

Local Invariant Features

Prof. Giuseppe Lisanti

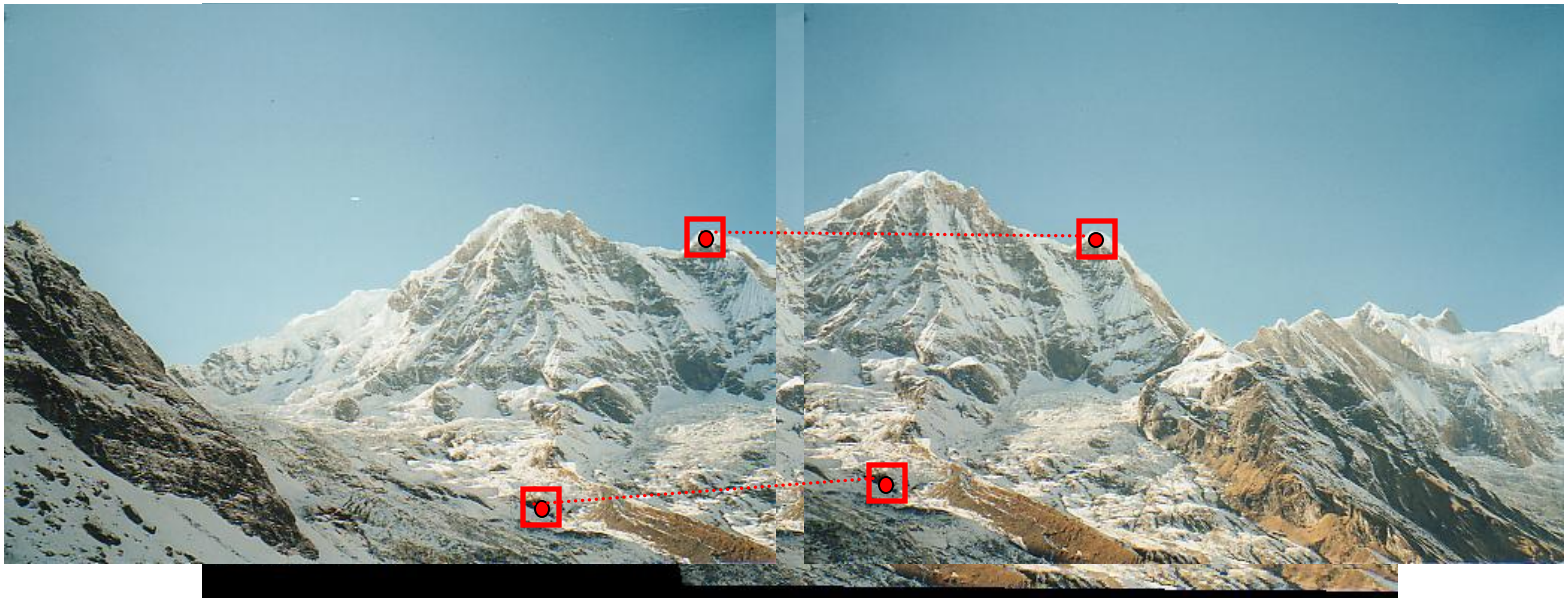
giuseppe.lisanti@unibo.it

Local Invariant Feature

- Several Computer Vision tasks deal with finding **“Corresponding Points”** between two (or more) images of a scene
 - Correspondences = image points which are the projection of the same 3D point in different views of the scene
 - Establishing correspondences may be difficult as these points may look different in the different views (e.g. due to viewpoint variations and/or lighting changes)

Panorama Stitching

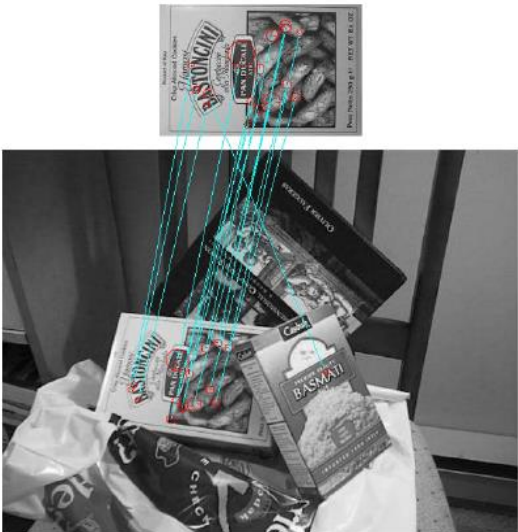
- Create a larger image by aligning two images of the same scene
 - The two images may be aligned by estimating a homography, which requires at least 4 correspondences (more is better)
 - Find “salient points” independently in the two images
 - Compute a local “description” for each salient point
 - Compare descriptions to find matching pairs



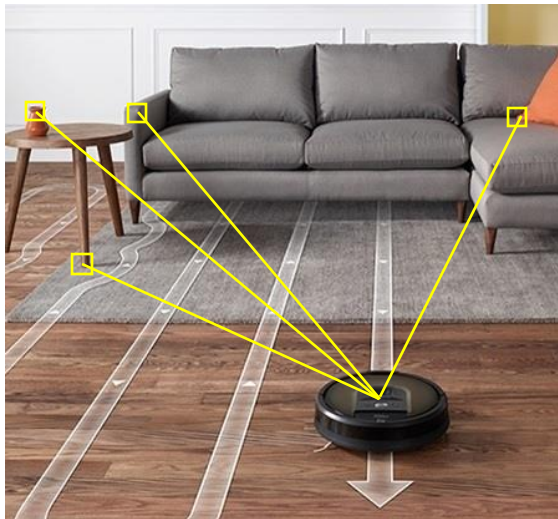
Local Invariant Feature

- Several Computer Vision tasks deal with finding **“Corresponding Points”** between two (or more) images of a scene
 - Correspondences = image points which are the projection of the same 3D point in different views of the scene
- Establishing correspondences may be difficult as these points may look different in the different views (e.g. due to viewpoint variations and/or lighting changes)

Instance-level Object Detection



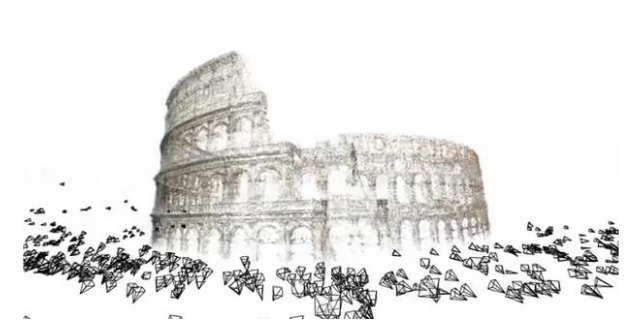
Robot Navigation



Augmented Reality



3D Reconstruction

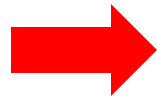


The Local Invariant Features Paradigm

- The task of establishing correspondences is split into 3 successive steps:
 - **Detection** of salient points (aka keypoints, interest points, feature points...)
 - **Description** - computation of a suitable **descriptor** based on pixels in the keypoint neighbourhood
 - **Matching** descriptors between images
- Descriptors should be **invariant** (robust) to as many transformations as possible



match

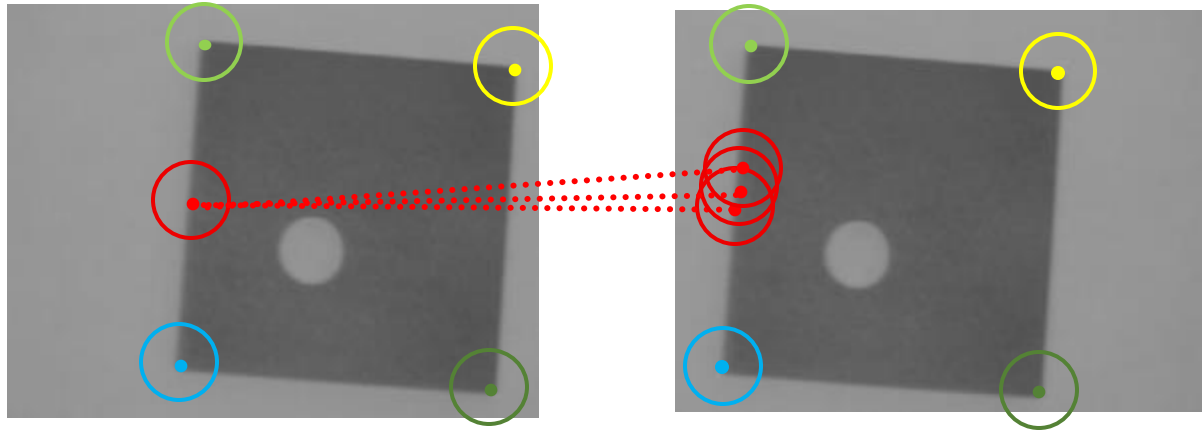


Properties of good detectors/descriptors

- Detector
 - **Repeatability**: it should find the same keypoints in different views of the scene despite the transformations undergone by the images
 - **Saliency**: it should find keypoints surrounded by informative patterns (good for making them discriminative for the matching)
- Descriptor
 - **Distinctiveness vs. Robustness Trade-off**: the description algorithm should capture the salient information around a keypoint, so to keep important tokens and disregard changes due to nuisances (e.g. light changes) and noise
 - **Compactness**: the description should be as concise as possible, to minimize memory occupancy and allow for efficient matching
- Speed is desirable for both, and in particular for detectors, which need to be run on the whole image (while descriptors are computed at keypoints only)

Interest Points vs. Edges

- Can edges be found repeatably across different images?
- Are these points easily identified across different images?
- Edge pixels can be hardly told apart as they look very similar along the direction perpendicular to the gradient
 - Edges are locally ambiguous, many other points that look just the same



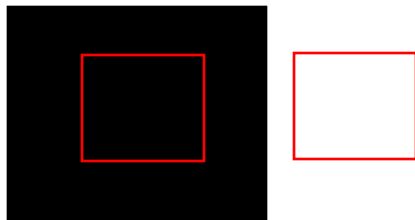
- Pixels exhibiting a large variation along all directions => better for establishing reliable correspondences

Moravec Interest Point Detector

- The cornerness at \mathbf{p} is given by the minimum squared difference between the patch (e.g. 7x7) centered at \mathbf{p} and those centered at its 8 neighbours.

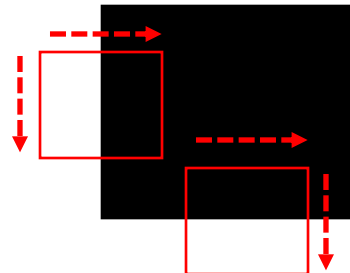
$$C(\mathbf{p}) = \min_{\mathbf{q} \in n_8(\mathbf{p})} \|N(\mathbf{p}) - N(\mathbf{q})\|^2$$

uniform region: no change
in all directions



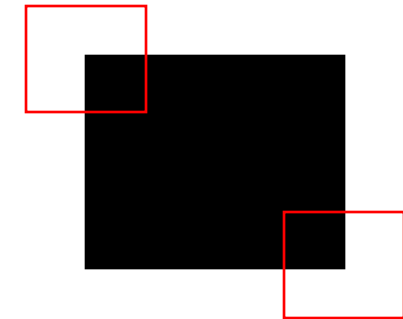
C is small

edge: no change along the
edge direction



C is small

corner: significant change in
all directions.

