SCALE INVARIANT FEATURE TRANSFORM (SIFT)

Lowe, D.G., 2004. Distinctive image features from scale-invariant key points. *International journal of computer vision, 60(2),* pp.91-110.

Distinctive Image Features from Scale-Invariant Keypoints

David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

January 5, 2004

Abstract

This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition can robustly identify objects among clutter and occlusion while achieving near real-time performance.

Accepted for publication in the *International Journal of Computer Vision*, 2004.

Why local features ?

Locality

- features are local, so robust to occlusion and clutter

Distinctiveness

- can differentiate a large database of objects

Quantity

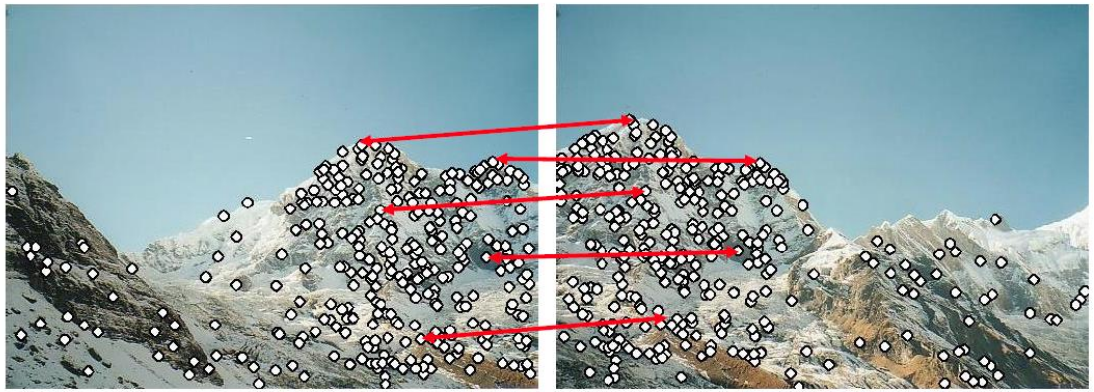
- hundreds or thousands in a single image

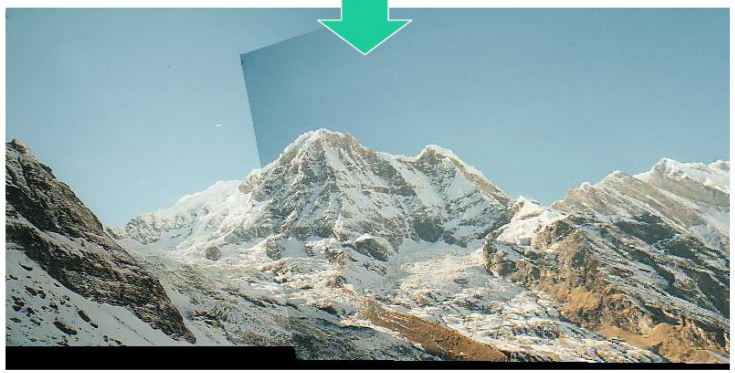
Efficiency

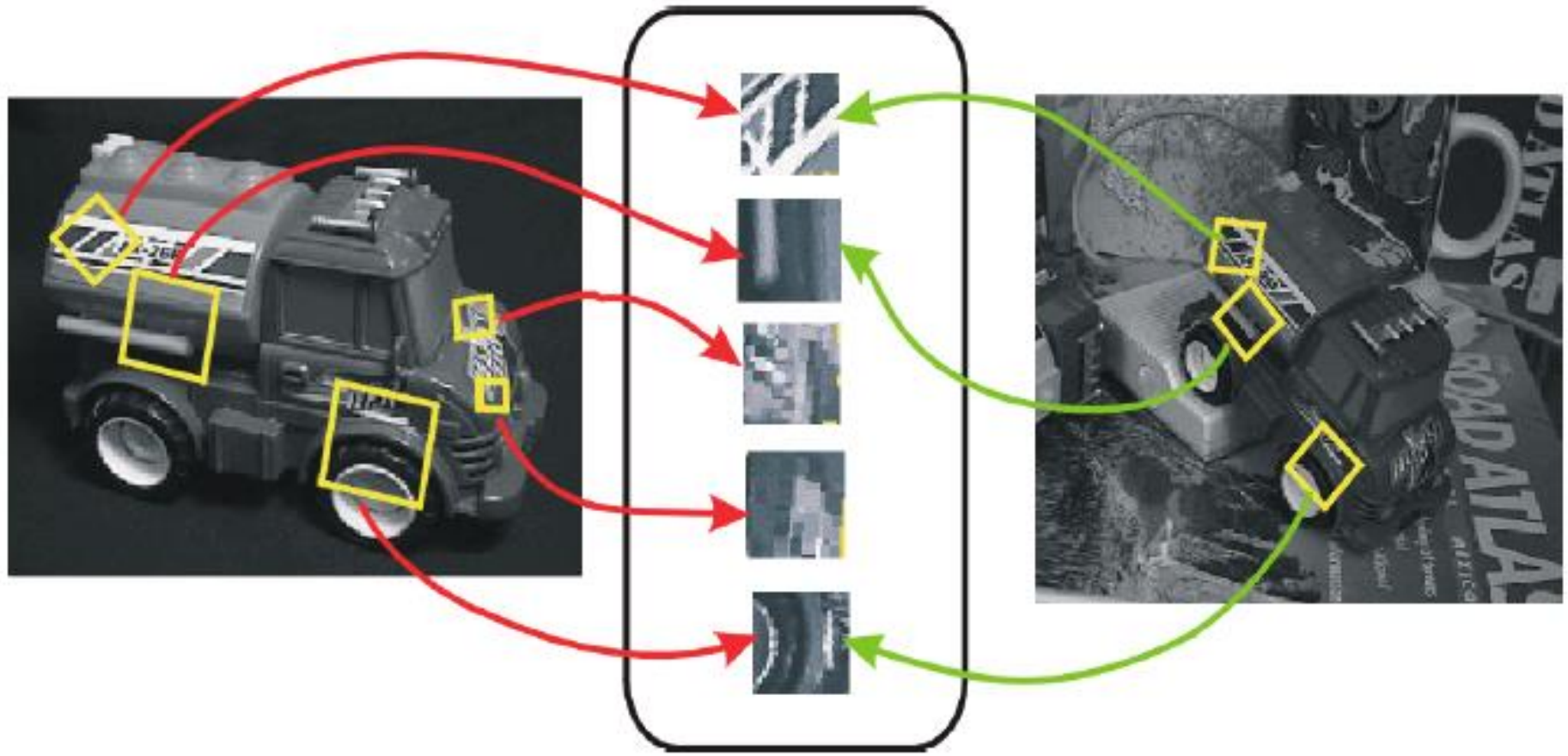
- real-time performance achievable

Generality

- exploit different types of features in different situations







Feature Descriptors

What about edges?

Edges can be invariant to brightness changes but typically not invariant to other transformations.

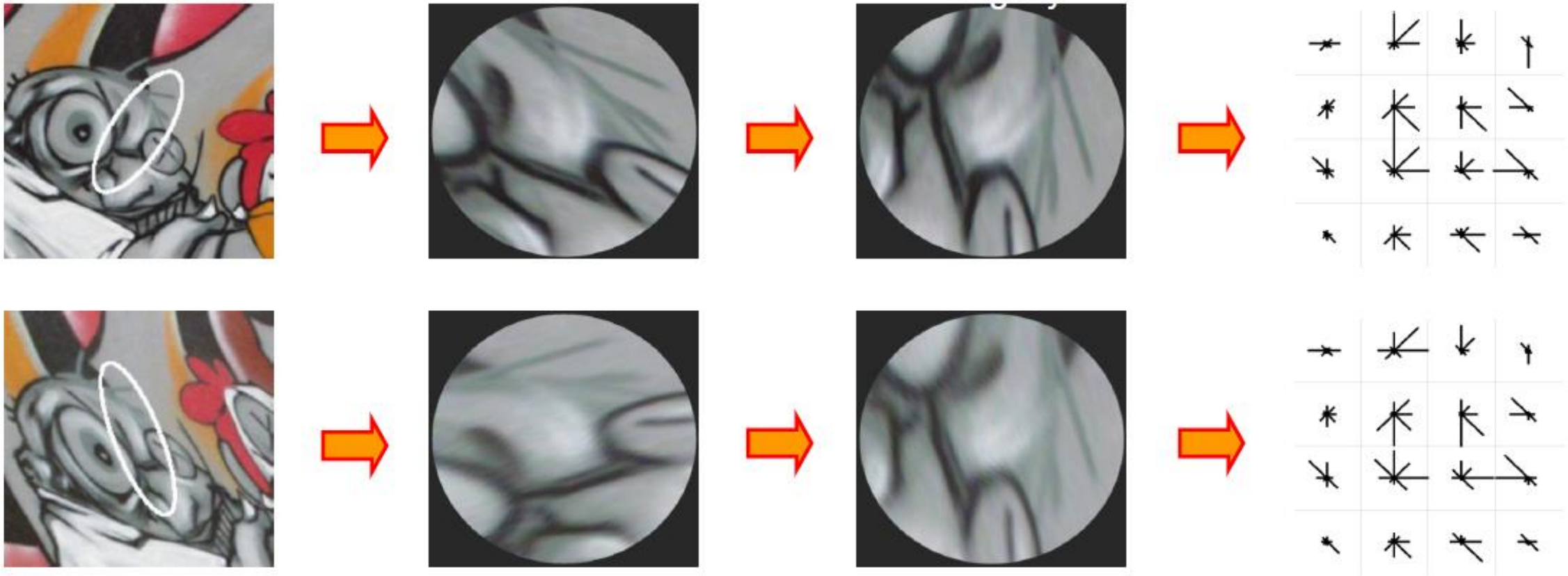
A good local feature should be tolerant to

