SIFT

Scale Invariant Feature Transform

Overview

A SIFT feature is a selected image region (also called keypoint) with an associated descriptor.

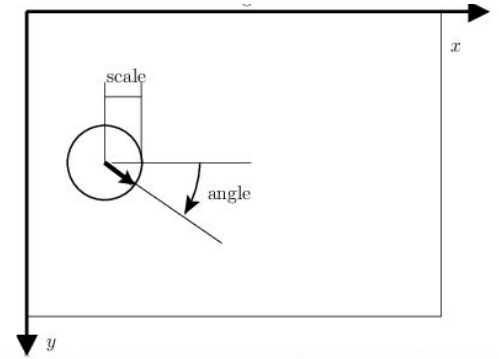
Keypoints are extracted by the **SIFT detector** and their descriptors are computed by the **SIFT descriptor**.

It is also common to use independently the SIFT detector (i.e. computing the keypoints without descriptors) or the SIFT descriptor (i.e. computing descriptors of custom keypoints).

SIFT Detector

A SIFT *keypoint* is a circular image region with an orientation. It is described by a geometric *frame* of four parameters:

- The keypoint center coordinates *x* and *y*
- Its *scale* (the radius of the region)
- and its *orientation* (an angle expressed in radians).



SIFT keypoints are circular image regions with an orientation.

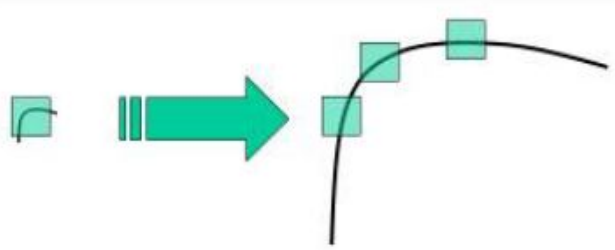
The SIFT detector uses as keypoints image structures which resemble “blobs”. By searching for blobs at multiple scales and positions, the SIFT detector is invariant (or, more accurately, covariant) to translation, rotations, and re scaling of the image.

Motivation

Harris is rotation-invariant, which means, even if the image is rotated, we can find the same corners. It is obvious because corners remain corners in rotated image also. But what about scaling?

A corner may not be a corner if the image is scaled.

For example, check a simple image below.
A corner in a small image within a small window is flat when it is zoomed in the same window.
So Harris corner is not scale invariant.



Four steps involved in SIFT algorithm

1) **Scale-space Extrema Detection**
    - **Search over multiple scales and image locations**
2) **Keypoint Localization**
    - **Fit a model to determine location and scale.**
    - **Select keypoints based on a measure of stability.**
3) **Orientation Assignment**
    - **Compute best orientation for each keypoint region.**
4) **Keypoint Descriptor**
    - **Use local image gradients at selected scale and rotation to describe each keypoint region.**