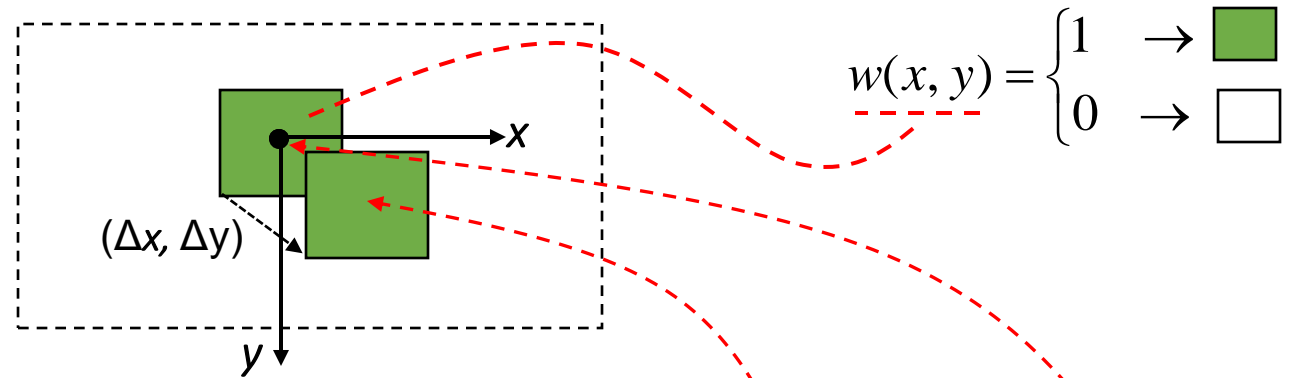


C is high

- After computing the cornerness \Rightarrow threshold and then NMS

Harris Corner Detector

- Harris&Stephens [2] proposed to rely on a **continuous formulation** of the Moravec's "error" function
- Assume to shift the image with a generic infinitesimal shift $(\Delta x, \Delta y)$:
 - $w(x,y)$ is a window set to 1 around the pixel under evaluation and 0 in all the image
 - in the end consider only the pixel around the x,y position
- Due to the shift being infinitesimal, we can deploy Taylor's expansion of the intensity function at (x,y) :



$$E(\Delta x, \Delta y) = \sum_{x,y} w(x,y) (I(x + \Delta x, y + \Delta y) - I(x,y))^2$$

$$f(x + \Delta x) = f(x) + f'(x)\Delta x$$

$$I(x + \Delta x, y + \Delta y) \cong I(x, y) + \frac{\partial I(x, y)}{\partial x} \Delta x + \frac{\partial I(x, y)}{\partial y} \Delta y$$

$$I(x + \Delta x, y + \Delta y) - I(x, y) \cong \underbrace{\frac{\partial I(x, y)}{\partial x}}_{I_x} \Delta x + \underbrace{\frac{\partial I(x, y)}{\partial y}}_{I_y} \Delta y = I_x(x, y) \Delta x + I_y(x, y) \Delta y$$

Harris Corner Detector

$$E(\Delta x, \Delta y) = \sum_{x,y} w(x,y) \left(I_x(x,y) \Delta x + I_y(x,y) \Delta y \right)^2 =$$

$$= \sum_{x,y} w(x,y) \left(I_x(x,y)^2 \Delta x^2 + I_y(x,y)^2 \Delta y^2 + 2 I_x(x,y) I_y(x,y) \Delta x \Delta y \right)$$

$$= \sum_{x,y} w(x,y) \left(\begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{pmatrix} I_x(x,y)^2 & I_x(x,y) I_y(x,y) \\ I_x(x,y) I_y(x,y) & I_y(x,y)^2 \end{pmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)$$

$$\begin{matrix} 1 \times 1 \\ \text{1x2} \end{matrix} \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{matrix} 2 \times 2 \end{matrix} \left(\begin{matrix} \sum_{x,y} w(x,y) I_x(x,y)^2 & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y(x,y)^2 \end{matrix} \right) \begin{matrix} 2 \times 1 \\ \end{matrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$E(\Delta x, \Delta y) = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

weighted sum of derivatives around the point

Harris Corner Detector

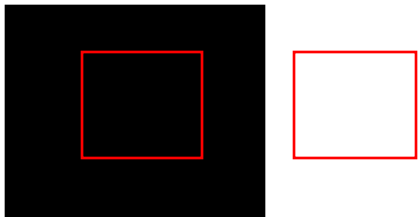
- **M** encodes the local image structure around the considered pixel.

- Let us **hypothesize** that M is a diagonal matrix:

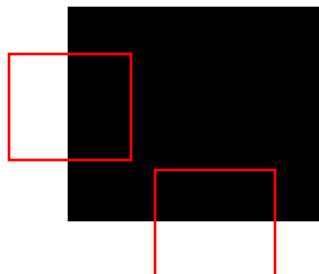
$$M = \begin{pmatrix} \sum_{x,y} w(x,y) I_x(x,y)^2 & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y(x,y)^2 \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

$$E(\Delta x, \Delta y) = [\Delta x \ \Delta y] M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = [\Delta x \ \Delta y] \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \lambda_1 \Delta x^2 + \lambda_2 \Delta y^2$$

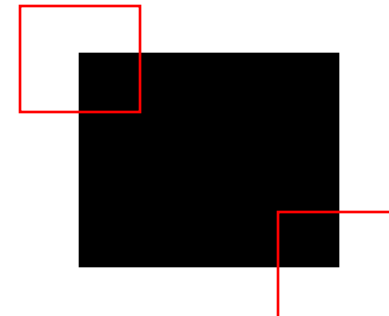
$\lambda_1, \lambda_2 \cong 0$: Flat



$\lambda_1 \gg \lambda_2$: Edge



$\lambda_1, \lambda_2 \uparrow$: Corner



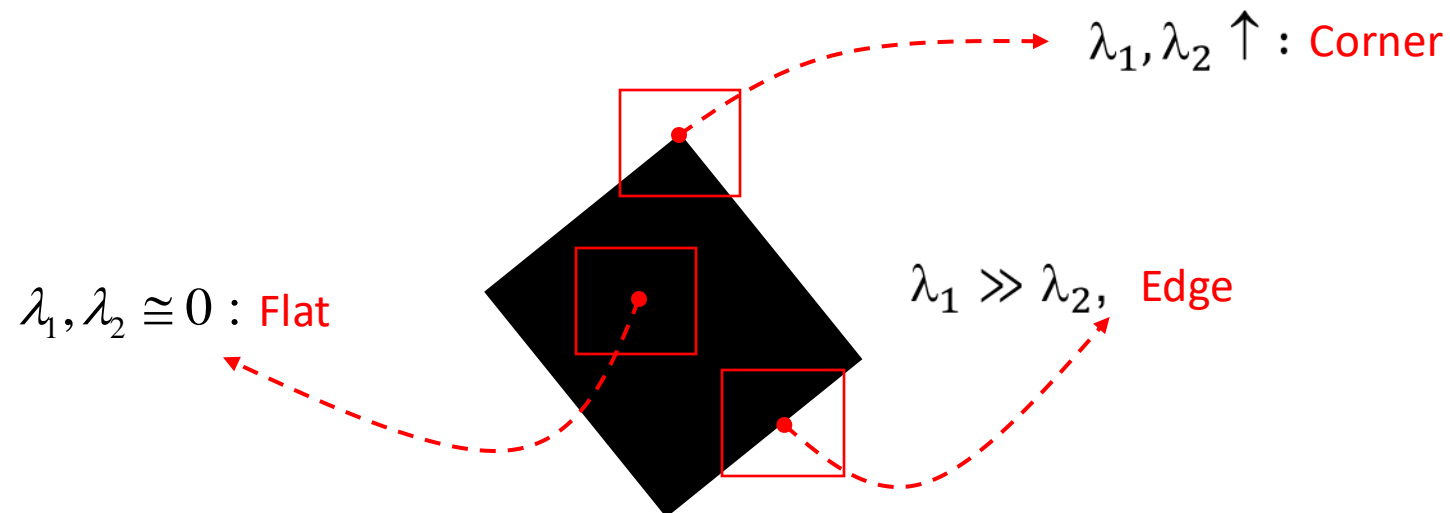
Harris Corner Detector

- The previous considerations have general validity as M is real and symmetric, and thus can always be diagonalized by a rotation of the image coordinate system

$$M = R \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R^T$$

- The columns of R are the orthogonal unit eigenvectors of M
- λ_i the corresponding eigenvalues
- R^T is the rotation matrix that aligns the image axes to the eigenvectors of M

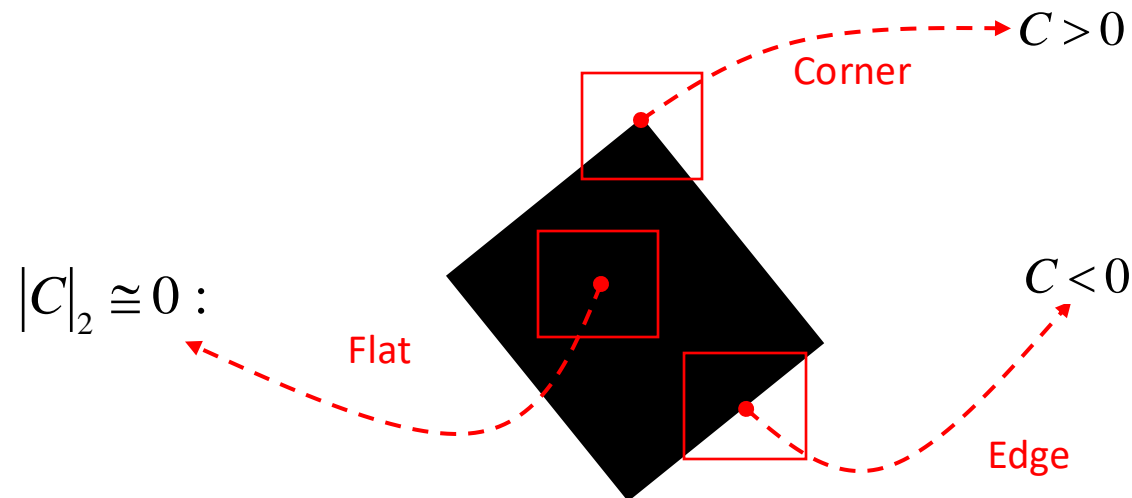
We just compute the eigenvalues of M !



Harris Corner Detector

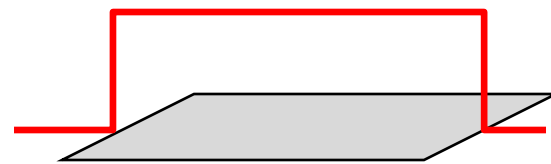
- You need to compute eigenvalues at each pixel => costly
- Compute a more efficient “cornerness” function:
- Analysis of the above function would show that:

$$C = \det(M) - k \cdot \text{tr}(M)^2 = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

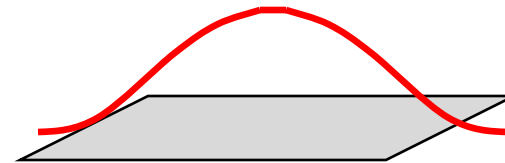


Harris Corner Detector

- The Harris corner detection algorithm can thus be summarized as follows:
 1. Compute C at each pixel
 2. Select all pixels where C is higher than a chosen positive threshold (T)
 3. Within the previous set, detect as corners only those pixels that are local maxima of C (NMS)
- It is worth highlighting that the weighting function $w(x,y)$ used by the Harris corner detector is Gaussian rather than Box-shaped, so to assign more weight to closer pixels and less weight to those farther away