- Image noise
- Changes in illumination
- Uniform scaling
- Rotation
- Minor changes in viewing direction

What is SIFT?

It is a technique for **detecting** salient, stable feature points in an image.

For every such point, it also provides a set of **“features”**that “characterize/describe” a small image region around the point. These features are invariant to rotation and scale.



Detection stages for SIFT features:

- Scale-space extrema detection

  This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".

- Keypoint localization

  These are maxima and minima in the Difference of Gaussian image we calculate in step 1

- Orientation assignment

  An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.

- Generation of keypoint descriptors.

  Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features.

https://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

- Step 1: Scale-space extrema detection
  - Constructing a scale space
  - Generate progressively blurred out images using Gaussian filter for different values of 'σ' with different image sizes

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

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

L - Blurred image
G - Gaussian Blur operator
I – Input Image


Scale = 0

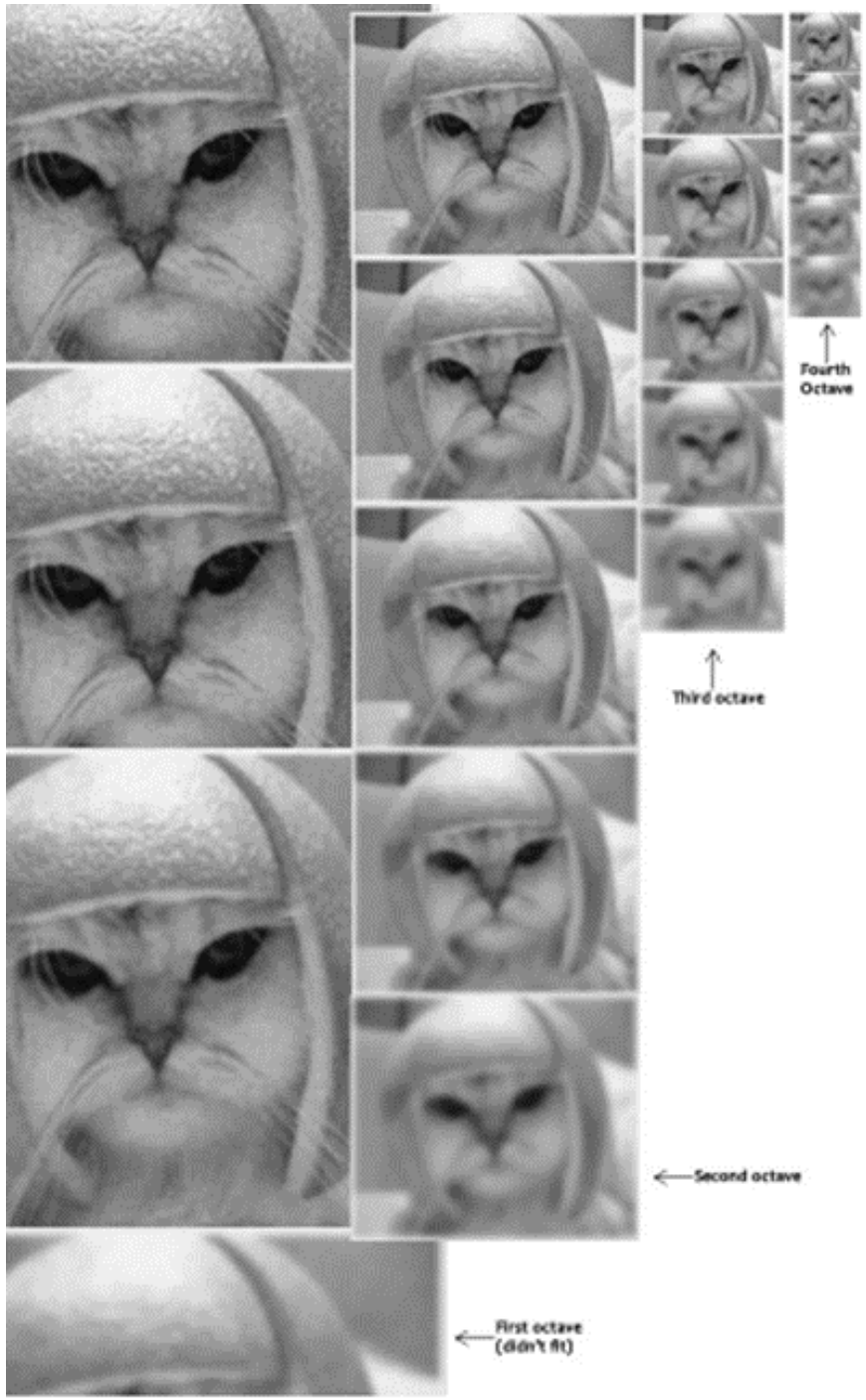

Scale = 1


Scale = 4


Scale = 16


Scale = 64


Scale = 256



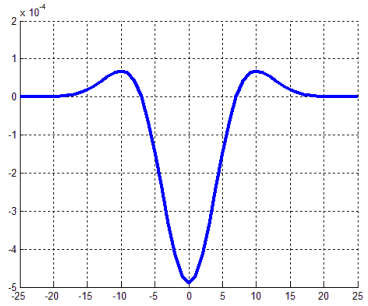
- Each octave's image size is half the previous one
- σ differs by a factor of ‘k’ (preferably k = $\sqrt{2}$) with adjacent scales

octave \ scale →				
0.707107	1.000000	1.414214	2.000000	2.828427
1.414214	2.000000	2.828427	4.000000	5.656854
2.828427	4.000000	5.656854	8.000000	11.313708
5.656854	8.000000	11.313708	16.000000	22.627417

- In the first step of SIFT, you generate several octaves of the original image.
- Each octave's image size is half the previous one. Within an octave, images are progressively blurred using the Gaussian Blur operator.

- Now we use those blurred images to generate another set of images, the Difference of Gaussians (DoG).
- These DoG images are a great for finding out interesting key points in the image.

