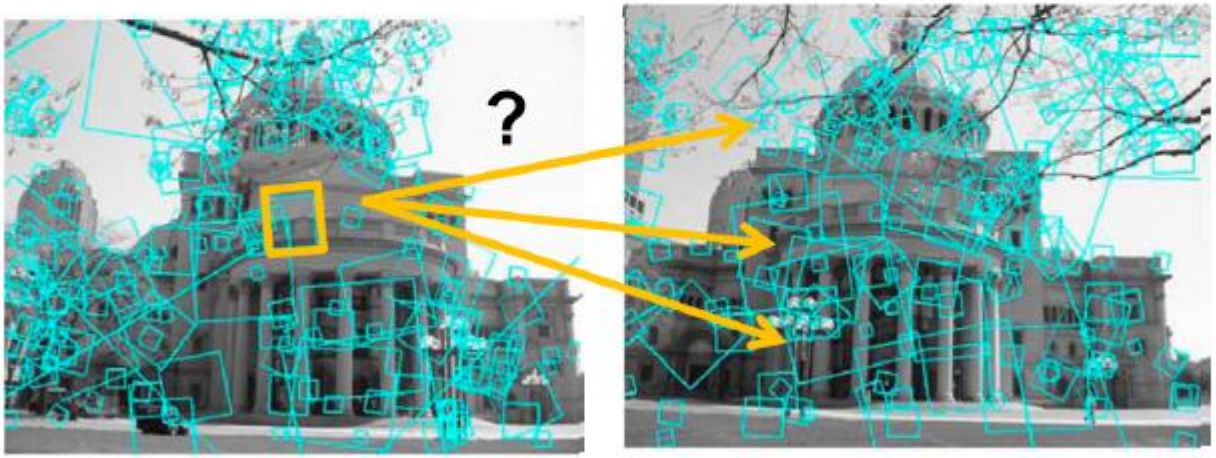
Local Features

- **Detection**: Identify the interest points.
- **Description**: Extract vector feature descriptor around each interest point.
- **Matching**: Determine correspondence between descriptors in two views.



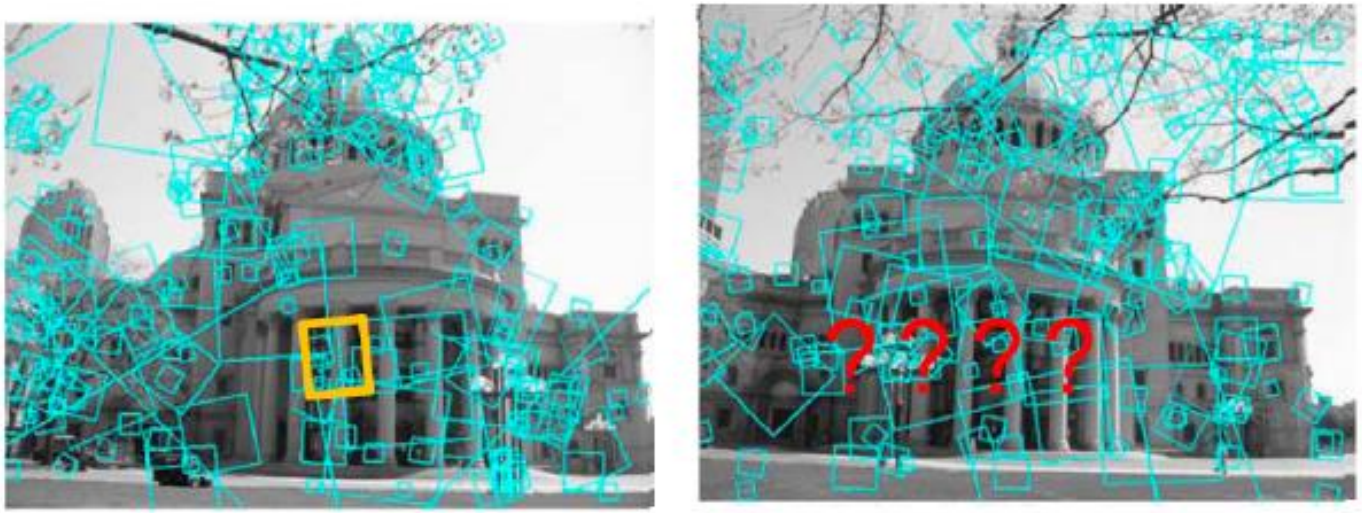
Matching Local Features

- To **generate candidate matches**, find patches that have the most similar appearance (e.g., lowest SSD)
- Simplest approach: **compare them all**, take the closest (or closest k, or within a thresholded distance)



Ambiguous matches

- At what SSD value do we have a good match?
- To add robustness, consider ratio of distance to best match to distance to second best match
  - If low, first match looks good.
  - If high, could be ambiguous match.



Closest / NextClosest< threshold(0.6 for example)