1 in window, 0 outside

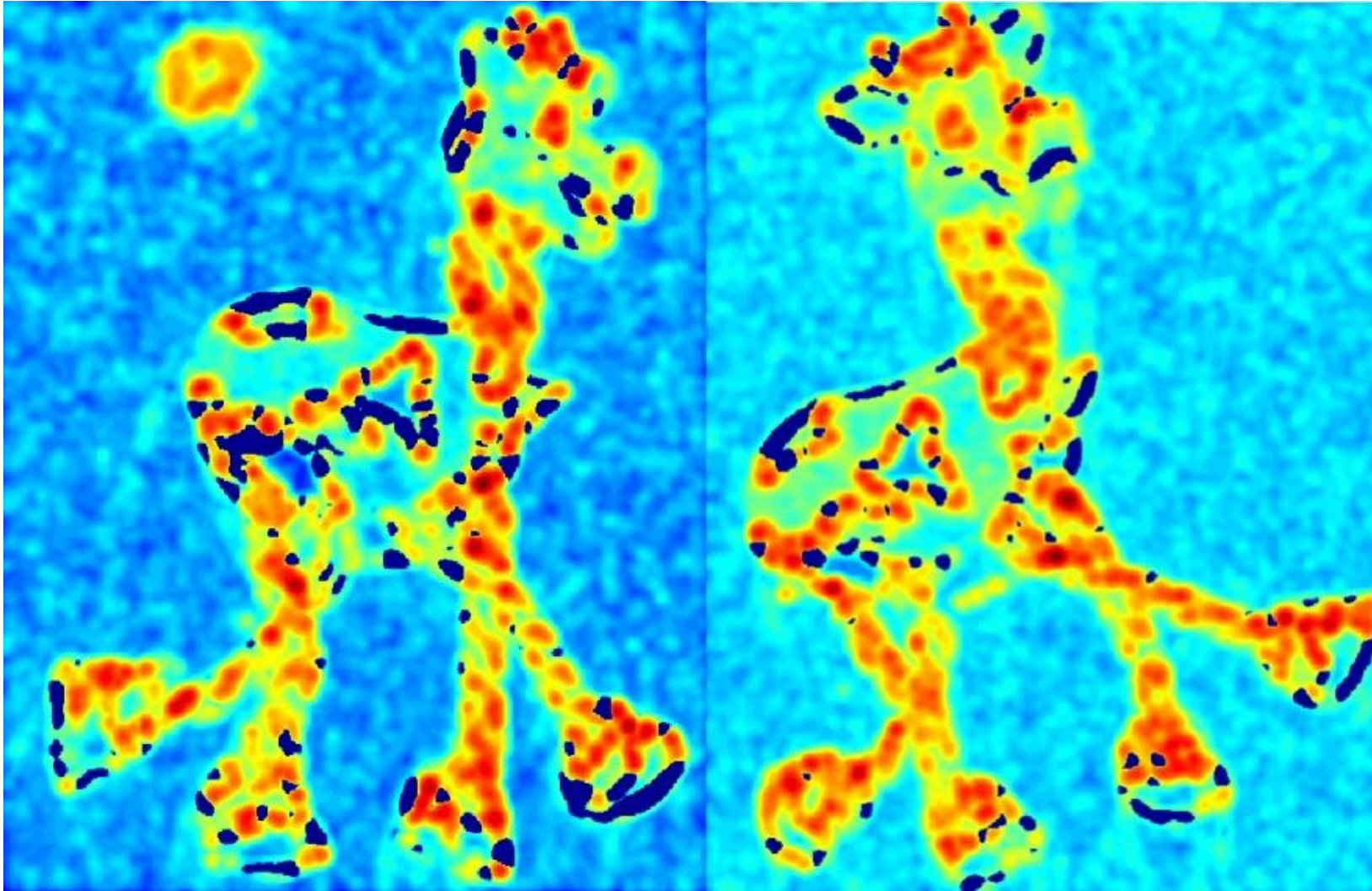


Gaussian

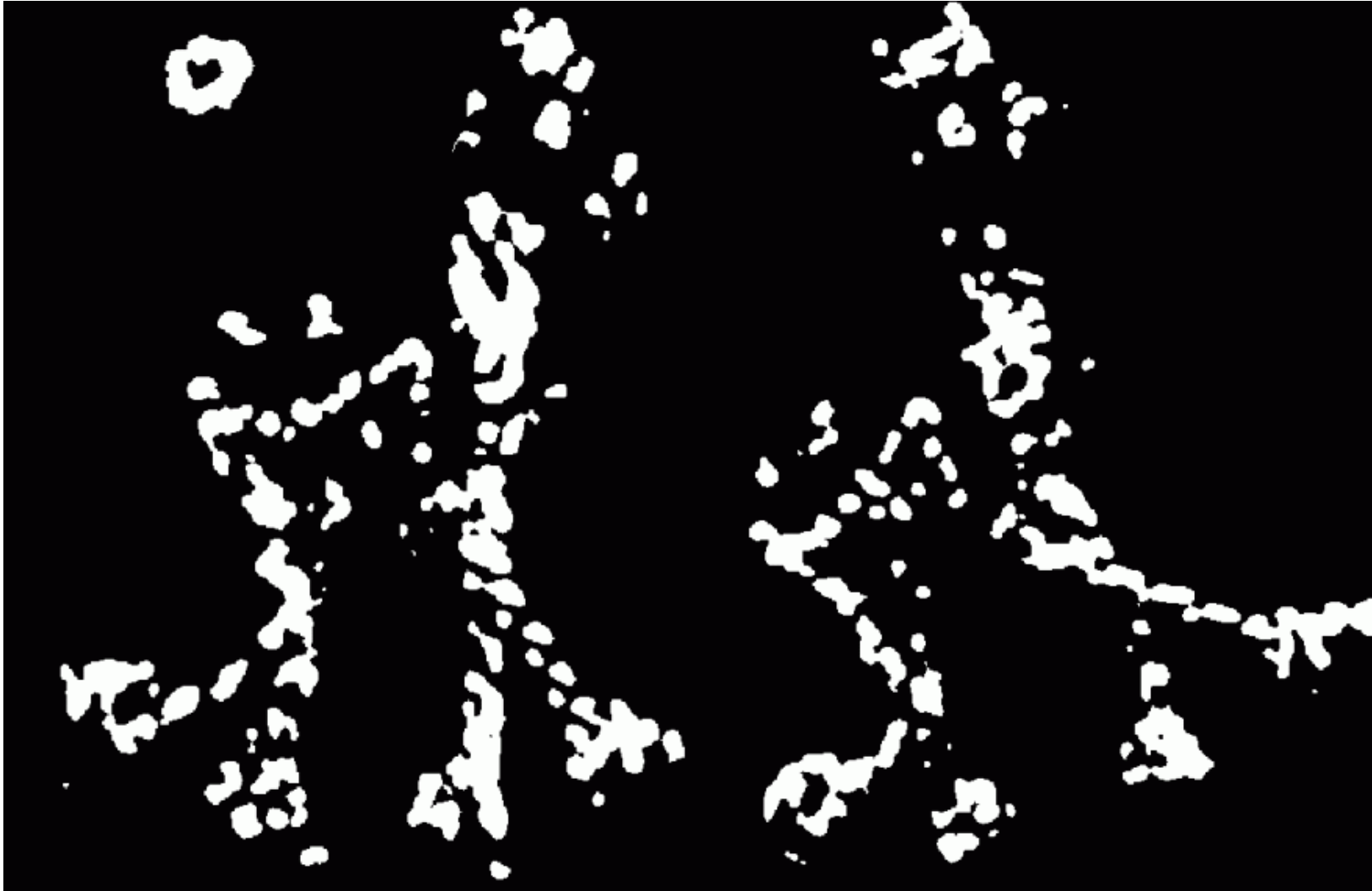
Harris Corner Detector



Harris Corner Detector



Harris Corner Detector



Harris Corner Detector

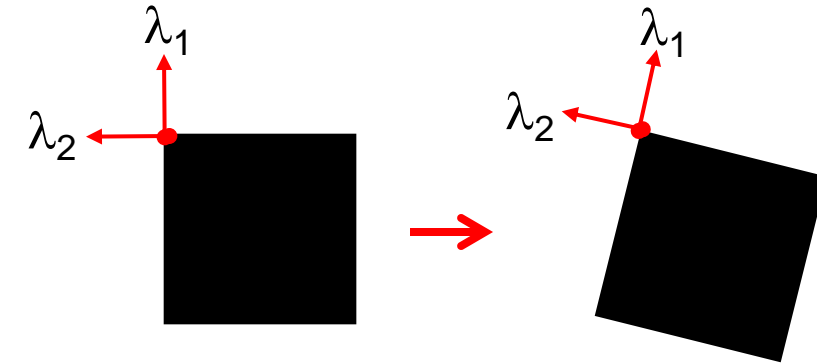


Harris Corner Detector

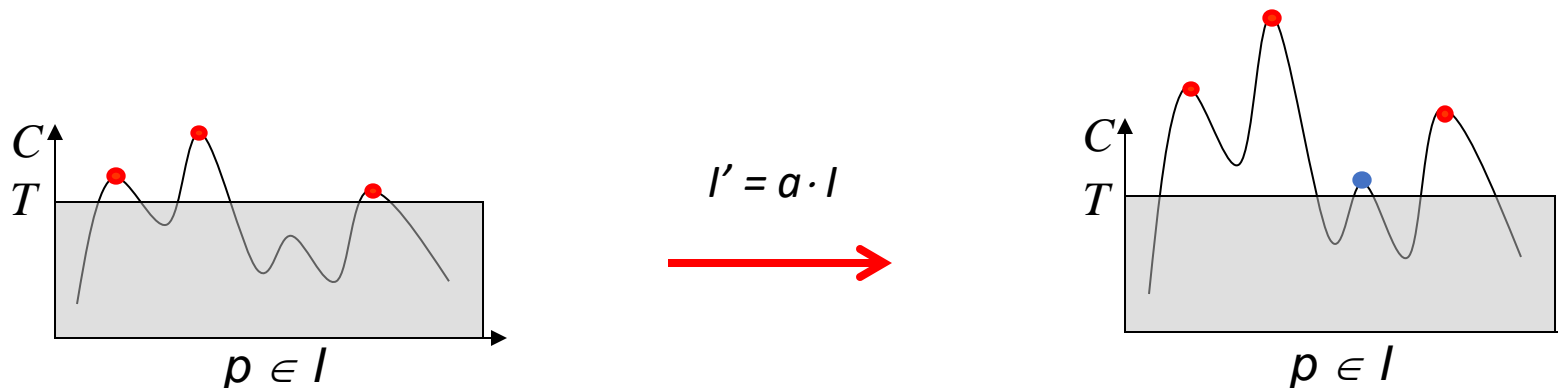


Invariance Properties

- Objects in images undergo changes (rotation , illumination, scale)
- **Rotation invariance**: eigenvalues of M are invariant to a rotation of the image axes and thus so is Harris cornerness function
- **No invariance to an affine intensity change**
 - **Yes**, for additive bias ($I' = I + b$) due to the use of derivatives
 - **No**, to multiplication by a gain factor ($I' = a \cdot I$) => derivatives get multiplied by the same factor

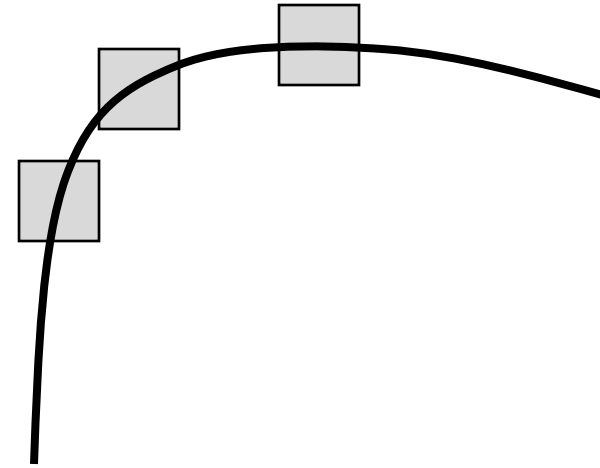


$$I' = I + b$$
$$I_x = I(i, j + 1) - I(i, j)$$
$$I'_x = I(i, j + 1) + b - I(i, j) - b$$



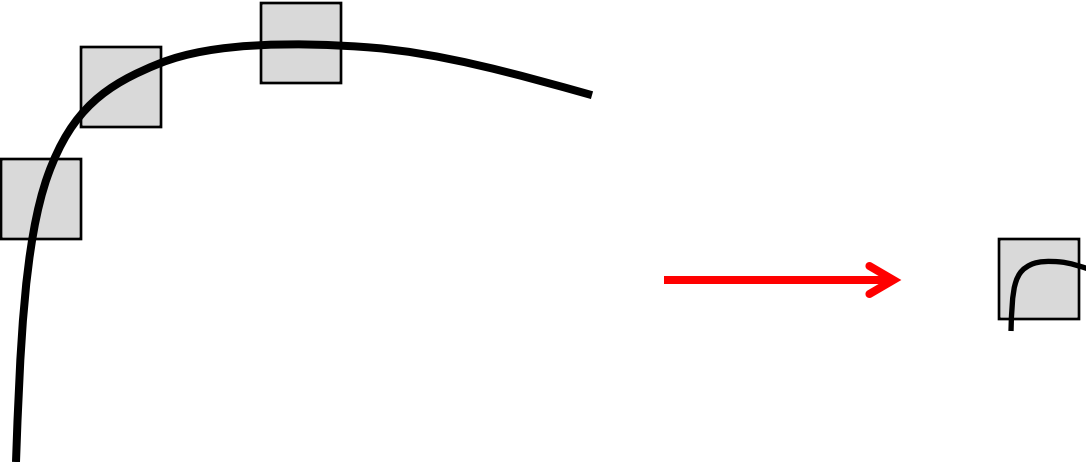
Invariance Properties

- Scale invariance?



Invariance Properties

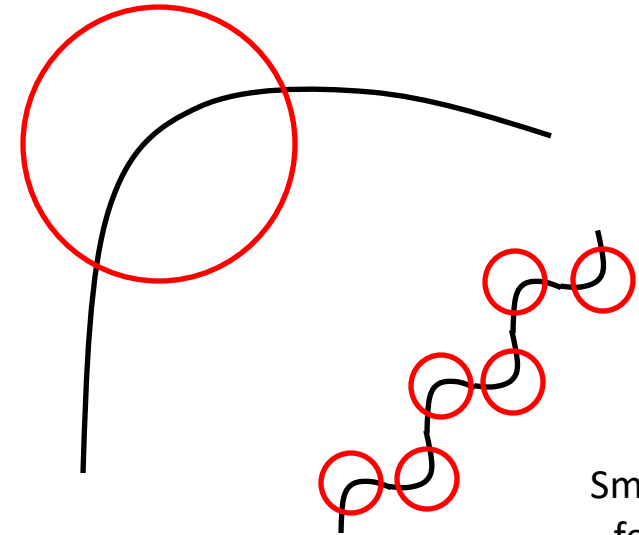
- Scale invariance => NO

- Given the chosen size of the detection window, all the points along the border are likely to be classified as edges
 - Should the object appear smaller in the image, use of the same window size would lead to detect a corner
- 
- The use of a fixed detection window size makes it impossible to repeatably detect homologous features when they appear at different scales in images
 - Harris is not scale invariant

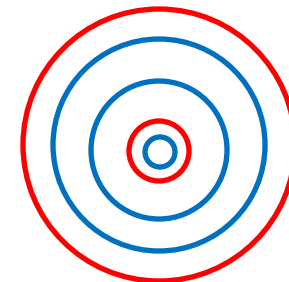
Invariance Properties

- An image contains features at different scales, i.e. points that stand-out as interesting as long as a proper neighbourhood size is chosen to evaluate the chosen interestingness criterion
- Detecting all features requires to analyze the image across the range of scales “deemed as relevant”
- The more features we gather the higher the chance to match across pictures
- What do we do?
 - Do we take more scale for that point? or
 - Do we look for the best scale for that point?