Laplacian of Gaussian

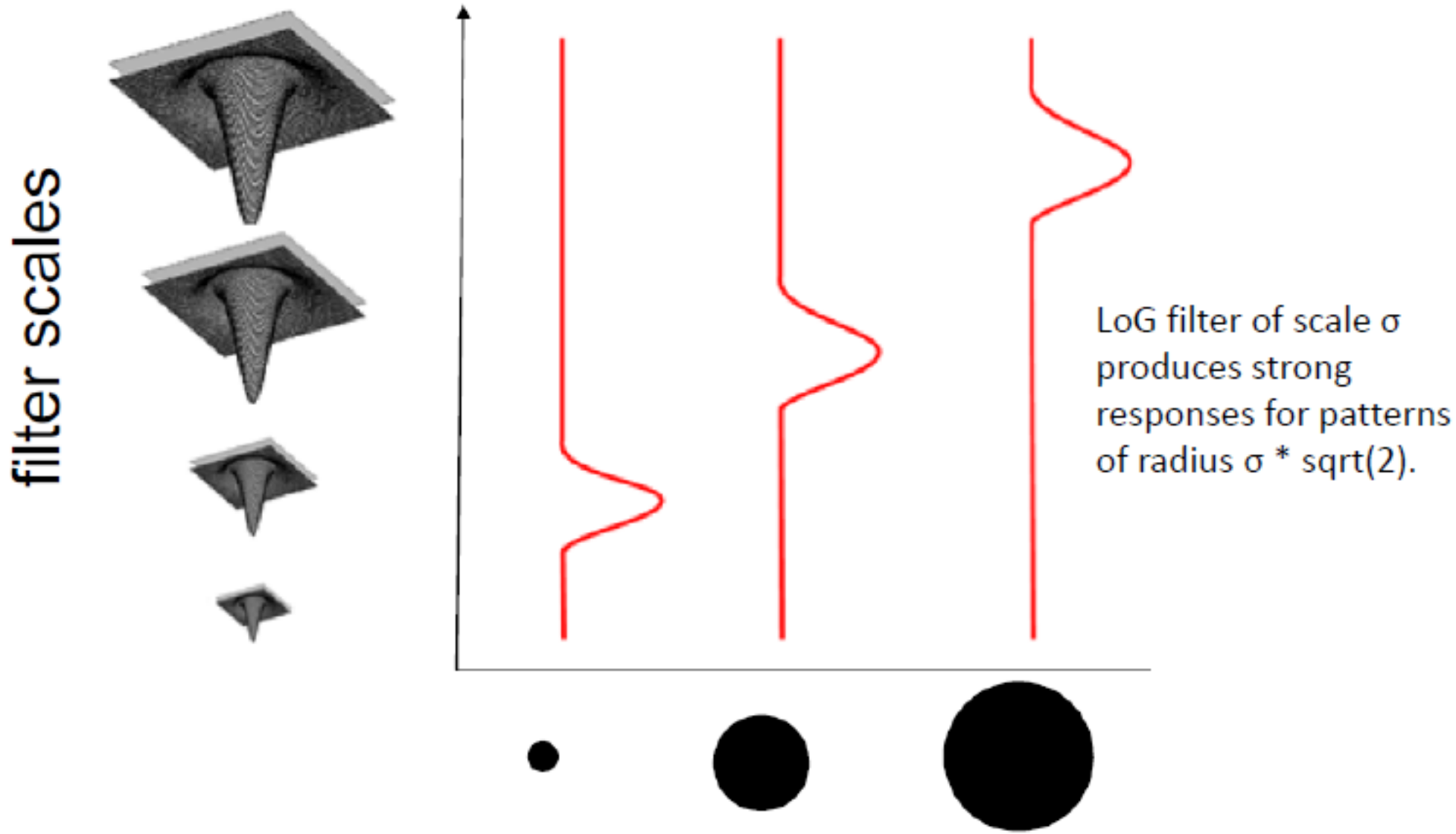


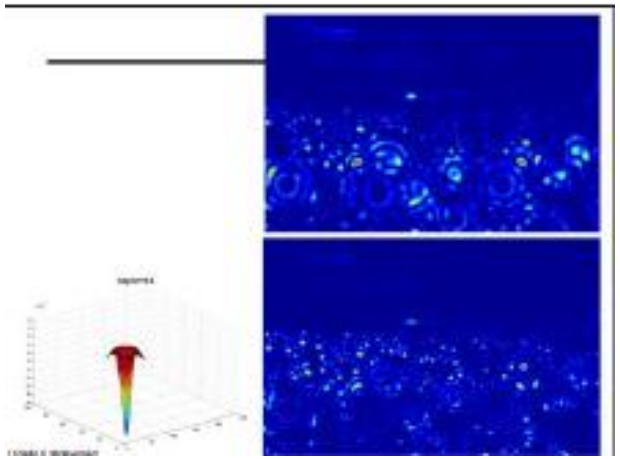
- The Laplacian of Gaussian (LoG) operation can apply to these images to extract features.
- It locates edges and corners on the image. These edges and corners are good for finding keypoints.
- But the second order derivative is extremely sensitive to noise. The blur smoothes it out the noise and stabilizes the second order derivative.
- However, calculating all those second order derivatives is computationally intensive

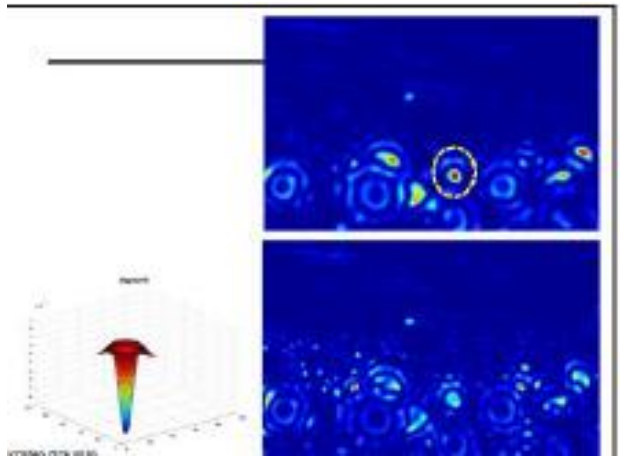
$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}}$$

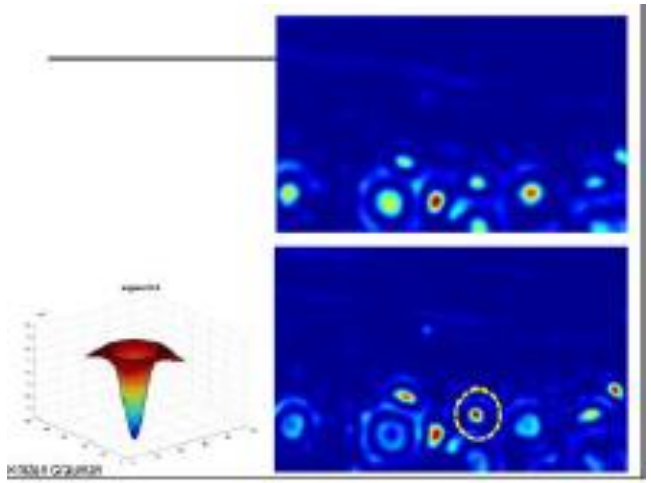
$$\nabla^2 G(x, y, \sigma) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

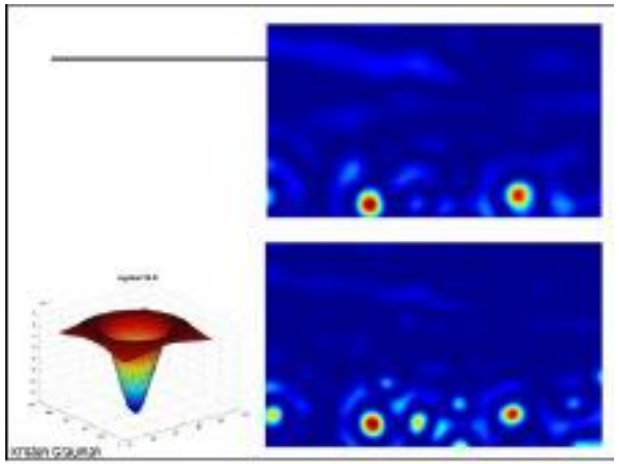
Laplacian-of-Gaussian = “blob” detector $\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$