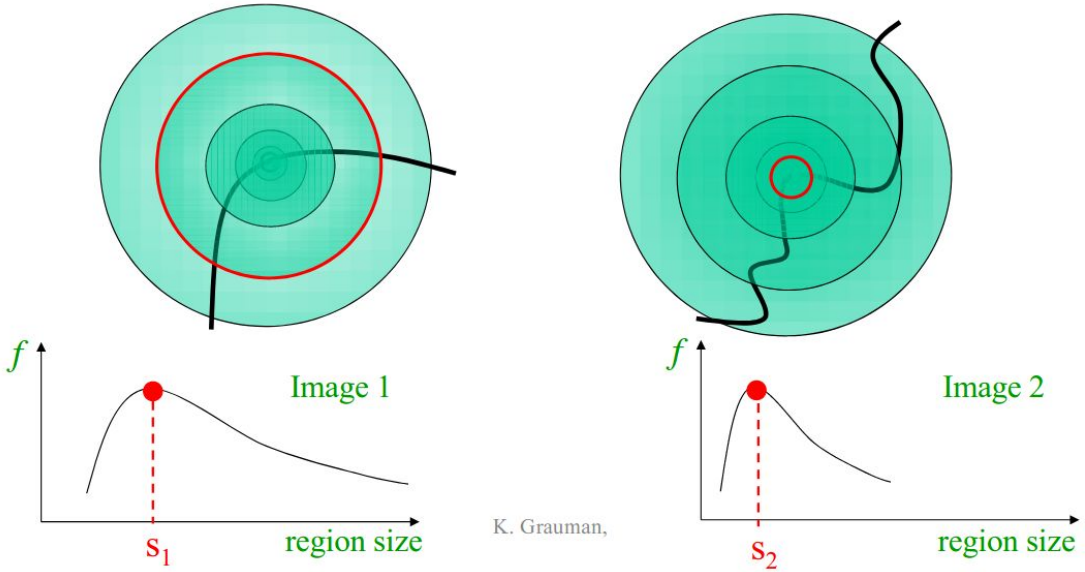
Scale-space Extrema Detection

We can’t use the same window to detect keypoints with different scale. It is OK with small corner. But to detect larger corners we need larger windows.

To solve this, scale-filtering is used. In it, Laplacian of Gaussian is found for the image with various sigma values. Gaussian kernel with low sigma gives high value for small corner while gaussian kernel with high sigma fits well for larger corner.

We can find the local maxima across the scale and space which gives us a list of (x,y,sigma) values which means there is a potential keypoint at (x,y) at sigma scale. But this LoG is a little costly, so SIFT algorithm uses Difference of Gaussians which is an approximation of LoG.

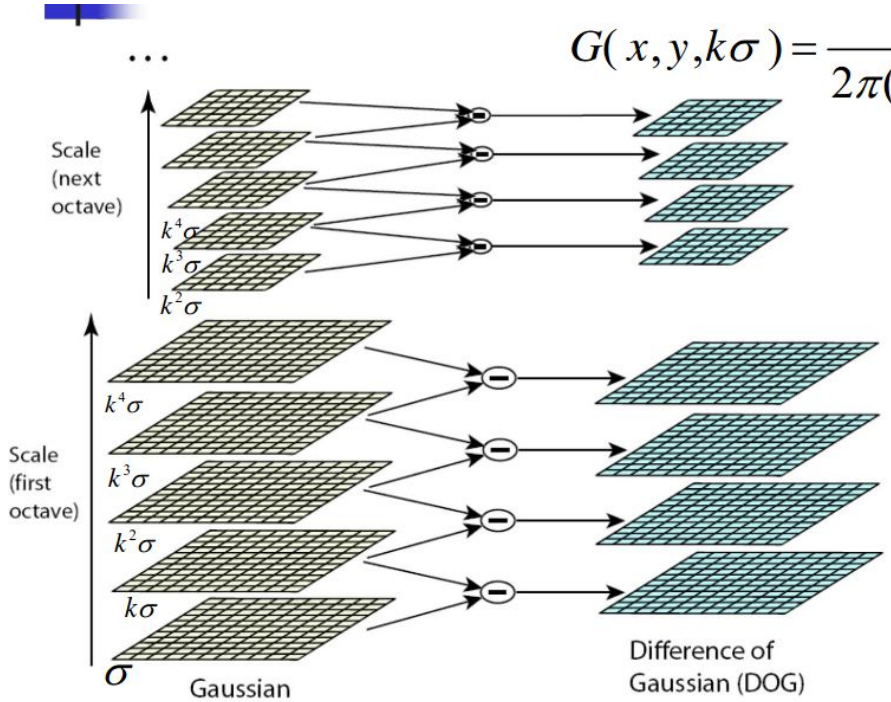
Scale Selection



Gaussian kernel with low sigma gives high value for small corner while gaussian kernel with high sigma fits well for larger corner.