Large scale
features

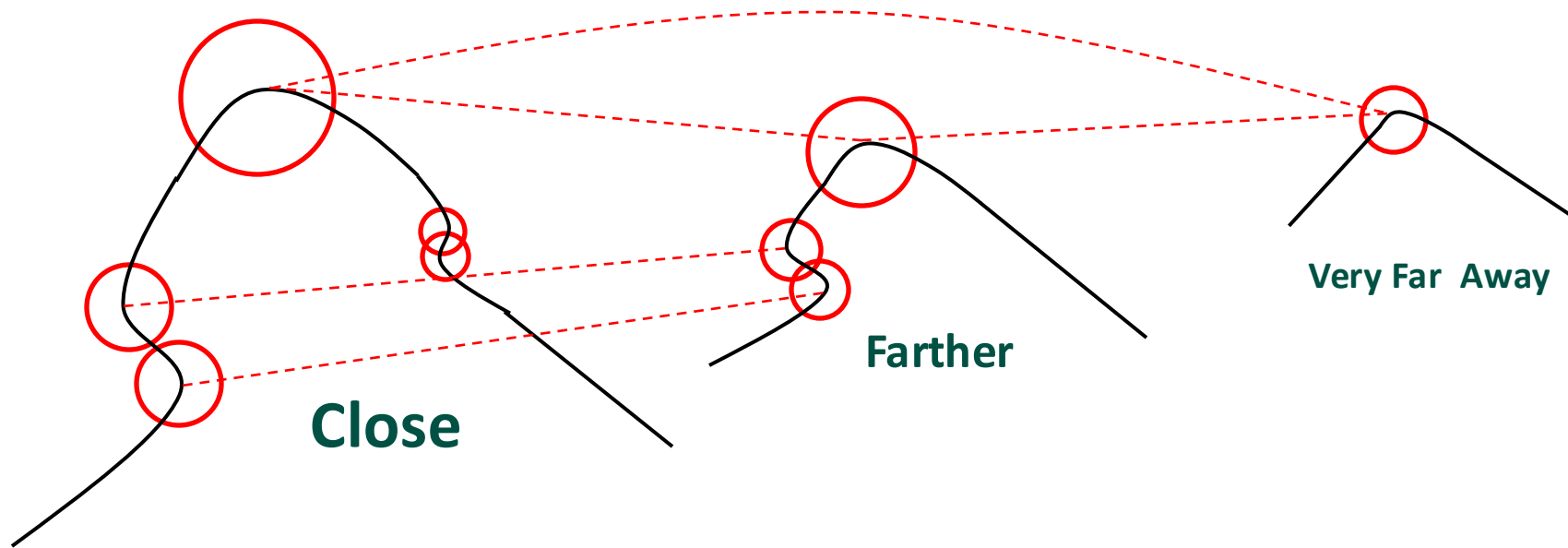


Small scale
features



Scale Invariance

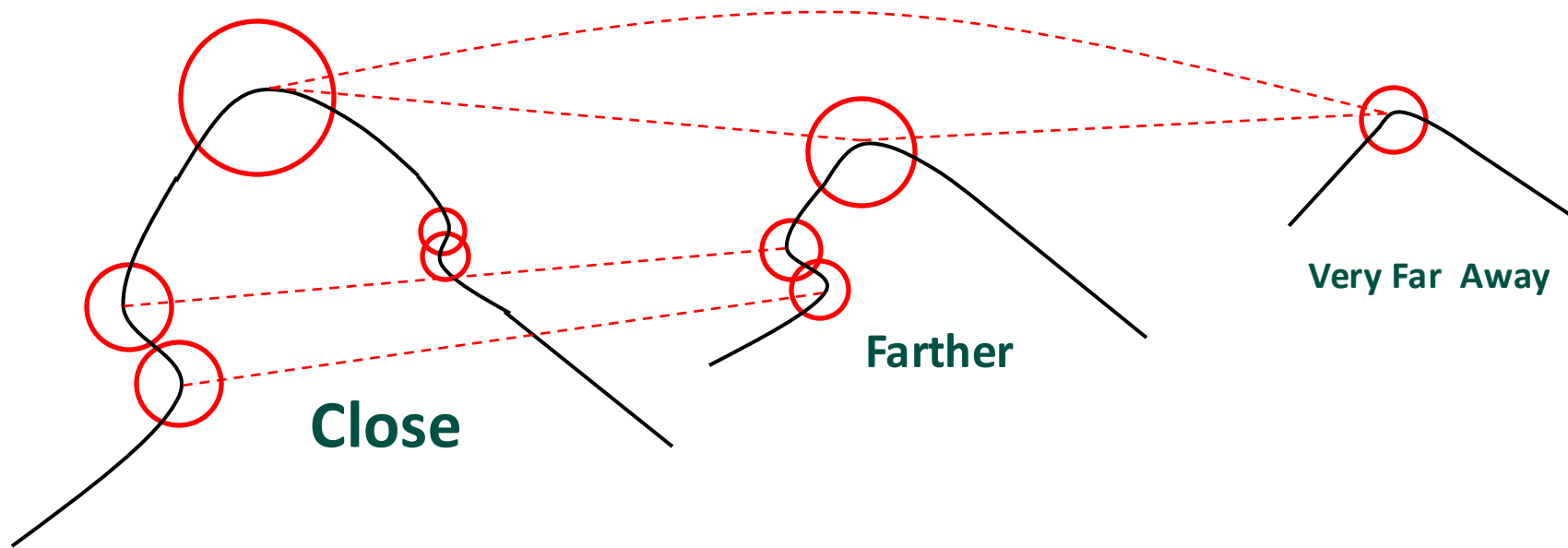
- Depending on the acquisition settings (distance and focal length) an object may look differently in the image, in particular it may exhibit more/less details (i.e. features)



- Features do exist within a certain range of scales
 - Finding similar features despite the different scales at which they may appear in different images
 - The same feature would simply be detected at different steps within a multi-scale image analysis process

Scale Invariance

- We detect features in order to *match* their *descriptors*.



- Yet, the neighbourhoods surrounding large scale features are far richer of details than those around small scale ones. The higher the scale the finer the details present in the neighbourhood of a feature.
- To make it possible to compute similar descriptors – and so match them- the details that do not appear across the range of scales should be cancelled-out by means of *image smoothing*.

Scale-Space

- Scale invariance is the main issue addressed by **second generation** local invariant features
- key finding: apply a **fixed-size detection** tool on increasingly **down-sampled** and **smoothed** versions of the input image:



Small Features



Larger Features



Very Large Features

We increase the blur
as we reduce the
image size

- When satisfying some specific mathematical properties, this representation is known as **Scale Space**
 - As you move along scales, small details should continuously disappear and no structure should be introduced

Gaussian Scale-Space

- A **Scale-Space** is a one-parameter family of images created from the original one so that the structures at smaller scales are successively suppressed by smoothing operations
 - one would not wish to create new structures while smoothing the images
- Several researchers [*] have studied the problem and shown that a **Scale-Space** must be realized by **Gaussian Smoothing**

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

- A Scale-Space is created by repeatedly smoothing the original image with larger and larger Gaussian kernels

*A. P. Witkin. "Scale-space filtering", *International Joint Conf. Artificial Intelligence*, 1983.

*J. Koenderink. "The structure of images", *Biological Cybernetics*, 50:363–370, 1984.

Feature Detection & Scale Selection

- The Gaussian Scale-Space is only a tool to represent the input image at different scales
 - it neither includes any criterion to detect features nor to select their *characteristic scale*
 - As features do exist across a range of scales...how do we establish at which scale a feature turns out maximally interesting and should therefore be described?
- The fundamental research work on *multi-scale feature detection* and *automatic scale selection* was proposed in [*] => compute suitable combinations of *scale-normalized derivatives* of the Gaussian Scale-Space (normalized Gaussian derivatives) and find their *extrema*
 - As we filter more (higher sigma) derivatives tends to become weaker
 - to compensate Lindberg proposes to multiply/normalize derivatives by sigma (scale-normalized)
 - We have stacked images => we look for extrema in a 3D space (pixel positions as well as scale)

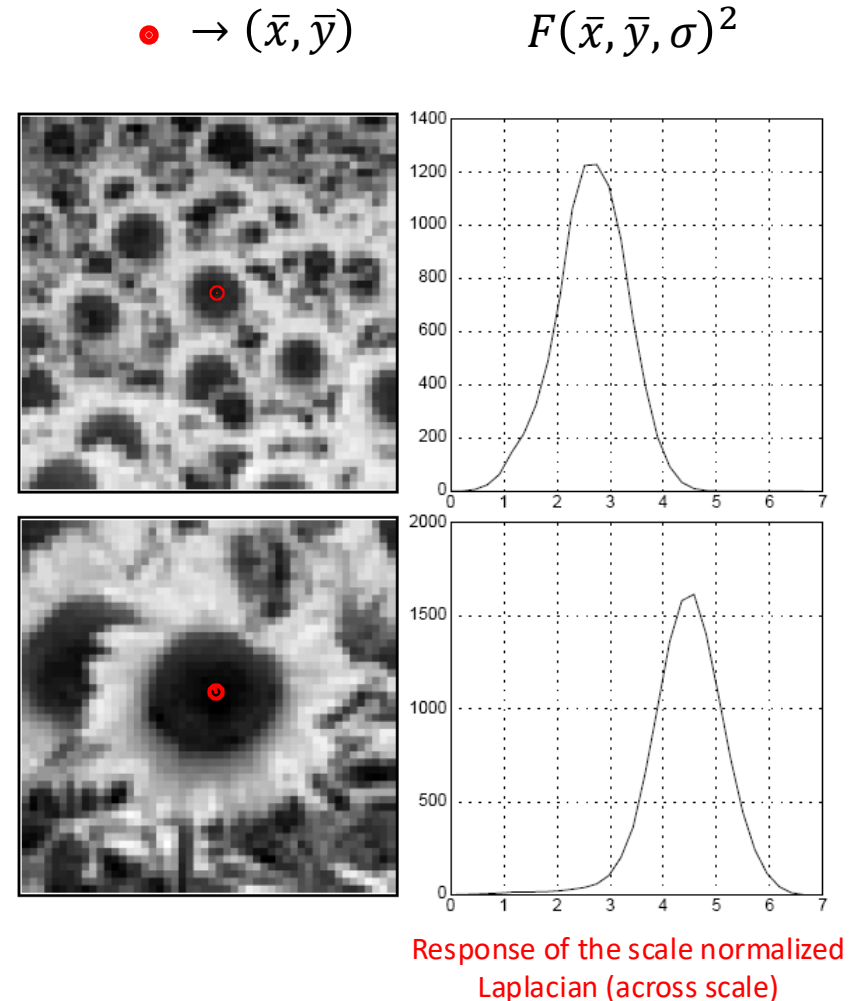
*T. Lindeberg. "Feature detection with automatic scale selection", *IJCV*, 1998

Scale-Normalized LOG

- Scale-normalized Laplacian of Gaussian (LOG)

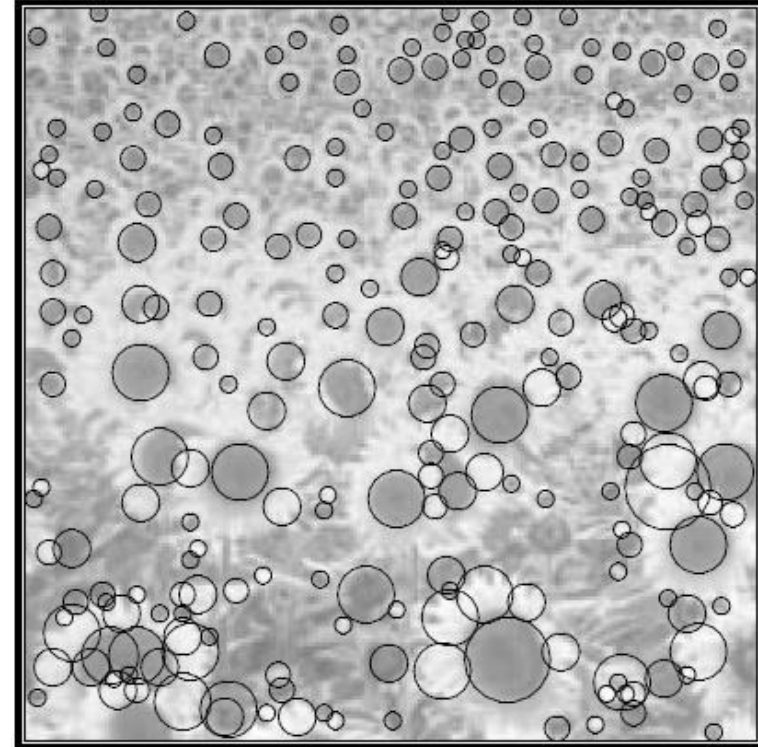
$$F(x, y, \sigma) = \sigma^2 \nabla^2 L(x, y, \sigma) = \underbrace{\sigma^2 (\nabla^2 G(x, y, \sigma) * I(x, y))}_{\text{normalization}}$$

- Considering the centers (red points) of the two dark blobs
 - => the extremum of the scale-normalized LOG is found at a larger scale for the larger blob
- The ratio between the two characteristic scales is roughly the same as the ratio between the sizes (diameters) of the two blobs
- The idea is to look for extrema across x, y, σ
 - => we have found a feature that has a position given by x and y and a scale given by the sigma at which it is maximum



Multi-Scale Feature Detection

- Features (**Blob-like**) and scales detected as extrema of the scale-normalized LOG
- LoG filter extrema locates “blobs”
 - maxima = dark blobs on light background
 - minima = light blobs on dark background