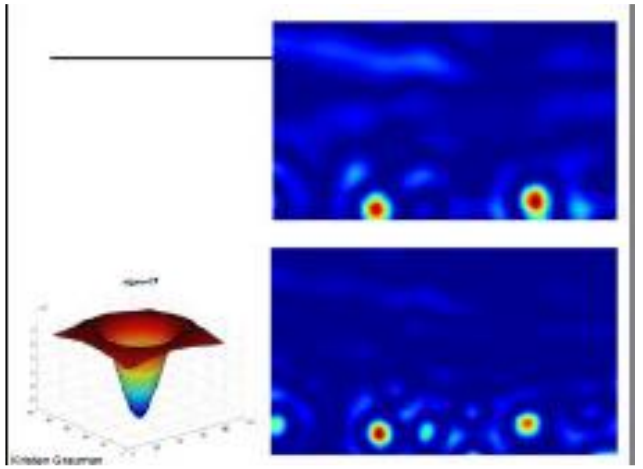


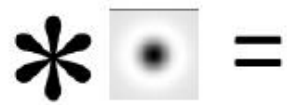


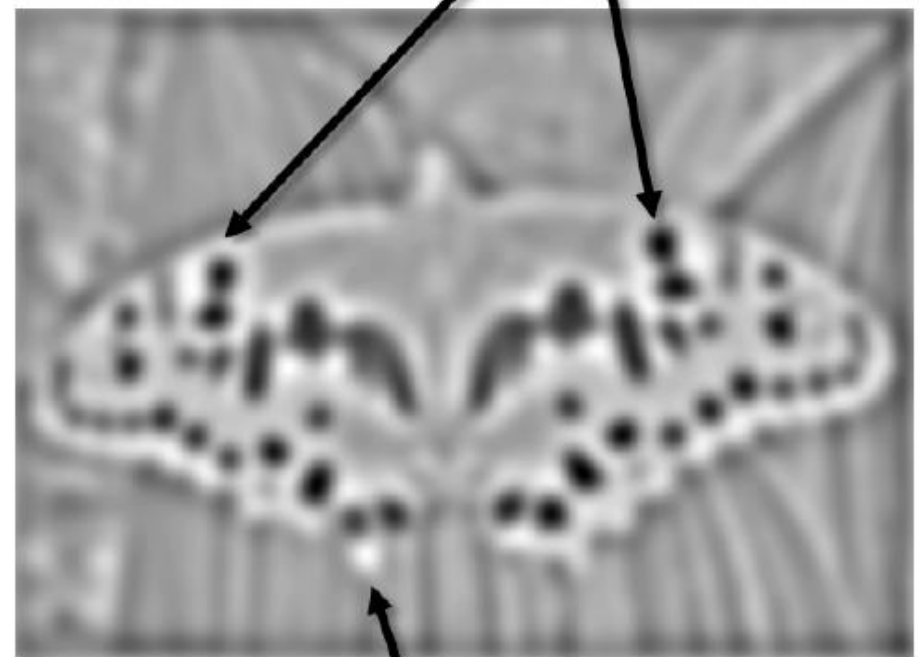






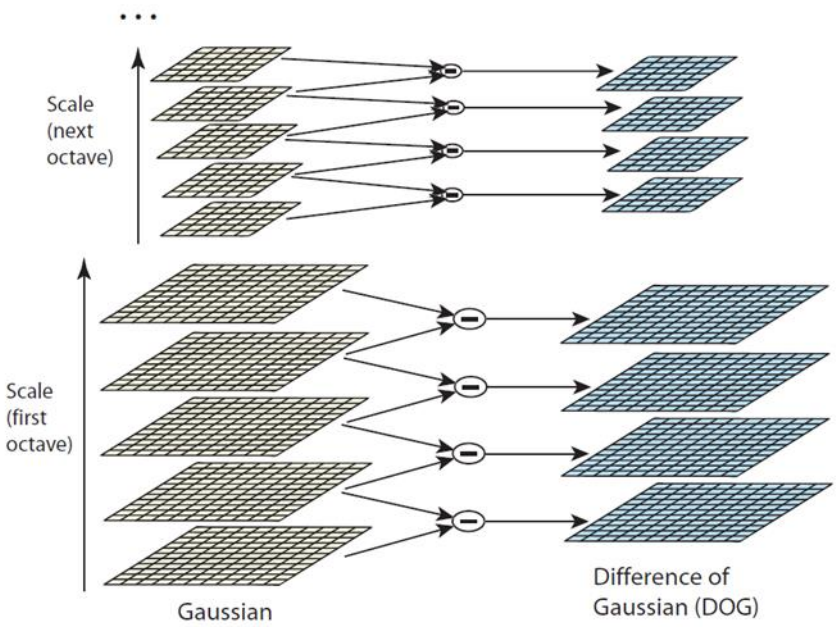


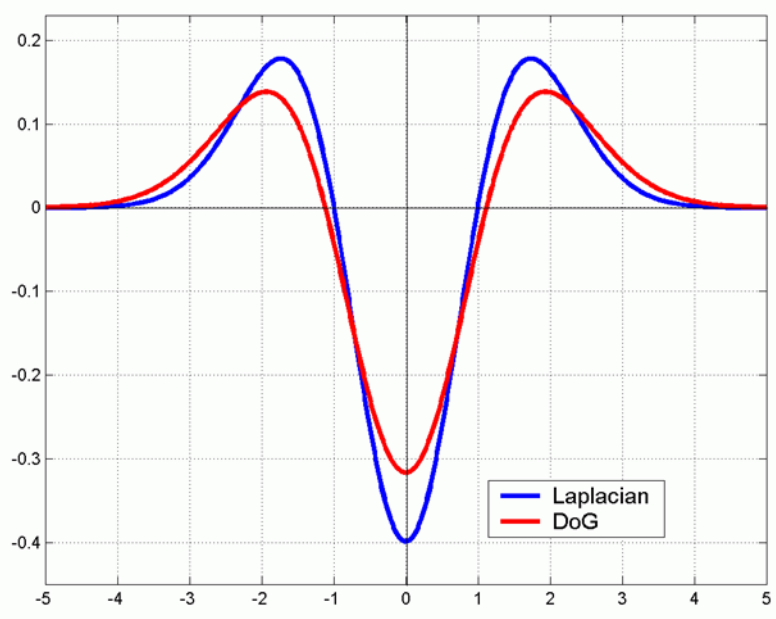



minima
maximum

Difference of Gaussians (DoG)

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G.$$

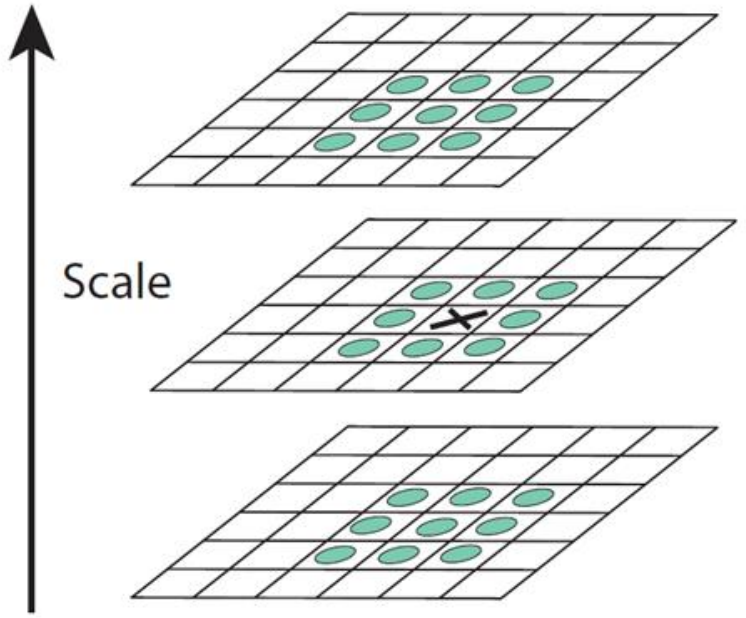




- Scale space is separated into octaves:
  - Octave 1 uses scale σ
  - Octave 2 uses scale 2σ
- Adjacent Gaussians are subtracted to produce the DOG
- After each octave, the Gaussian image is down-sampled by a factor of 2 to produce an image ¼ the size to start the next level.

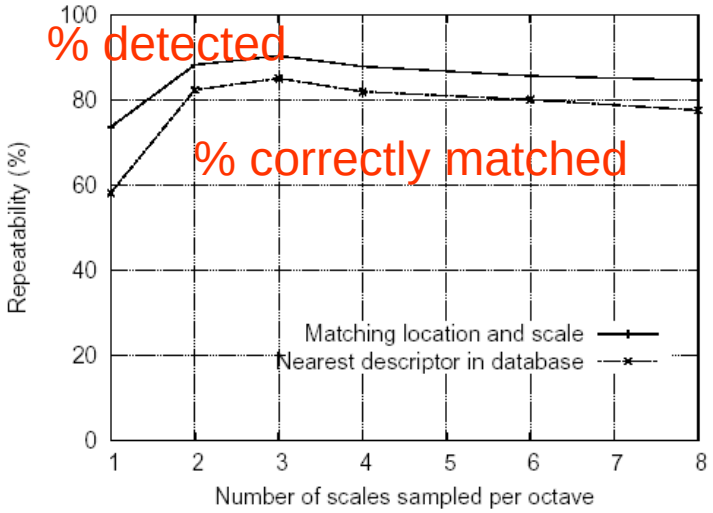
Step 2 : Keypoint Detection

- In order to detect the local maxima and minima of DoG space (D(x, y, σ)), each sample point is compared with its neighbours
- Eight neighbours in the current image and nine neighbours in the scale above and below are used
- A point is selected as a keypoint only if it is larger than all of these neighbours or smaller than all of them

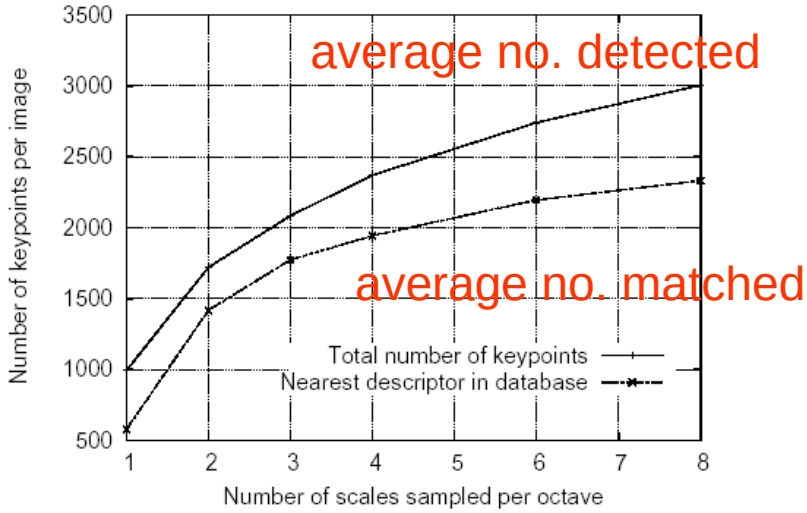


For each max or min found, output is the **location** and the **scale**.