Gaussian Pyramid

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



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

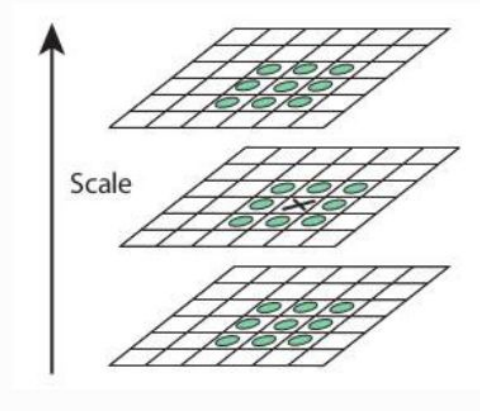
$$k = \sqrt{2}$$

Difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different sigma. 512X512 image is subsampled to say 256 X 256. [Resolution reduced to ¼]
In this case, number of octaves = 4, number of scale levels = 5, sigma = 0.70717, k = sqrt(2)

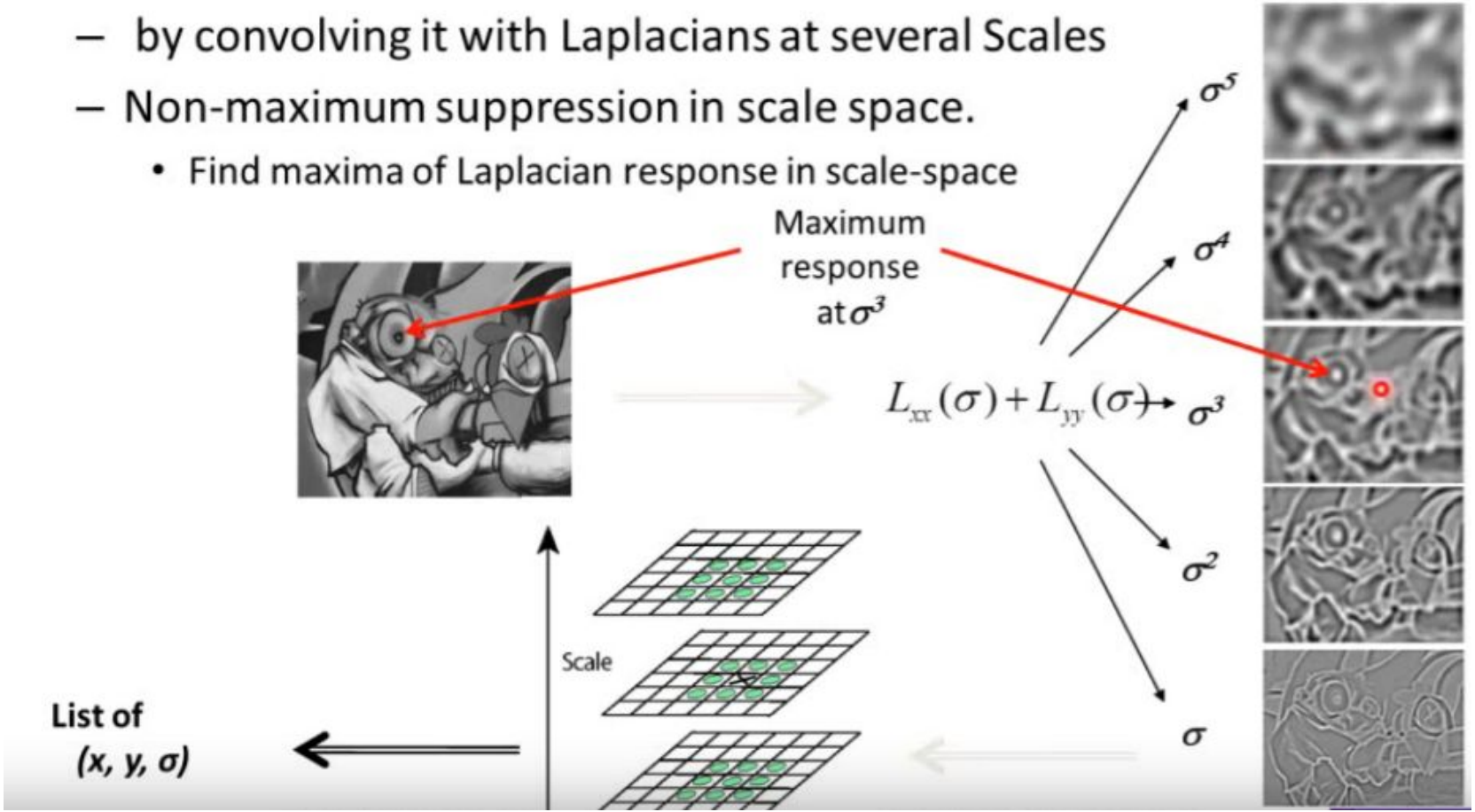
Potential Key Points

Compare a pixel (X) with 26 pixels in current and adjacent scales. One pixel in an image is compared with its 8 neighbours as well as 9 pixels in next scale and 9 pixels in previous scales.

Select a pixel (X) if larger/smaller than all 26 pixels