Difference of Gaussian (DoG)

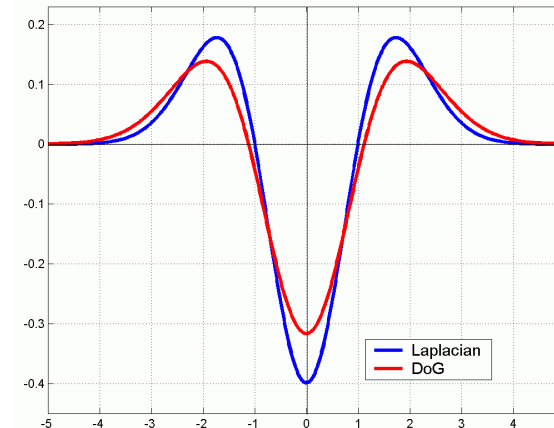
- Lowe [*] proposes to detect keypoints by seeking for the extrema of the DoG (Difference of Gaussian) function across the (x, y, σ) domain:

$$DoG(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = \overbrace{L(x, y, k\sigma)}^{\text{Gaussian scale-space}} - \overbrace{L(x, y, \sigma)}^{\text{Gaussian scale-space}}$$

- This approach provides a computationally efficient approximation of Lindeberg's scale-normalized LOG:

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx \underbrace{(k - 1)\sigma^2}_{\text{Scaled version of LoG}} \nabla^2 G(x, y, \sigma)$$

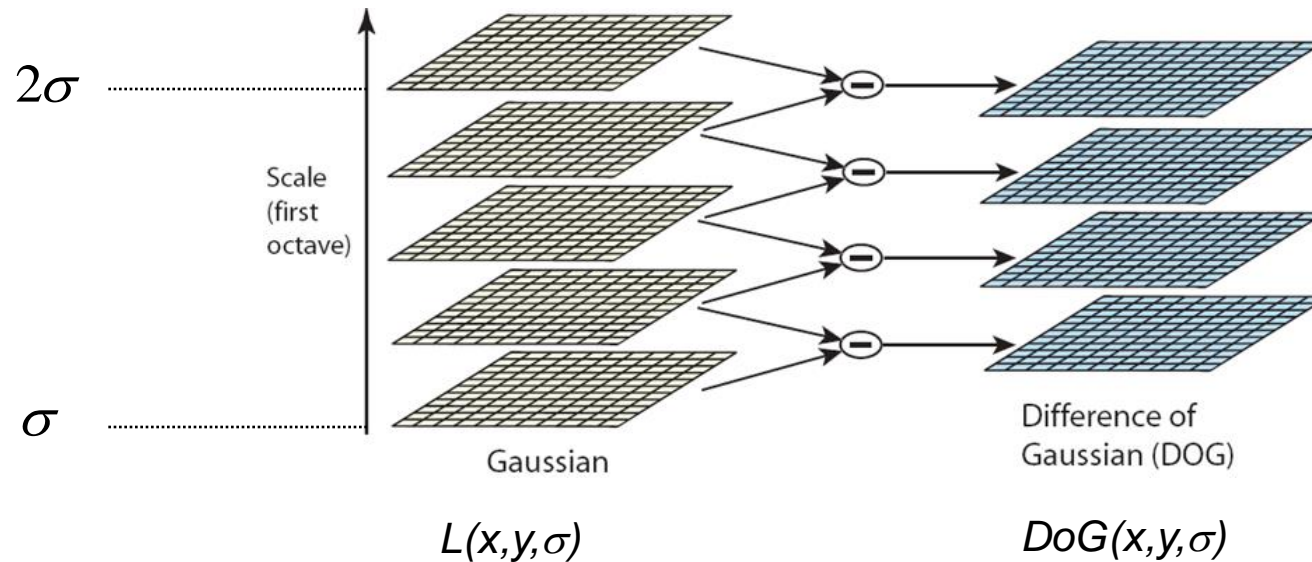
- Lowe proves that this is a scaled version of Lindberg
 - $(k-1)$ is a constant factor, it does not influence extrema location => the choice of k is not critical



- Both detectors are rotation invariant and find blob-like features (circularly symmetric filters)

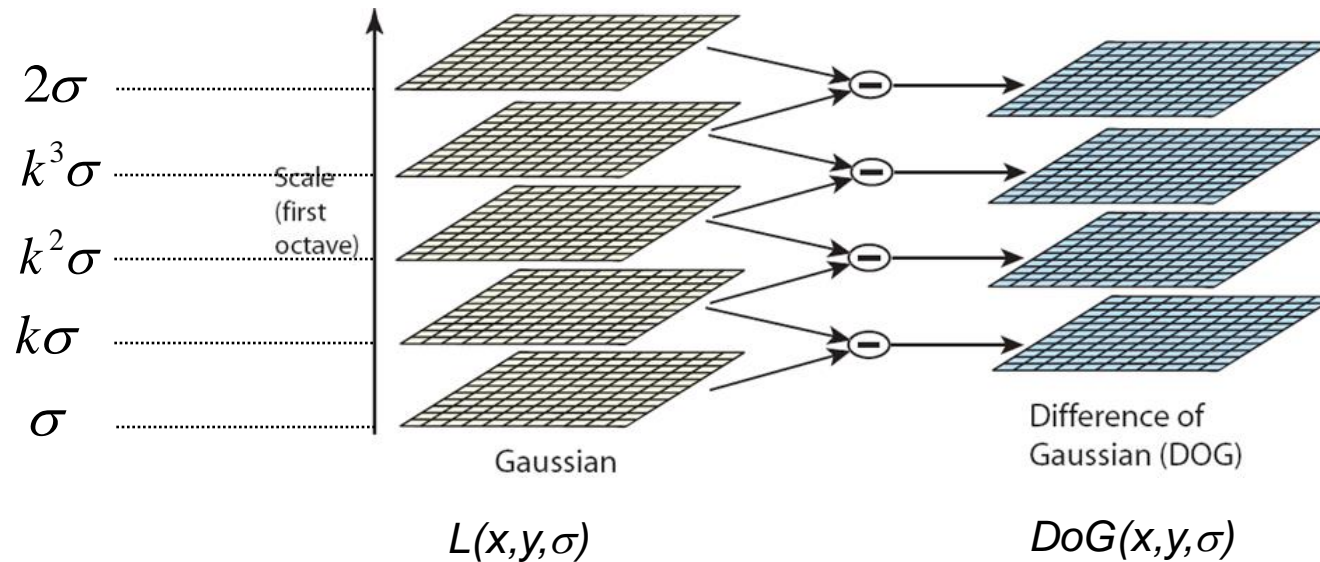
Difference of Gaussian (DoG)

- We compute several gaussian smoothed function within an octave
 - For each pair we take the differences => than we seek for extrema in the DoG



Difference of Gaussian (DoG)

- We compute several gaussian smoothed function within an octave
 - For each pair we take the differences => than we seek for extrema in the DoG



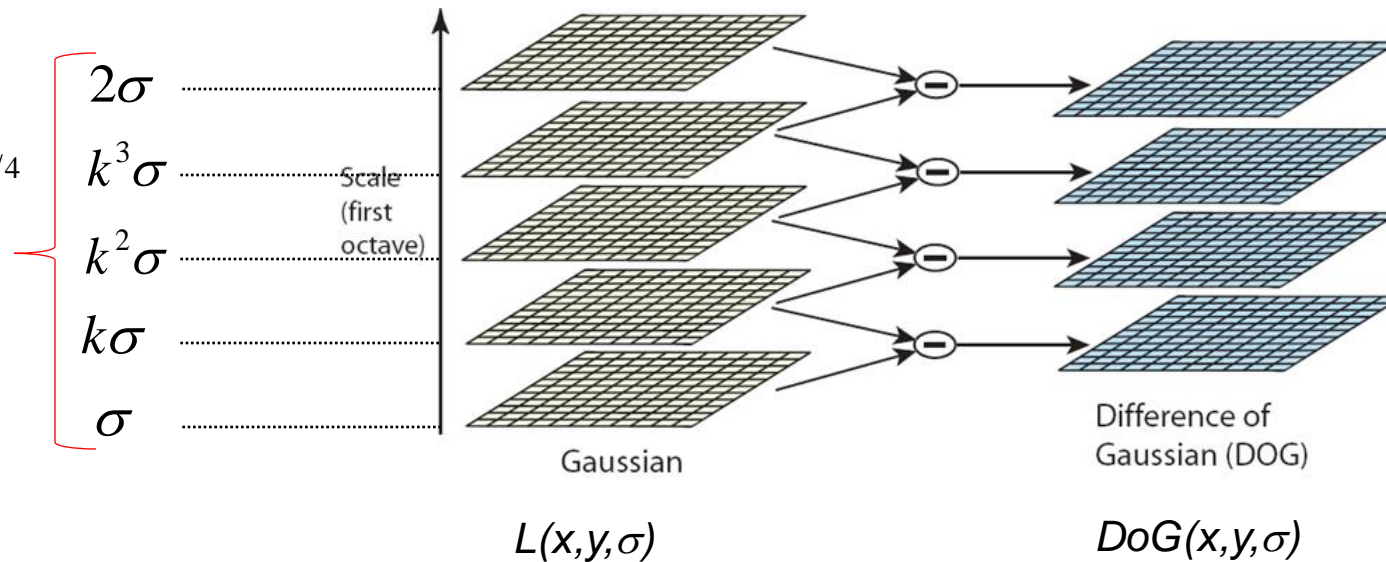
Difference of Gaussian (DoG)

- We compute several gaussian smoothed function within an octave
 - For each pair we take the differences => than we seek for extrema in the DoG

In an octave we
sample s scales:

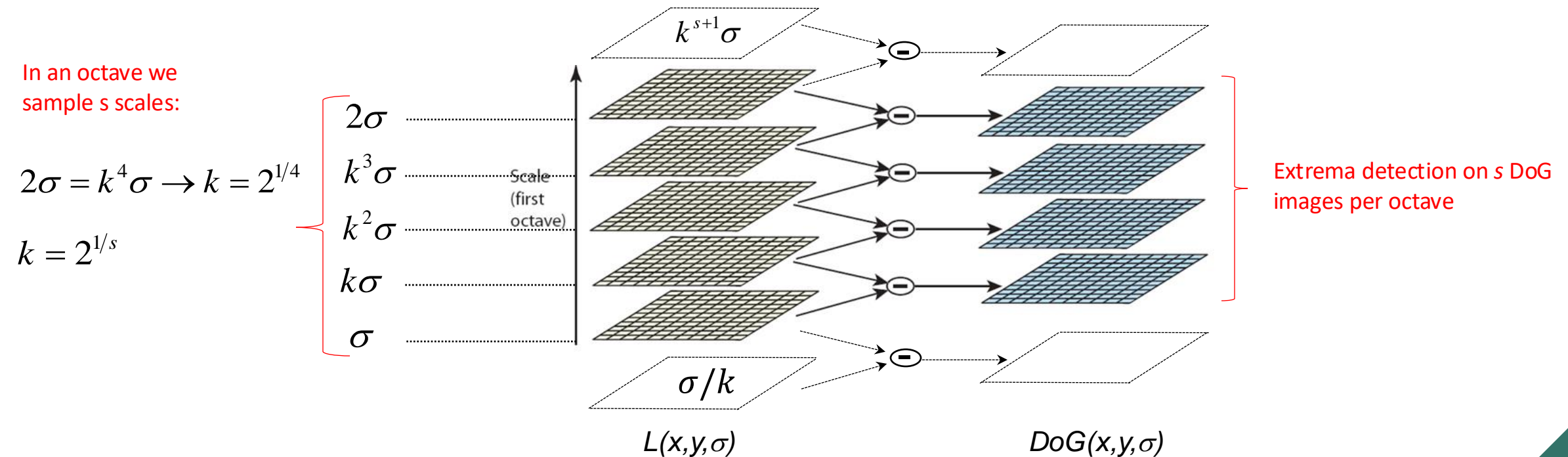
$$2\sigma = k^4 \sigma \rightarrow k = 2^{1/4}$$

$$k = 2^{1/s}$$

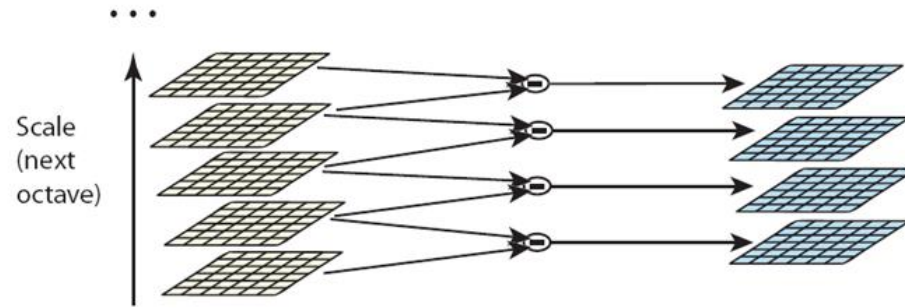


Difference of Gaussian (DoG)

- To compute the extrema on the blue one we need two additional Gaussian filtered layer