Scale-space extrema detection: experimental results over 32 images that were synthetically transformed and noise added.



Stability

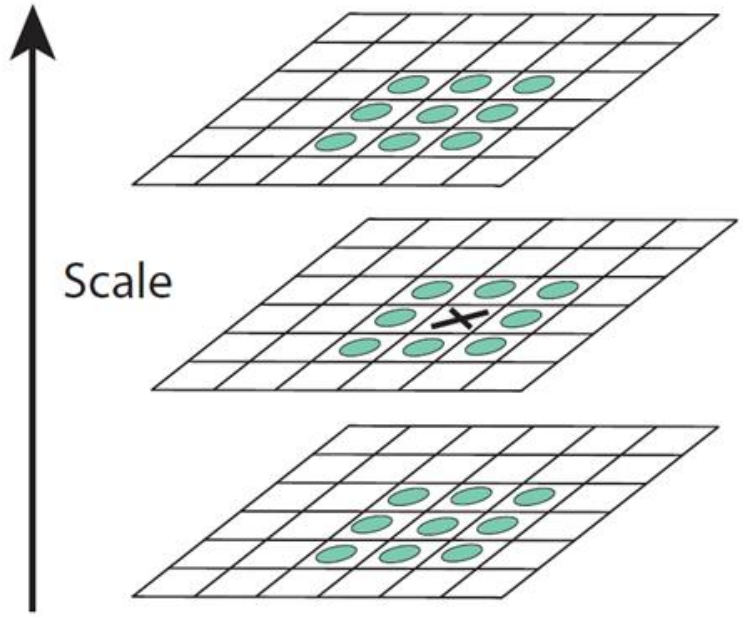


Expense

- Sampling in scale for efficiency
  - How many scales should be used per octave? S=?
    - More scales evaluated, more keypoints found
    - S < 3, stable keypoints increased too
    - S > 3, stable keypoints decreased
    - S = 3, maximum stable keypoints found

Keypoint Detection

- Usually, a non-maxima or non-minima position won't have to go through all 26 checks. A few initial checks will usually sufficient to discard some irrelevant points
- There won’t be enough neighbours in the lowermost and topmost scales and can skip those scales
- The detected maxima and minima may not be over exact pixel locations and need to be approximated to a nearest pixel location
- Using the available pixel data, subpixel values are generated using Taylor expansion of the image around the approximate key point



Getting rid of low contrast keypoints

If the magnitude of the intensity (i.e., without sign) at the current pixel in the DoG image (that is being checked for minima/maxima) is less than a certain value, it is rejected.

Removing edges

The idea is to calculate two gradients at the keypoint. Both perpendicular to each other. Based on the image around the keypoint, three possibilities exist. The image around the keypoint can be:

- **A flat region:** If this is the case, both gradients will be small.
- **An edge:** Here, one gradient will be big (perpendicular to the edge) and the other will be small (along the edge)
- **A "corner":** Here, both gradients will be big.

Corners are great keypoints. So we want just corners. If both gradients are big enough, we let it pass as a key point. Otherwise, it is rejected.

Eliminating the Edge Response

- Reject flats:
  - $|D(\hat{\mathbf{x}})| < 0.03$
- Reject edges:

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

Let α be the eigenvalue with larger magnitude and β the smaller.

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

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

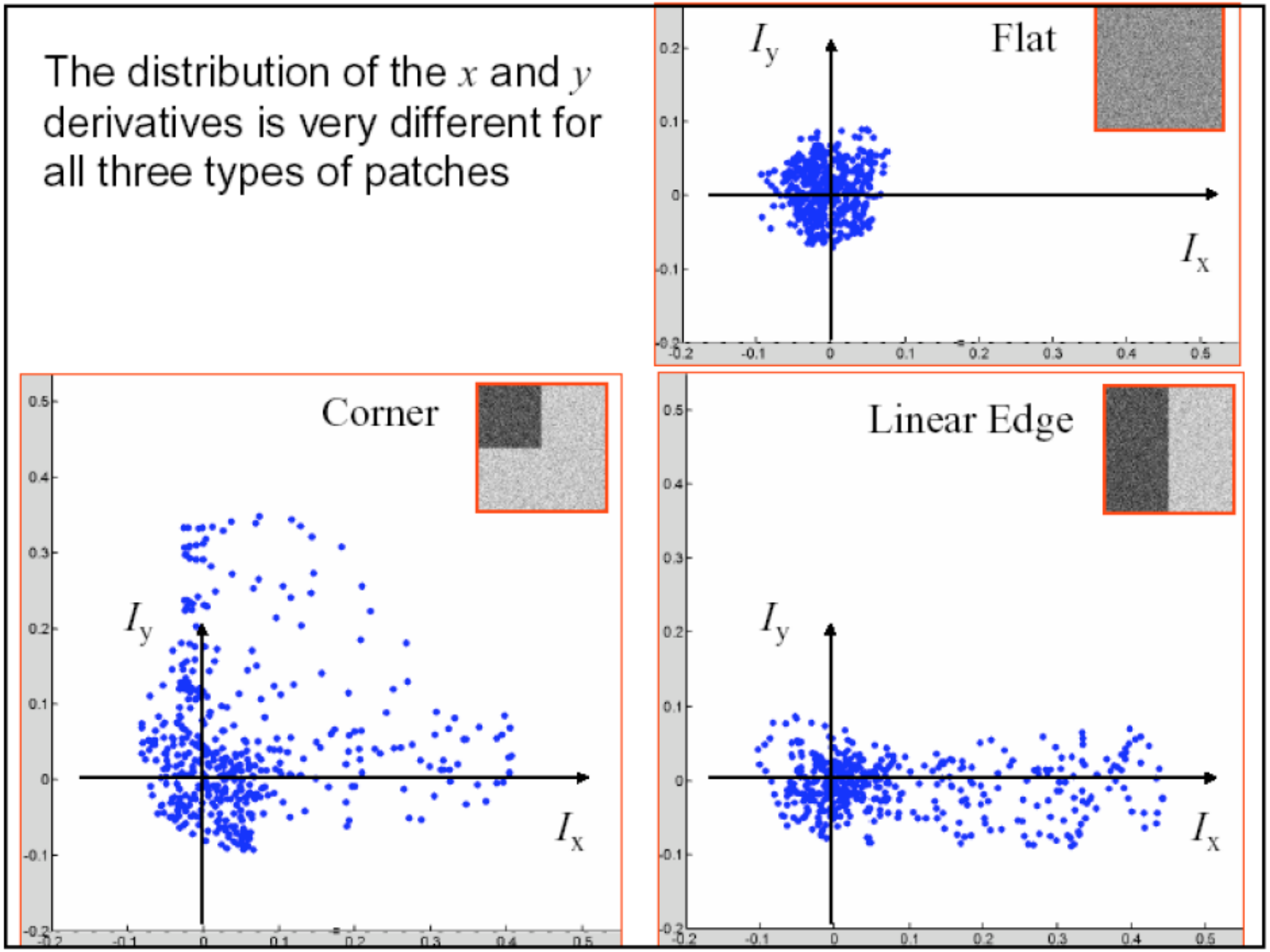
Let $r = \alpha/\beta$. So $\alpha = r\beta$

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

$(r+1)^2/r$ is at a min when the 2 eigenvalues are equal.

  - $r < 10$
- What does this look like?

The distribution of the x and y derivatives is very different for all three types of patches



Detected keypoints

Removal of low-contrast keypoints

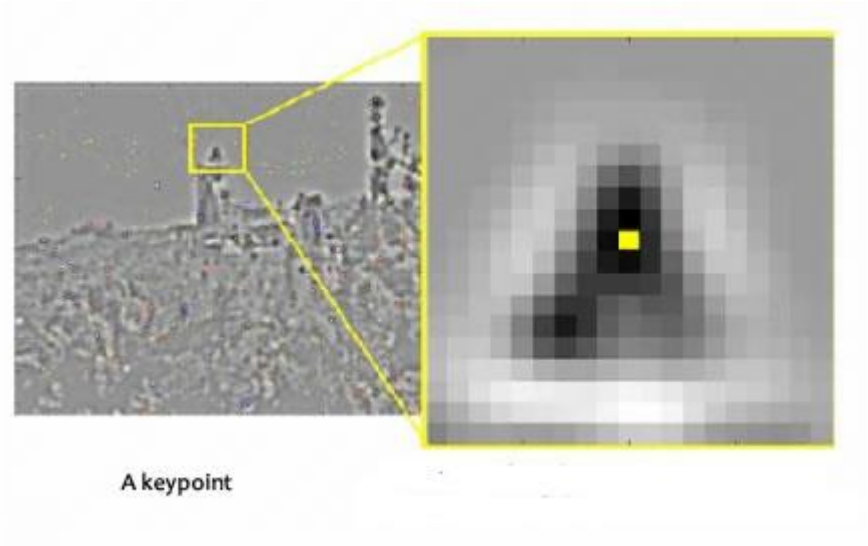
Removal of high-contrast keypoints residing on edges

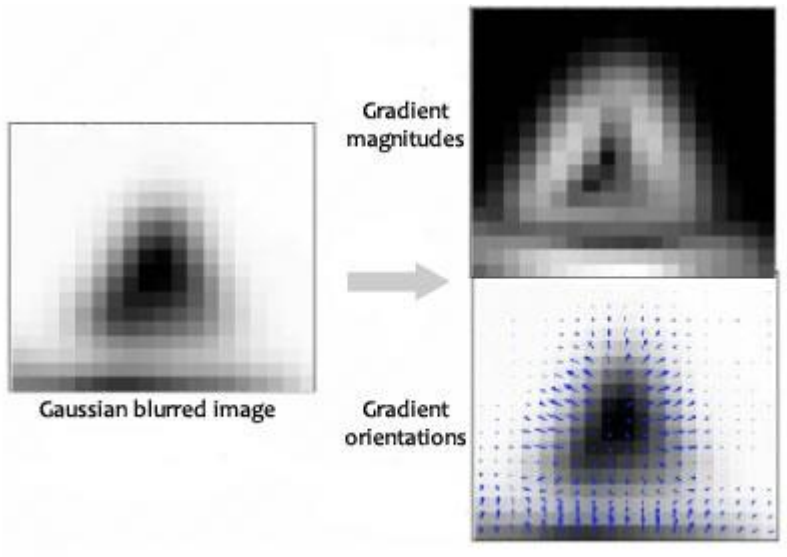
Step 3 : Keypoint orientations

- The idea is to collect gradient directions and magnitudes around each keypoint.
- Compute a consistent orientation to each keypoint based on local image properties
- This orientation can be used as a reference while computing feature descriptor
- Helps to make rotation/orientation invariant keypoint descriptors
- Orientation is obtained by collecting gradient directions and magnitudes around each keypoint

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

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$



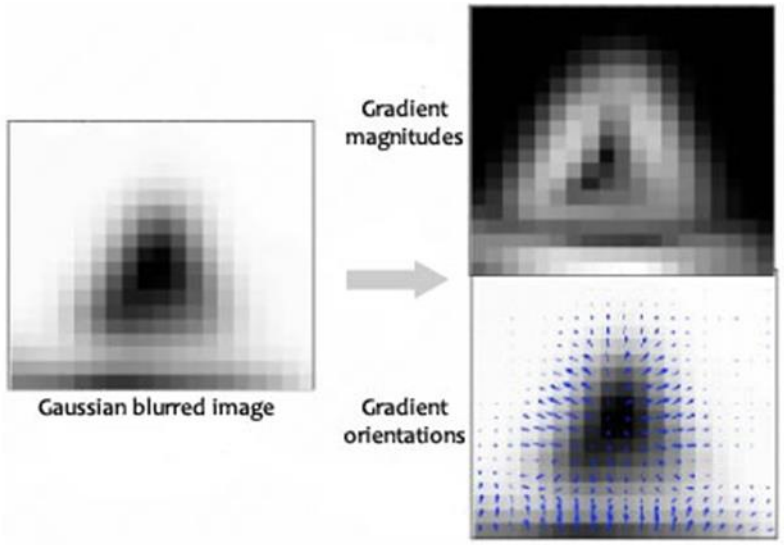


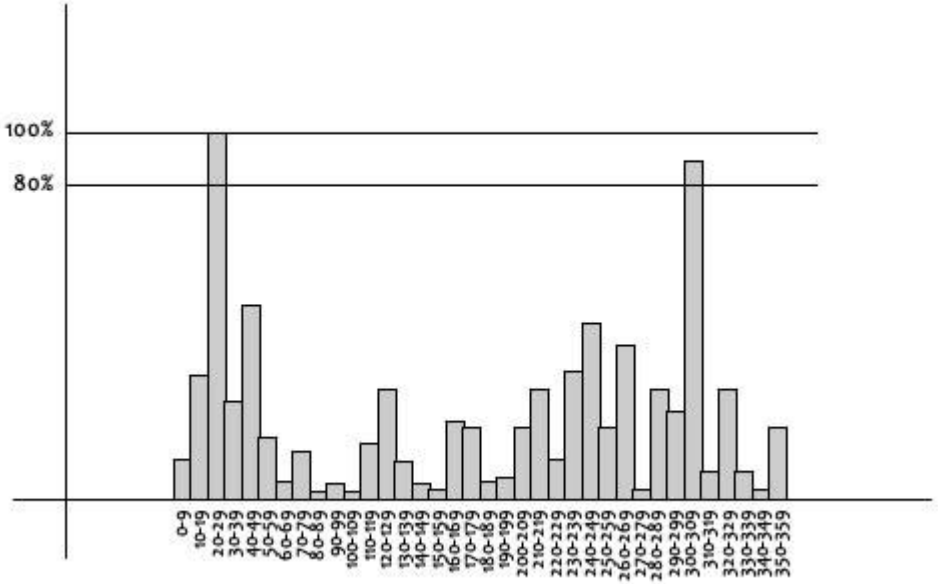
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

The magnitude and orientation is calculated for all pixels around the keypoint. Then, A histogram is created from this.

In this histogram, the 360 degrees of orientation are broken into 36 bins (each 10 degrees).



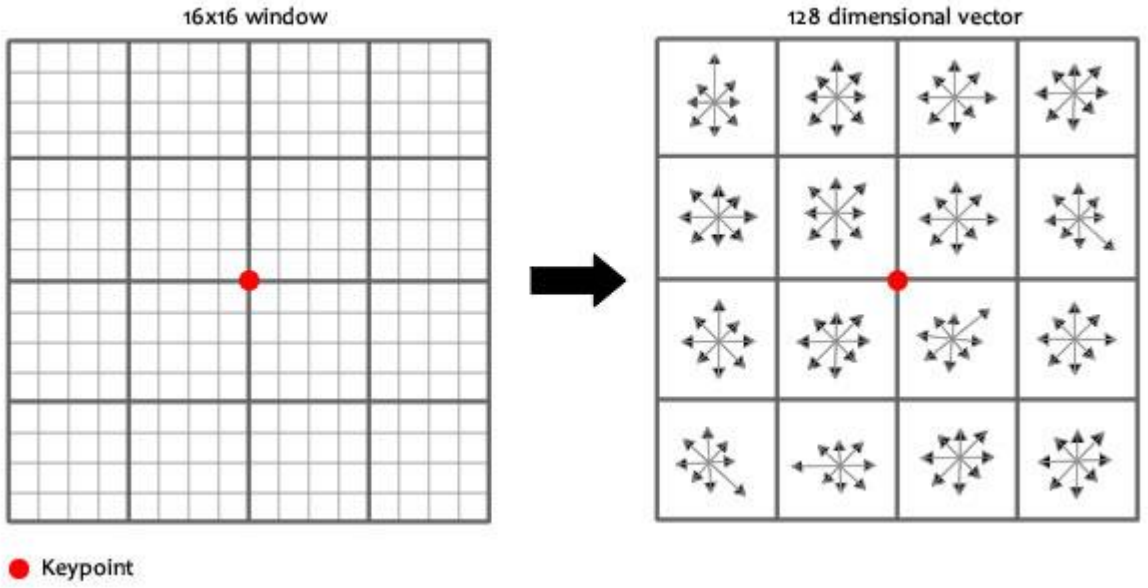


Above, you see the histogram peaks at 20-29 degrees. So, the keypoint is assigned orientation 3 (the third bin)

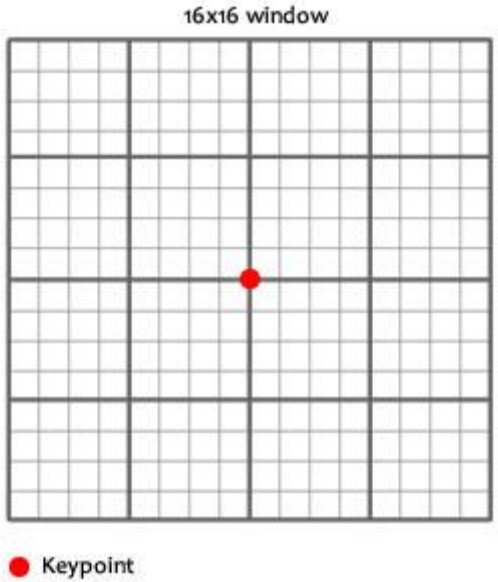
Generation of keypoint descriptors.

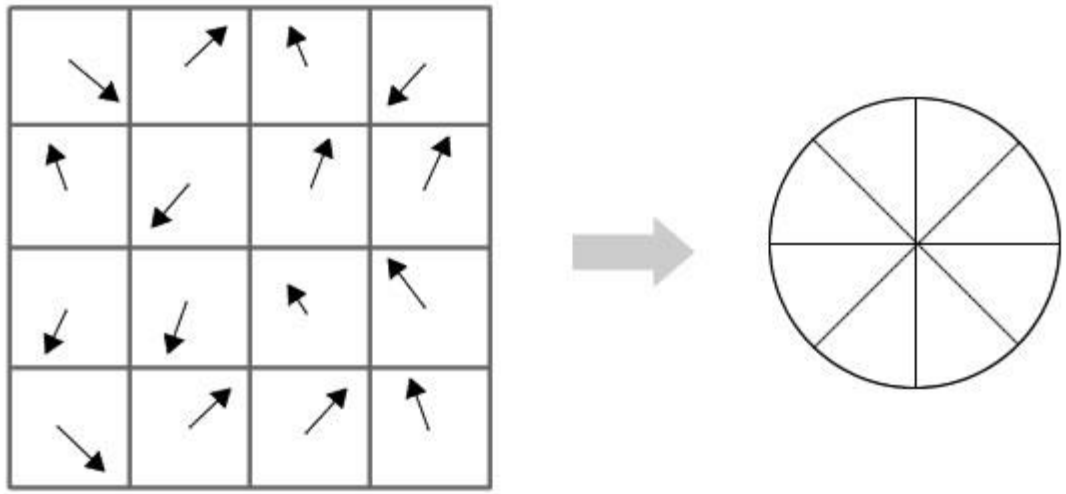
- At this point, each keypoint has
  - location
  - scale
  - orientation
- Next is to compute a descriptor for the local image region about each keypoint that is
  - highly distinctive
  - invariant as possible to variations such as changes in viewpoint and illumination

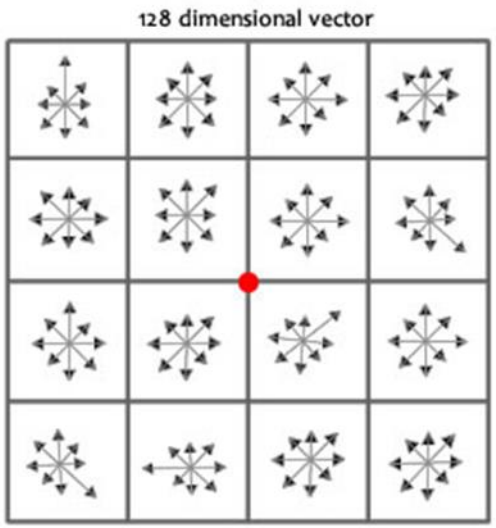
- We want to generate a very unique fingerprint for the keypoint.
- To do this, a 16x16 window around the keypoint. This 16x16 window is broken into sixteen 4x4 windows.

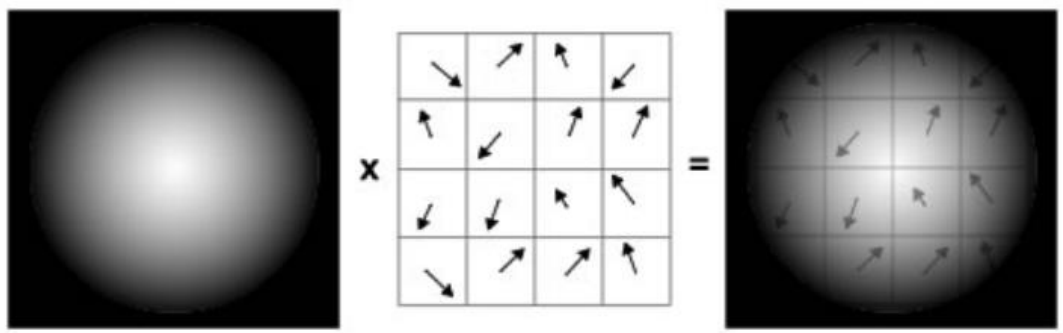


Within each 4x4 window, gradient magnitudes and orientations are calculated. These orientations are put into an 8 bin histogram.