=> Interest point



- Finding the characteristic scale of the blob
  - by convolving it with Laplacians at several Scales
  - Non-maximum suppression in scale space.
    - Find maxima of Laplacian response in scale-space



Key Points after extrema detection

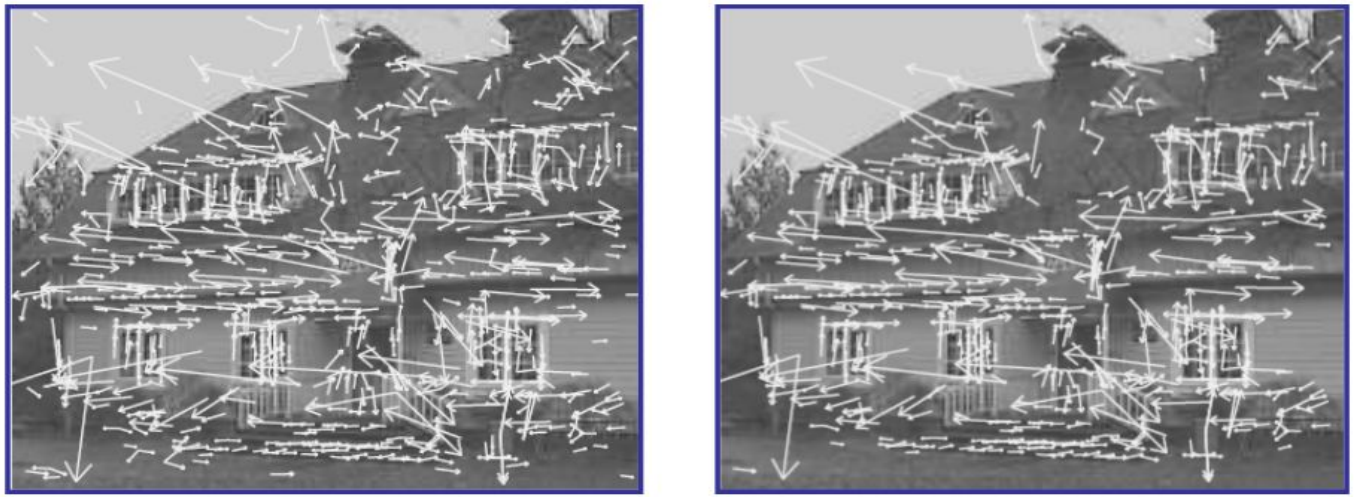
original image

extrema locations

832 initial keypoints

Keypoint localization

Once potential keypoints locations are found, they have to be refined to get more accurate results.
If the intensity at this extrema is less than a threshold value (0.03 as per the paper), it is rejected. This threshold is called **contrastThreshold** in OpenCV



from 832 key points to 729 key points, th=0.03.