Difference of Gaussian (DoG)

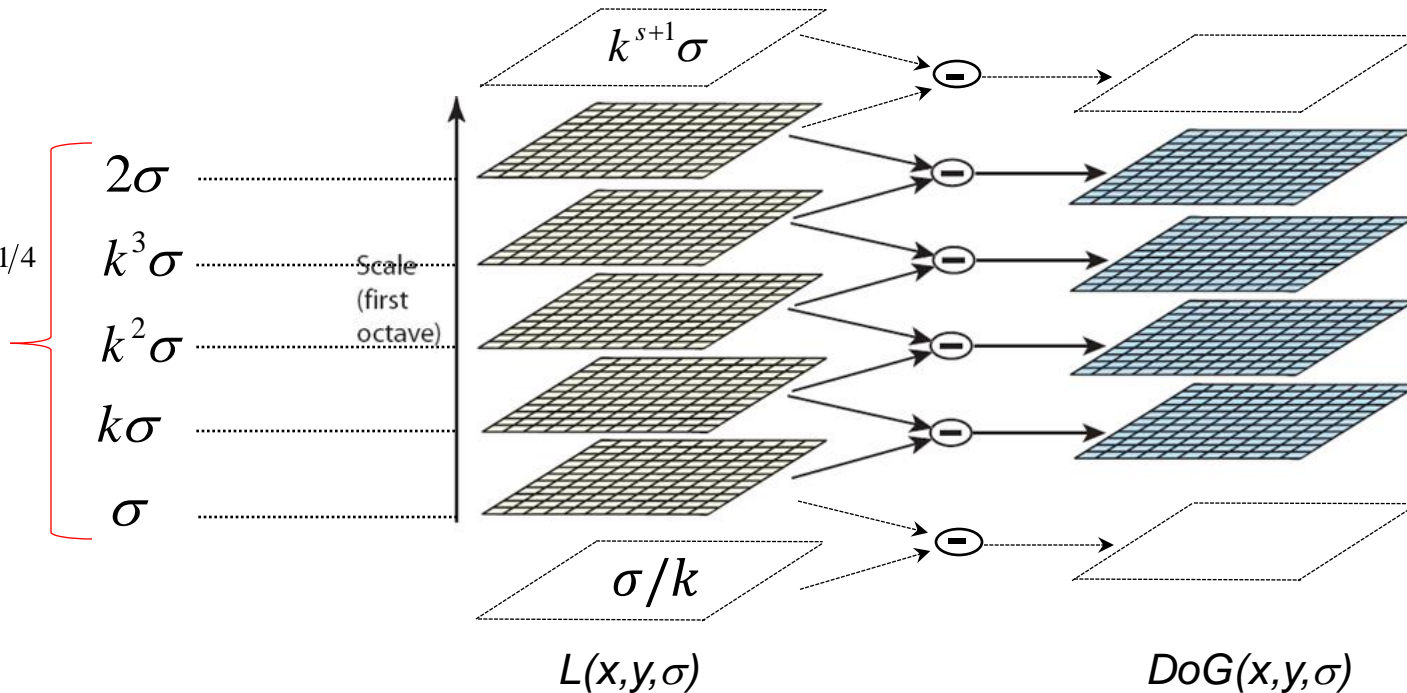


Rather than doubling σ we shrink the image taking half of the columns and half of the rows

In an octave we sample s scales:

$$2\sigma = k^4\sigma \rightarrow k = 2^{1/4}$$

$$k = 2^{1/s}$$

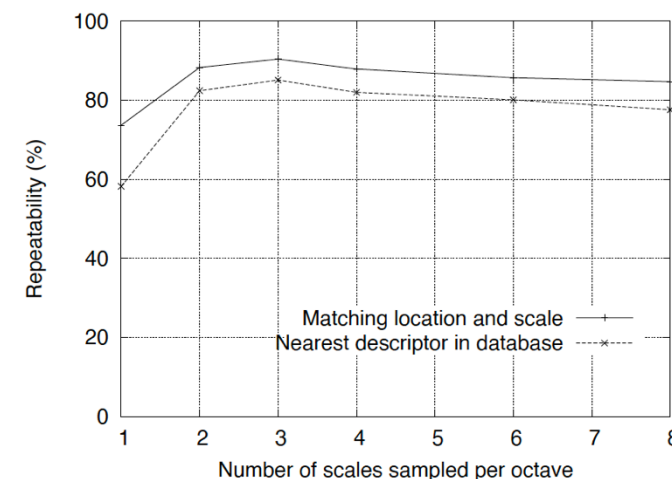
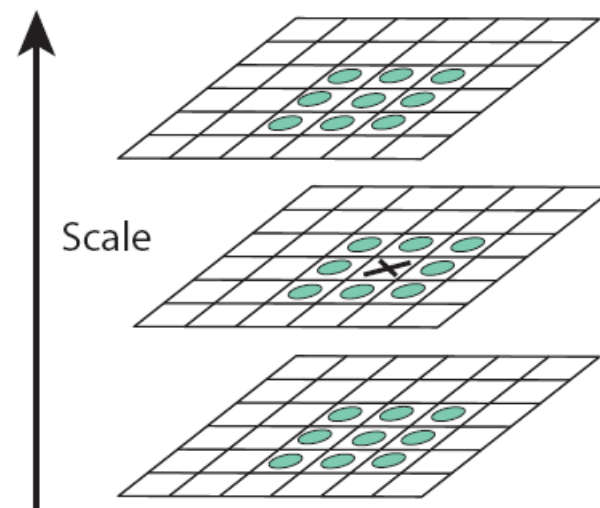


Extrema detection on s DoG images per octave

Keypoint Detection and Tuning

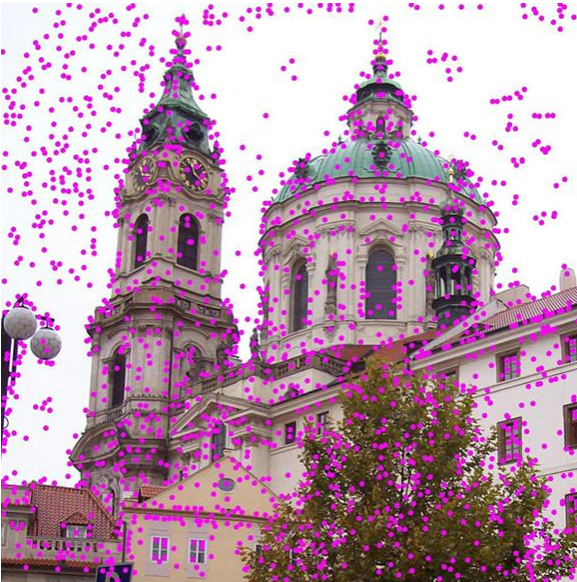
- Extrema detection: a point (x,y,σ) is detected as a keypoint iff its DoG is higher (lower) than that of the 26 neighbours (8 at the same scale and 18=9+9 at the two nearby scales) in the (x,y,σ) space
- According to the original article (parameter tuning)
 - the best number of scales within an octave is $s=3$
 - initial σ for each octave should be $\sigma = 1.6$
 - the input image is enlarged by a factor of 2 in both dimensions

DoG images

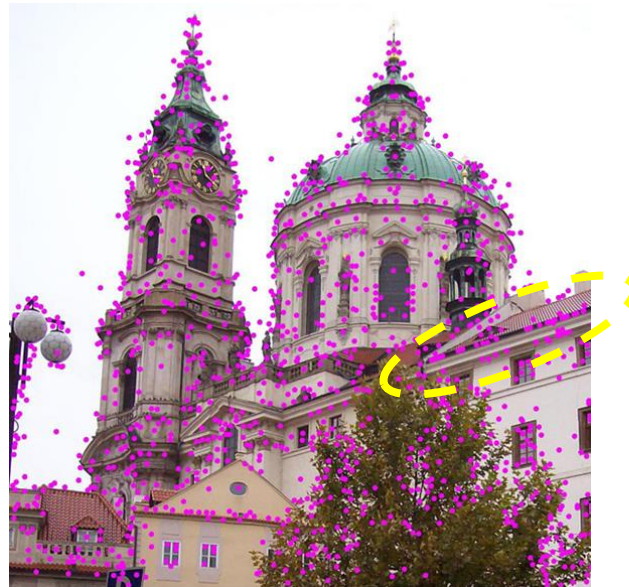


Exemplar DoG keypoints

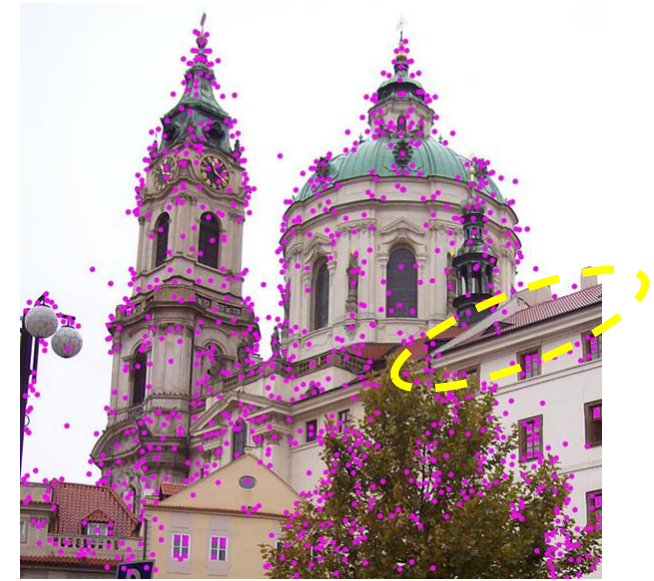
- Extrema featuring weak DoG response turn out scarcely repeatable...prune weak responses
- Lowe also notices that unstable keypoints featuring a sufficiently strong DoG may be found along edges and devises a further pruning step



DoG extrema



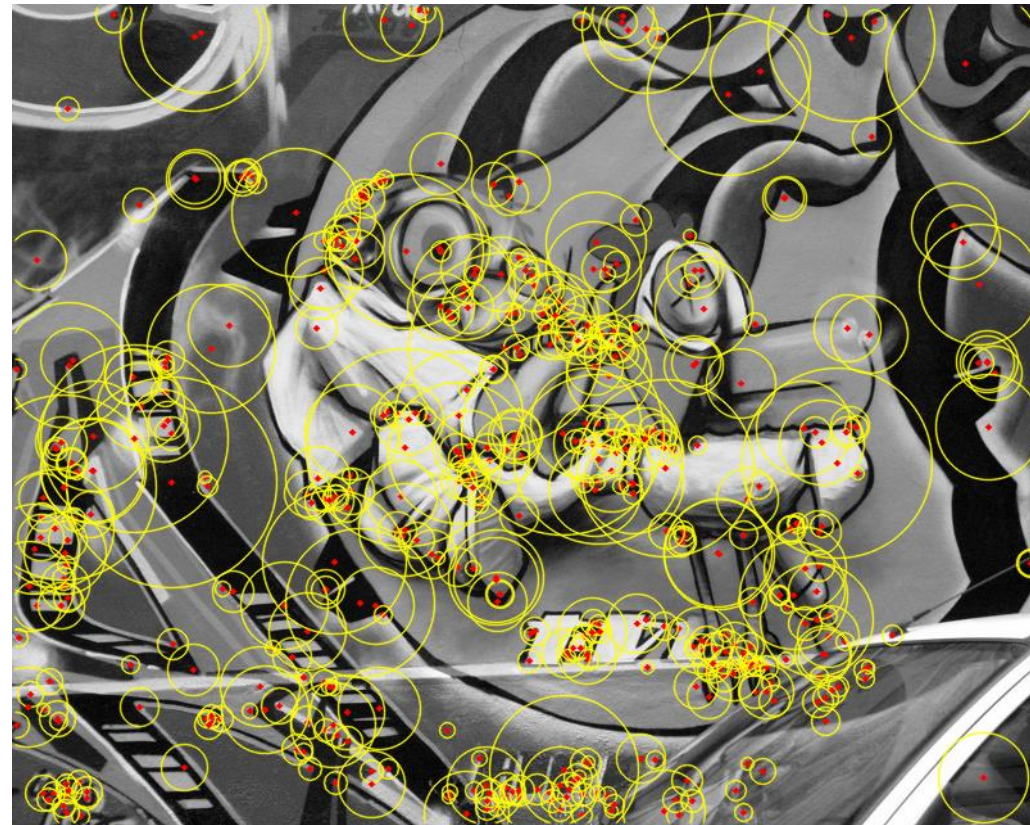
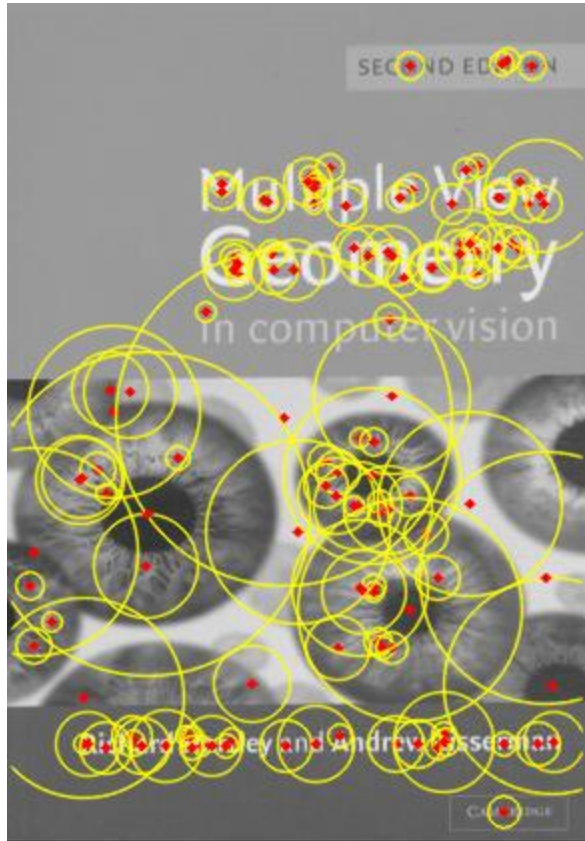
Keypoints after pruning weak responses



Final keypoints after pruning those located along edges

Exemplar DoG keypoints

The size of the circle is proportional to the scale of the feature (σ)

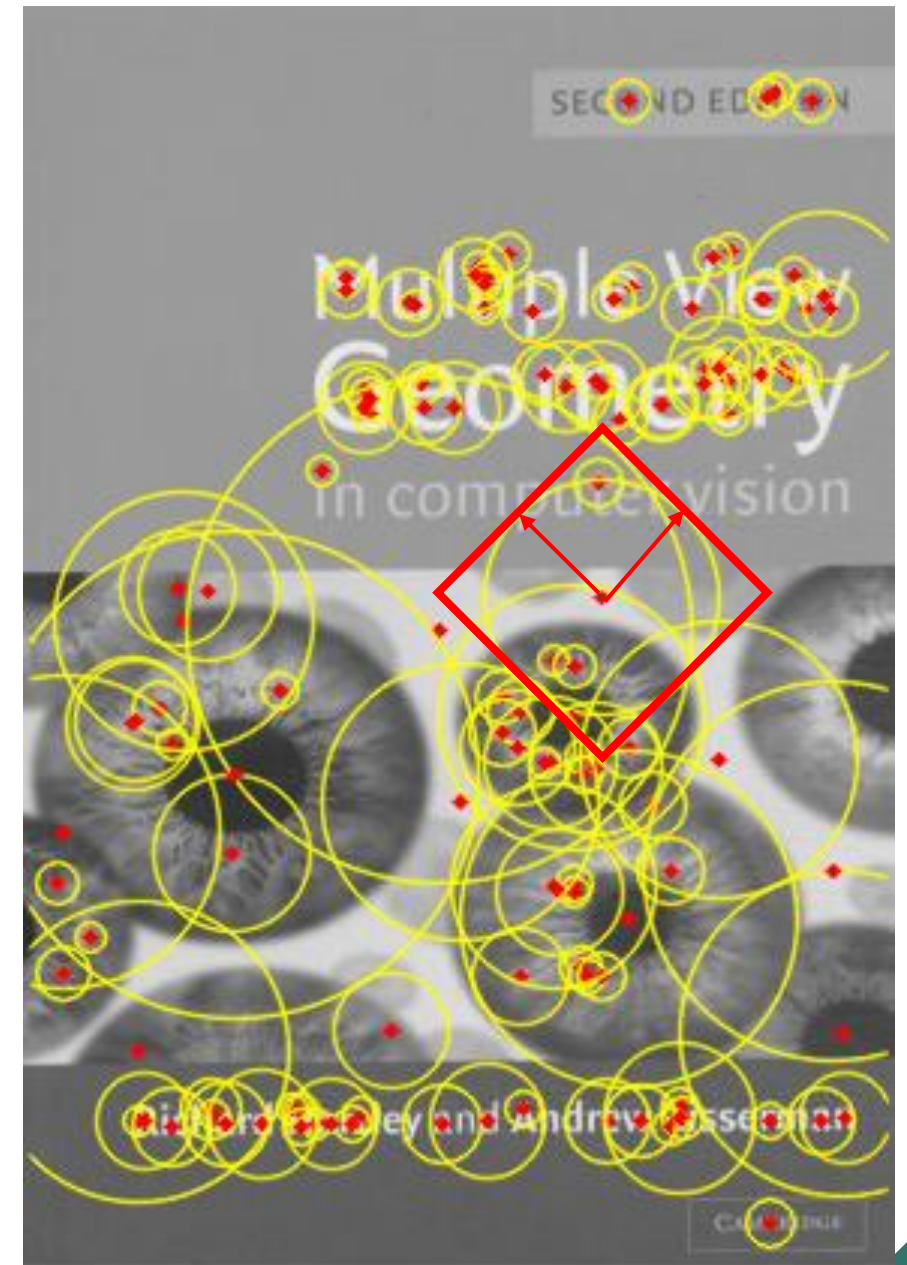
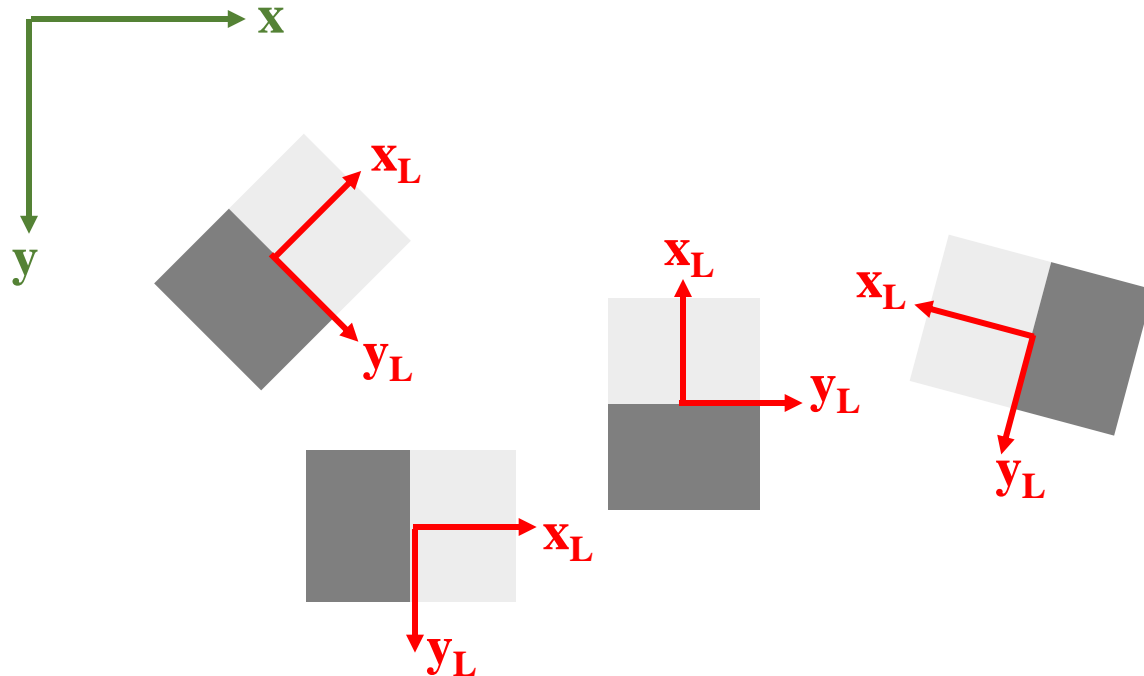


Scale and Rotation Invariance Description

- We need to define a Scale and Rotation Invariant Description
- Most keypoint detection algorithms follow an approach close to Lowe's one
 - i.e. finding extrema across a stack of images computed while increasing a scale parameter
- Once each keypoint has been extracted, a surrounding patch is considered to compute its descriptor (scale and rotation invariant)
 - Scale invariance: the patch is taken from the stack of images ($L(x, y, \sigma)$ in Lowe's) that correspond to the characteristic scale

Exemplar DoG keypoints

- We need to identify a prominent direction inherent to the patch (i.e., the **canonical orientation**) and, based on such direction, define a **local reference frame**. A reasonable choice consists in identifying the direction along which most of the gradient is found.
- When computing the descriptor, pixel coordinates are taken in the local reference frame.



Scale and Rotation Invariance Description

- We need to define a Scale and Rotation Invariant Description
- Most keypoint detection algorithms follow an approach close to Lowe's one
 - i.e. finding extrema across a stack of images computed while increasing a scale parameter
- Once each keypoint has been extracted, a surrounding patch is considered to compute its descriptor (scale and rotation invariant)
 - Scale invariance: the patch is taken from the stack of images ($L(x, y, \sigma)$ in Lowe's) that correspond to the characteristic scale
 - Rotation invariance: a canonical (aka characteristic) patch orientation is computed, so that the descriptor can then be computed on a ***canonically-oriented*** patch
 - Orientation wrt a new reference system, not the one of the image

Canonical Orientation

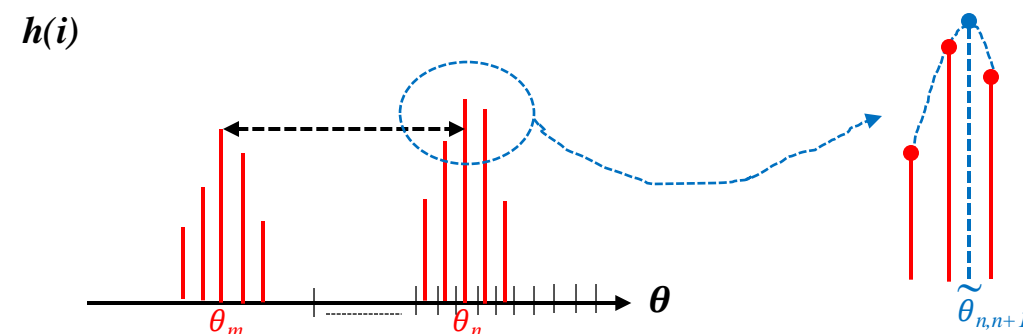
- Lowe proposes to compute the canonical orientation of DoG keypoints as follows:
 - Given the keypoint, the **magnitude** and **orientation** of the gradient are **computed at each pixel of the associated Gaussian-smoothed image, L**:

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

- Compute an **orientation histogram** by accumulating the contributions of the pixels belonging to a neighborhood of the keypoint location (bin size equal to 10°)
 - The contribution of each pixel to its designated orientation bin is given by the **gradient magnitude weighted by a Gaussian** with $\sigma = 1.5 \cdot \sigma_s$ (σ_s denotes the scale of the keypoint)
- The canonical direction will be the one along which most of the gradient is found (sense most of the change)

Canonical Orientation

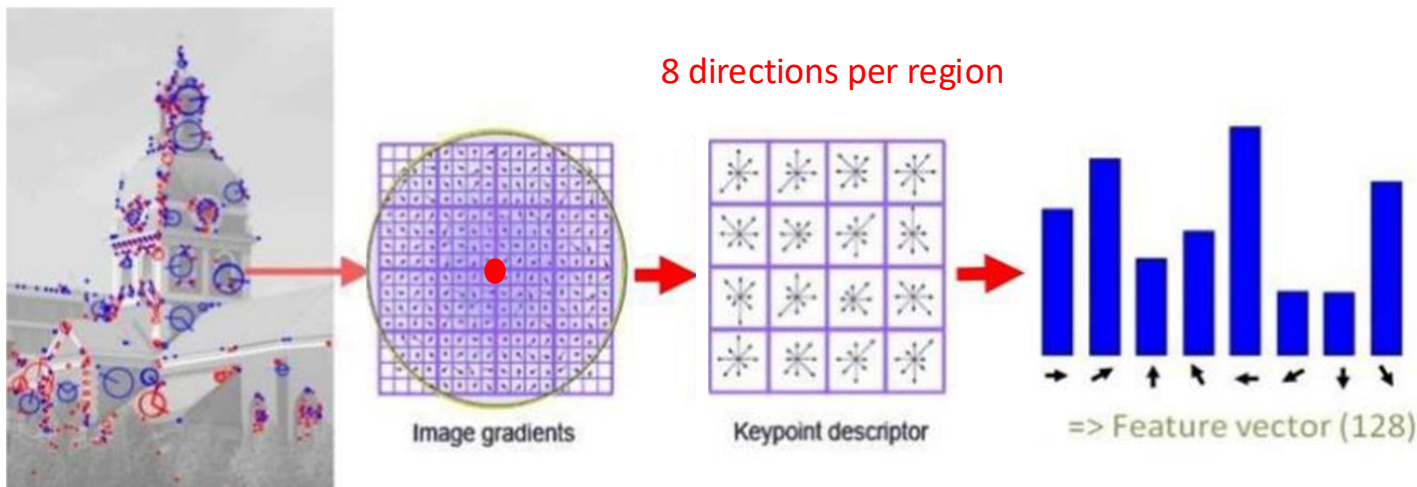
- The characteristic orientation of the keypoint is given by the highest peak of the orientation histogram
 - Other peaks higher than 80% of the main one would be kept as well
 - A keypoint may have multiple canonical orientations and, in turn, multiple descriptors sharing the same location/scale with diverse orientations
 - This has been found to occur quite rarely, about 15% of the keypoints
- Finally, a parabola is fit in the neighbourhood of each peak to achieve a more accurate estimation of the canonical orientation
 - The two bins adjacent to the found peak are considered



This is an ambiguous keypoint, we have two prominent directions

SIFT Descriptor

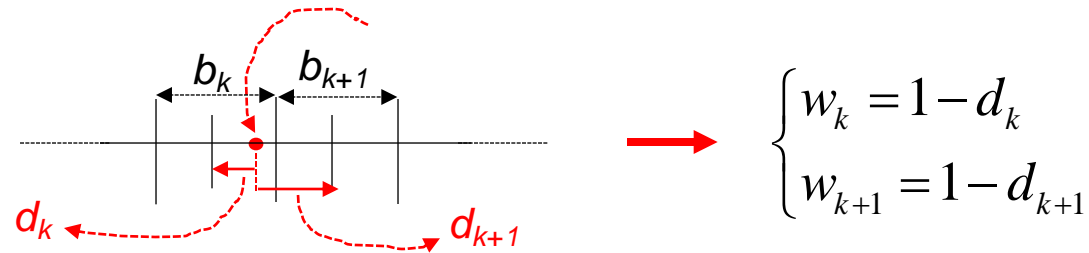
- The SIFT (Scale Invariant Feature Transform) descriptor is computed as follows:
 - A 16x16 **oriented** pixel grid around each keypoint is considered
 - This is further divided into 4x4 regions (each of size 4x4 pixels)
 - A gradient orientation histogram is created for each region
 - Each histogram has 8 bins (i.e. bin size 45°)
 - Each pixel in the region contributes to its designated bin according to
 - Gradient magnitude
 - Gaussian weighting function centred at the keypoint (with σ equal to half the grid size)



The descriptor size is given by the number of regions times the number of histogram bins per region, i.e.
 $4 \times 4 \times 8 = 128$
(histograms are concatenated)

SIFT Descriptor

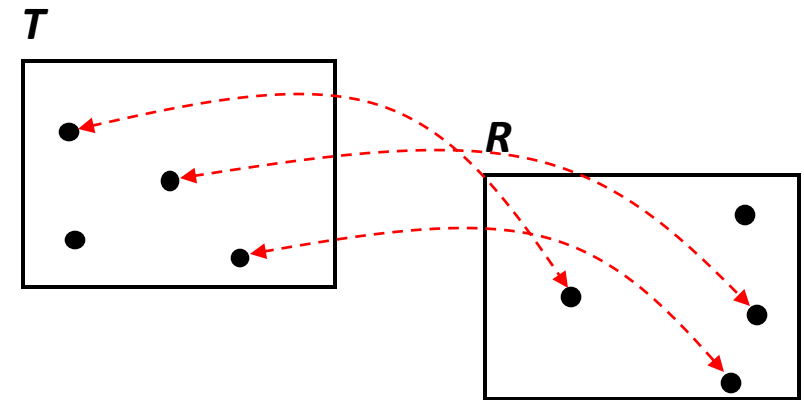
- SIFT is biologically inspired: there exist studies suggesting that neurons in the primary visual cortex (V1) do match gradient orientations robustly with respect to a certain degree of shift of the input pattern for recognition purposes
- To avoid boundary effects, a soft rather than hard assignment is employed in SIFT, whereby the contribution to two adjacent bins is weighted by the distance to the bin center:



- This is done within an histogram as well as between regions (the contribution is spread bilinearly between 4 adjacent regions). Hence, the overall scheme is referred to as ***trilinear interpolation***.
- The descriptor is normalized to unit length to gain invariance wrt affine intensity changes
 - To increase robustness to non-linear changes, all elements larger than 0.2 are saturated and the descriptor normalized again

Matching Process

- Descriptors (e.g. SIFT) are compared across diverse views of a scene to find corresponding keypoints
- This is a classical **Nearest Neighbour (NN) Search problem**:
 - Given a set S of points, p_i , in a metric space M and a query point $q \in M$, find the p_i closest to q .
- We wish to match the local features computed from an image under analysis (**target** image, T) to those already computed from a **reference** image (R) or a **set of reference images**
- For each feature in T we look for the most similar one in R :
 - The features in T represent the query points, q
 - The features in R provide set S
 - When matching SIFT descriptors the distance typically used is the **Euclidean distance**



Validating Matches

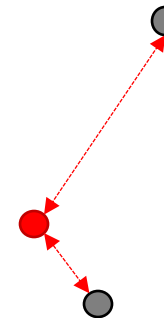
- The found NN does not necessarily provide a valid correspondence as some features in T may not have a corresponding feature in R (clutter and/or viewpoint changes)
- Enforce criteria to accept/reject a match found by the NN search process:
 - Simplest thing to do is to use a threshold

~~1) $d_{NN} \leq T$ (NN distance)~~

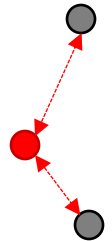
2) $\frac{d_{NN}}{d_{2-NN}} \leq T$ (ratio of distances)

- For the 2nd criteria (that is always less than 1) => the ratio should be small => the numerator must be small and the denominator must be large

Case 1



Case 2



Validating Matches

- The found NN does not necessarily provide a valid correspondence as some features in T may not have a corresponding feature in R (clutter and/or viewpoint changes)

- Enforce criteria to accept/reject a match found by the NN search process:

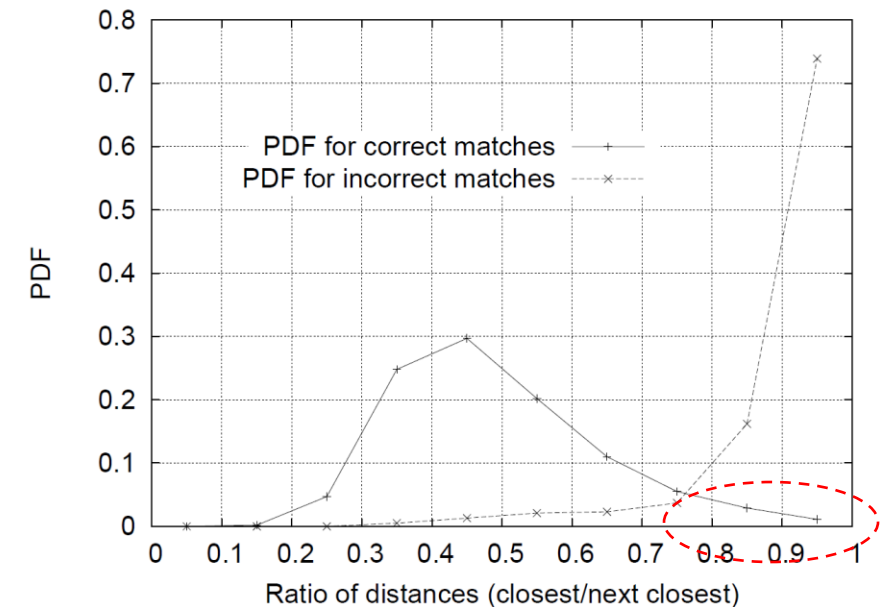
- Simplest thing to do is to use a threshold

~~1) $d_{NN} \leq T$ (NN distance)~~

2) $\frac{d_{NN}}{d_{2-NN}} \leq T$ (ratio of distances)

- For the 2nd criteria (that is always less than 1) => the ratio should be small => the numerator must be small and the denominator must be large

- Lowé shows that $T=0.8$ may allow for rejecting 90% of wrong matches while missing only 5% of those correct

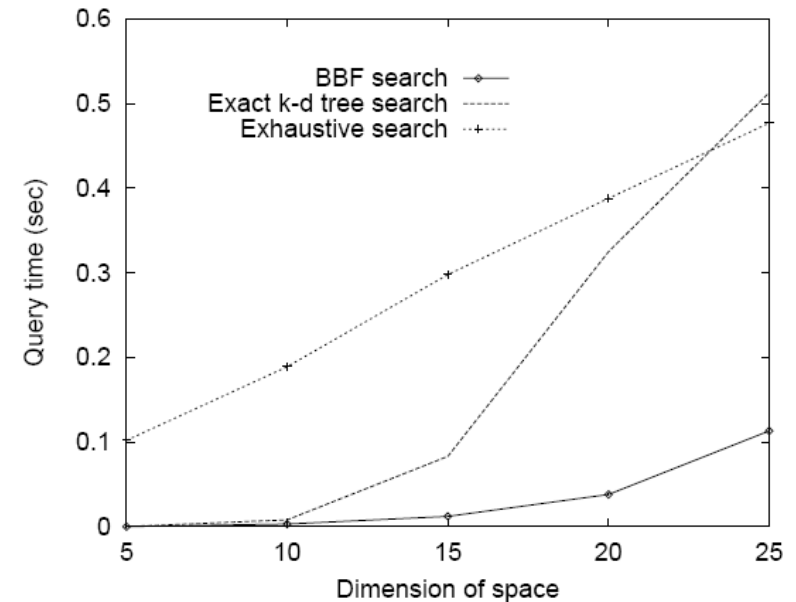


Efficient NN-Search

- Exhaustively searching for the NN of the query feature, q , has linear complexity in the size of S

This is slow!

- Efficient *indexing techniques* are exploited to speed-up the NN-search process
 - The main indexing technique exploited for feature matching is known as *k-d tree*, in particular, the preferred approach is the approximate variant referred to as *Best Bin First (BBF)*.
 - Unlike the basic k-d tree, the BBF is efficient also in high-dimensional spaces such as descriptor spaces



A few other major proposals

- A fast alternative to SIFT is **SURF** (Speeded-Up Robust Features) algorithm. Blob-like features are detected through efficiently computable filters inspired by Lindberg's Gaussian derivatives. The scale-space is achieved more efficiently by mean filtering
- **MSER** (Maximally Stable Extremal Regions) detects interest regions of arbitrary shapes, in particular approximately uniform areas either brighter or darker than their surroundings. Description of MSER region may then be carried out using SIFT
- **FAST** (Features from Accelerated Segment Test) is a very efficient detector of corner-like features.
- A variety of binary descriptors, such as **BRIEF**, **ORB** and **BRISK**, have been proposed to minimize memory occupancy and accelerate the matching step thanks to the use of the Hamming distance

References

- H. Moravec. “Rover visual obstacle avoidance”, International Joint Conference on Artificial Intelligence, 1981.
- C. Harris, M. Stephens. “A combined corner and edge detector”, Alvey Vision Conference, 1988.
- J. Shi, C. Tomasi (June 1994). "Good Features to Track". IEEE Computer Vision Pattern Recognition Conference (CVPR), 1994.
- A. P. Witkin. “Scale-space filtering”, International Joint Conf. Artificial Intelligence, 1983.
- J. Koenderink. “The structure of images”, Biological Cybernetics, 50:363–370, 1984.
- T. Lindeberg. “Feature detection with automatic scale selection”, International Journal of Computer Vision, 30(2):79–116, 1998.
- D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”, International Journal of Computer Vision, 20(2):91–110, 2004.
- J. Friedman, J. Bentley, and R. Finkel. “An algorithm for finding best matches in logarithmic expected time”, ACM Trans. Math. Software, 1977.
- J. Beis, D.G. Lowe. “Shape indexing using approximate nearest-neighbour search in high-dimensional spaces”, IEEE Computer Vision Pattern Recognition Conference (CVPR), 1997.
- H. Bay, T. Tuytelaars, and L. Van Gool. “Speeded-Up Robust Features (SURF)”, Computer Vision and Image Understanding, 2008.
- J. Matas, O. Chum, M. Urban, and T. Pajdla. “Robust wide baseline stereo from maximally stable extremal regions”, British Machine Vision Conference (BMVC), 2002.

References

- P.E. Forssen, D.G. Lowe “Shape Descriptors for Maximally Stable Extremal Regions “, IEEE Conference on Computer Vision (ICCV), 2007.
- E. Rosten, T. Drummond. “Machine learning for high-speed corner detection”, European Conference on Computer Vision (ECCV), 2006.
- M. Calonder, V. Lepetit, C. Strecha, P. Fua. “BRIEF: Binary Robust Independent Elementary Features”, European Conference on Computer Vision (ECCV), 2010.
- E. Rublee, V. Rabaud, K. Konolige, G. Bradski “ORB: an efficient alternative to SIFT or SURF”, IEEE Conference on Computer Vision (ICCV), 2011.
- S. Leutenegger, M. Chli, R. Y. Siegwart “BRISK: Binary Robust Invariant Scalable Keypoints”, IEEE Conference on Computer Vision (ICCV), 2011.
- F Tombari, A Franchi, L Di Stefano, “BOLD features to detect texture-less objects”, IEEE Conference on Computer Vision (ICCV), 2013.
- J. L. Schonberger, J.M. Frahm, “Structure-from-Motion Revisited”, IEEE Computer Vision Pattern Recognition Conference (CVPR), 2016.
- Marius Muja and David G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration", in International Conference on Computer Vision Theory and Applications (VISAPP'09), 2009.
- Jeff Johnson, Matthijs Douze, Hervé Jégou “Billion-scale similarity search with GPUs”, IEEE Transactions on Big Data, 2019