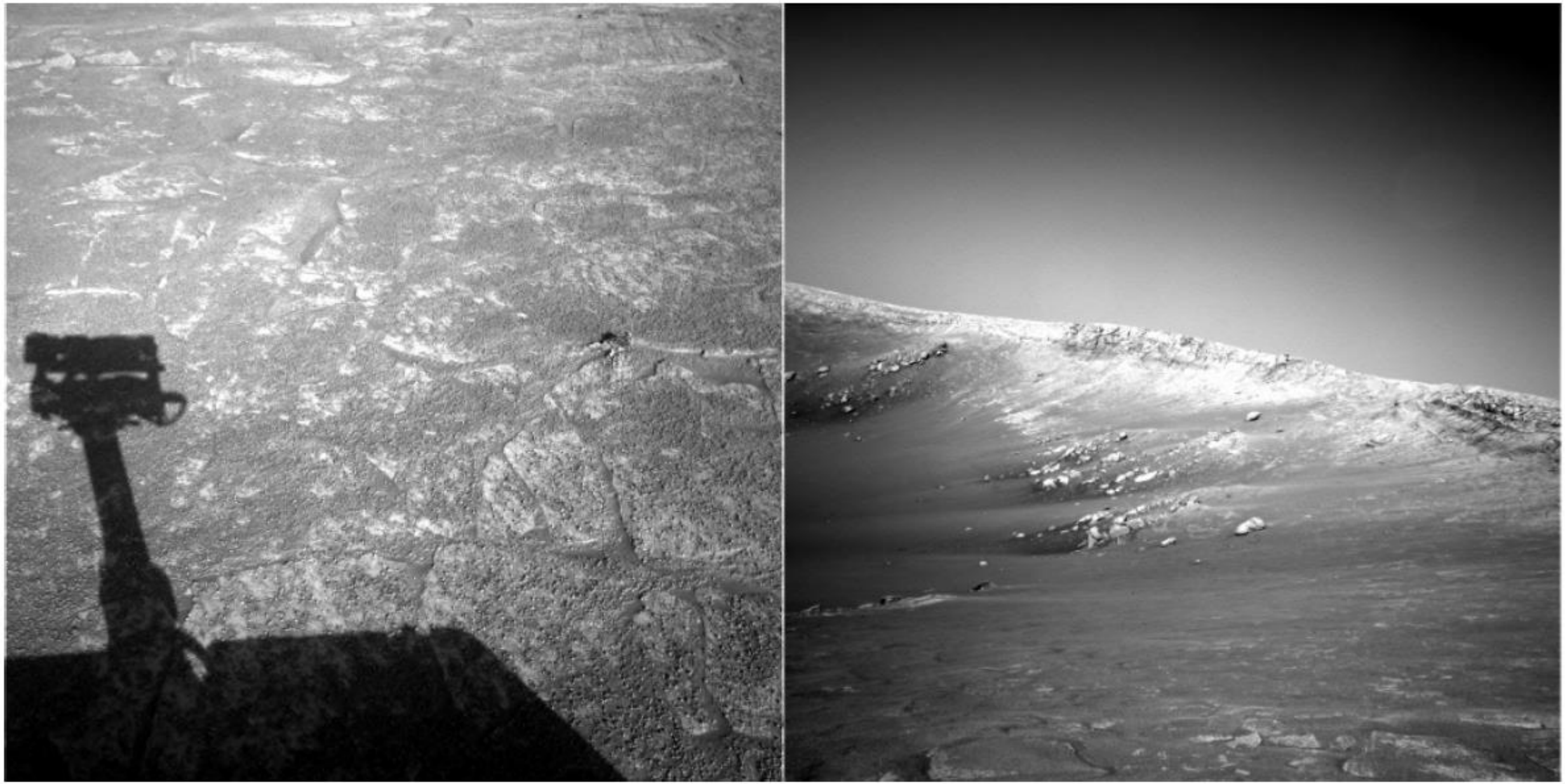




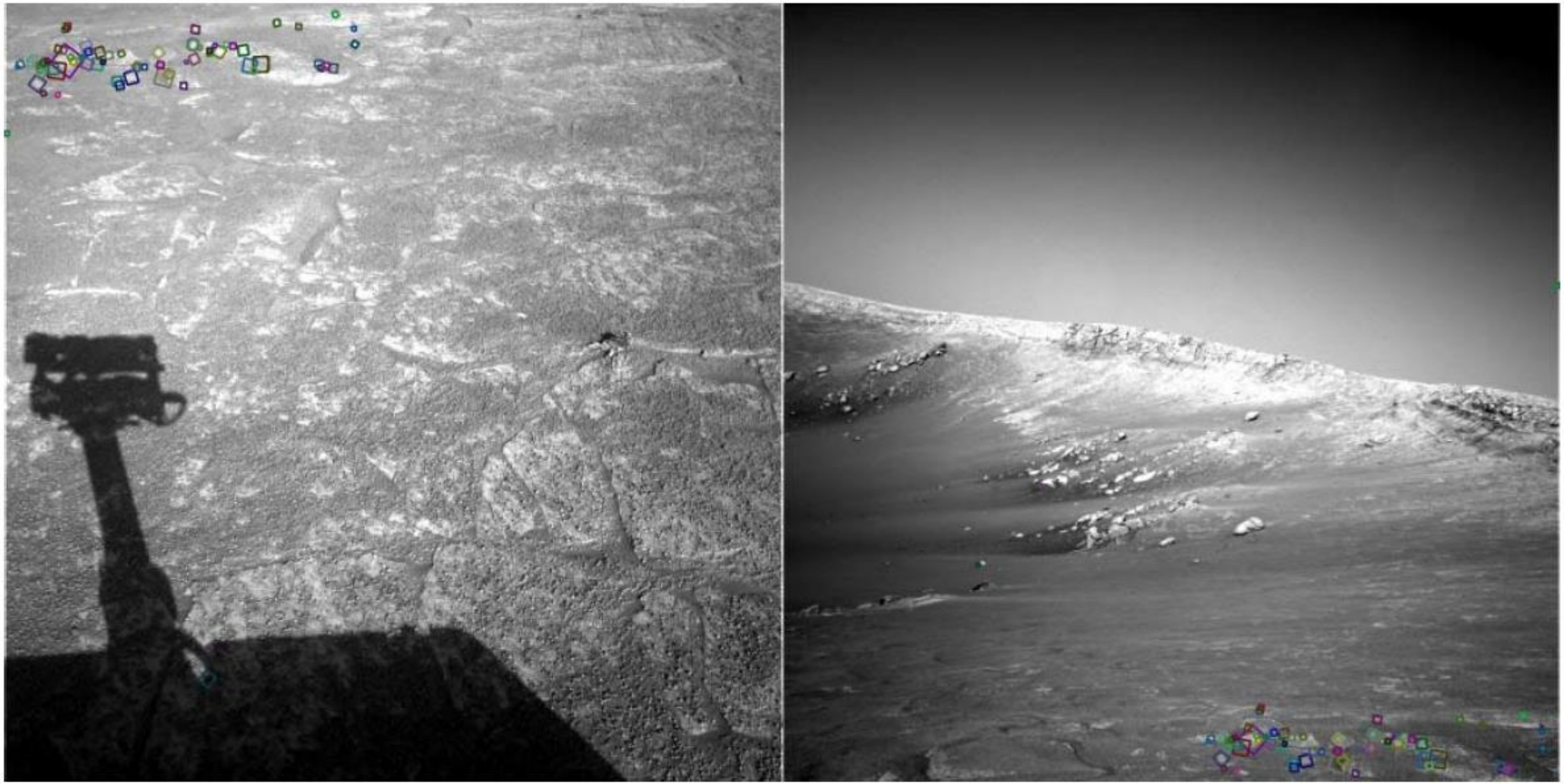


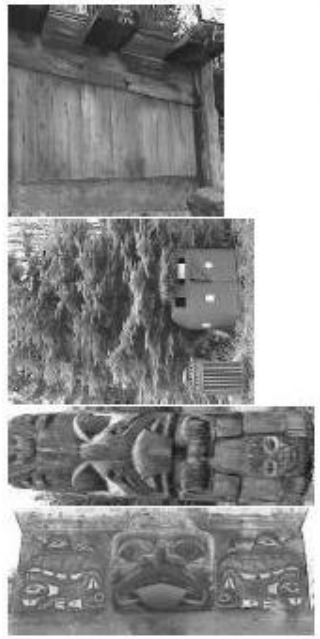


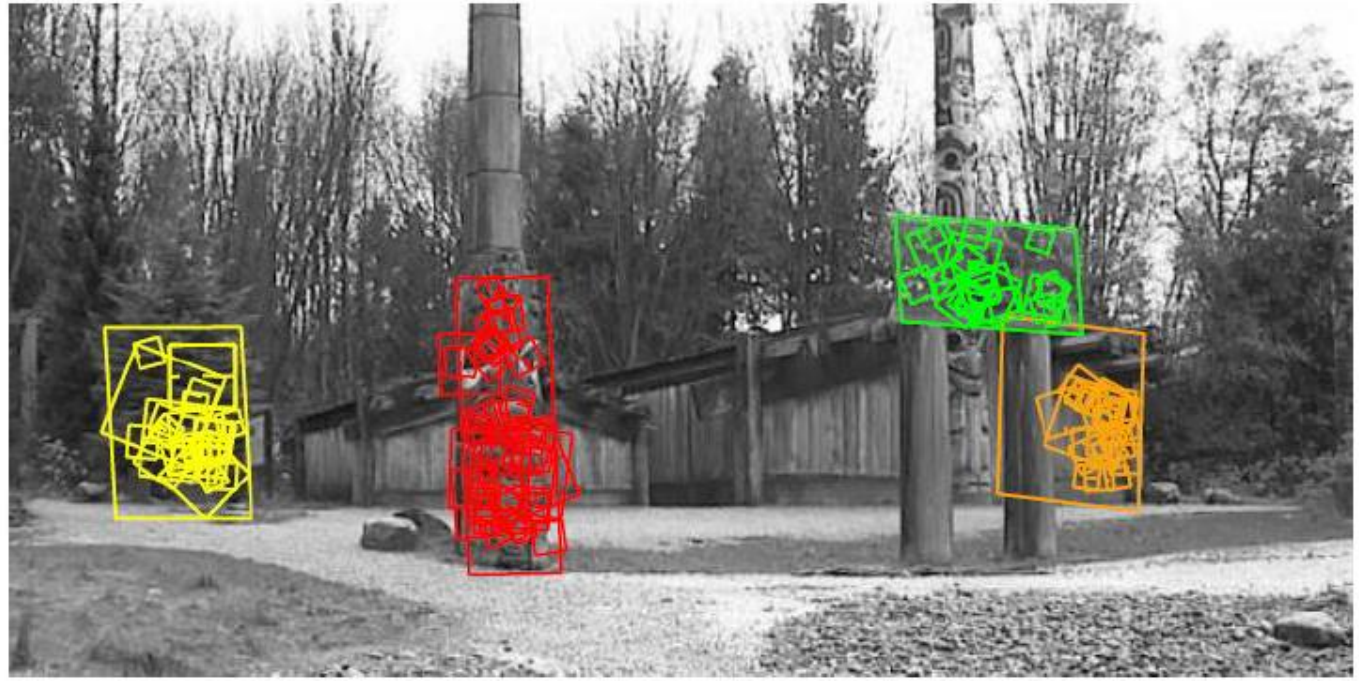
4 X 4 descriptors over 16 X 16 sample array were used in practice. 4 X 4 times 8 directions gives a vector of 128 values.



NASA Mars Rover images







Uses for SIFT

- Feature points are used also for:
  - Image alignment (homography, fundamental matrix)
  - 3D reconstruction (e.g. Photo Tourism)
  - Motion tracking
  - Object recognition
  - Indexing and database retrieval
  - Robot navigation
  - … many others