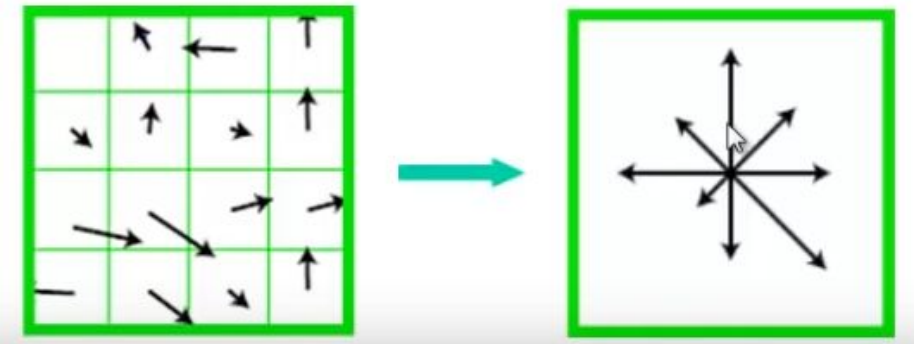
DoG has higher response for edges, so edges also need to be removed. For this, a concept similar to Harris corner detector is used. We know from Harris corner detector that for edges, one eigen value is larger than the other.

Hence, we take ratio of both eigenvalues, if this ratio is greater than a threshold, called **edgeThreshold** in OpenCV, that keypoint is discarded. It is given as 10 in paper.

from 729 key points to 536 key points.

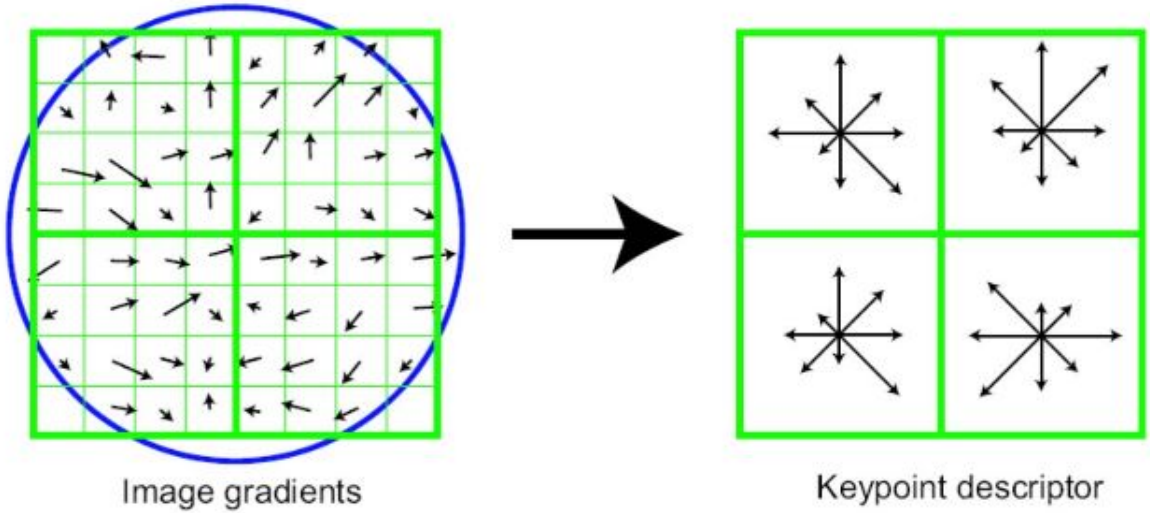
Orientation Assignment

- To achieve rotation invariance
- A neigbourhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region.
- An orientation histogram with 36 bins covering 360 degrees is created. (weighted by gradient magnitude).
- Select the peak as direction of the key point
- Each key point now has (x,y,scale,orientation)



Keypoint Descriptor

A 16x16 neighbourhood around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. => 16 such sub-blocks will be created.
For each subblock, we will form weighted histogram with (8 bins). Hence, SIFT descriptor will be of 16*8 => 128 dimensions



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

Extensibility: can easily be extended to wide range of differing feature types, with each adding robustness

References

- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html
- https://www.crcv.ucf.edu/wp-content/uploads/2019/03/Lecture-6-SIFT.pdf
- https://www.youtube.com/watch?v=NPcMS49V5hg
- https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf