Image Processing and Computer Vision

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)

"cucire"

Panorama Stitching

in order to stitch together two different images, we need an HOMOGRAPHY:

- it is a math. transformation, a 3x3 matrix, H, which can map every point p in one image to the exact position of that point in the other image, p'

$$p' = H * p$$

- 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)

With at least four correspondences, you have eight equations which is the minimum required to solve for the eight unknown parameters of the homography matrix.

An homography involves 8 parameters | moreover, the transformation is invariant under scaling (homogeneous)

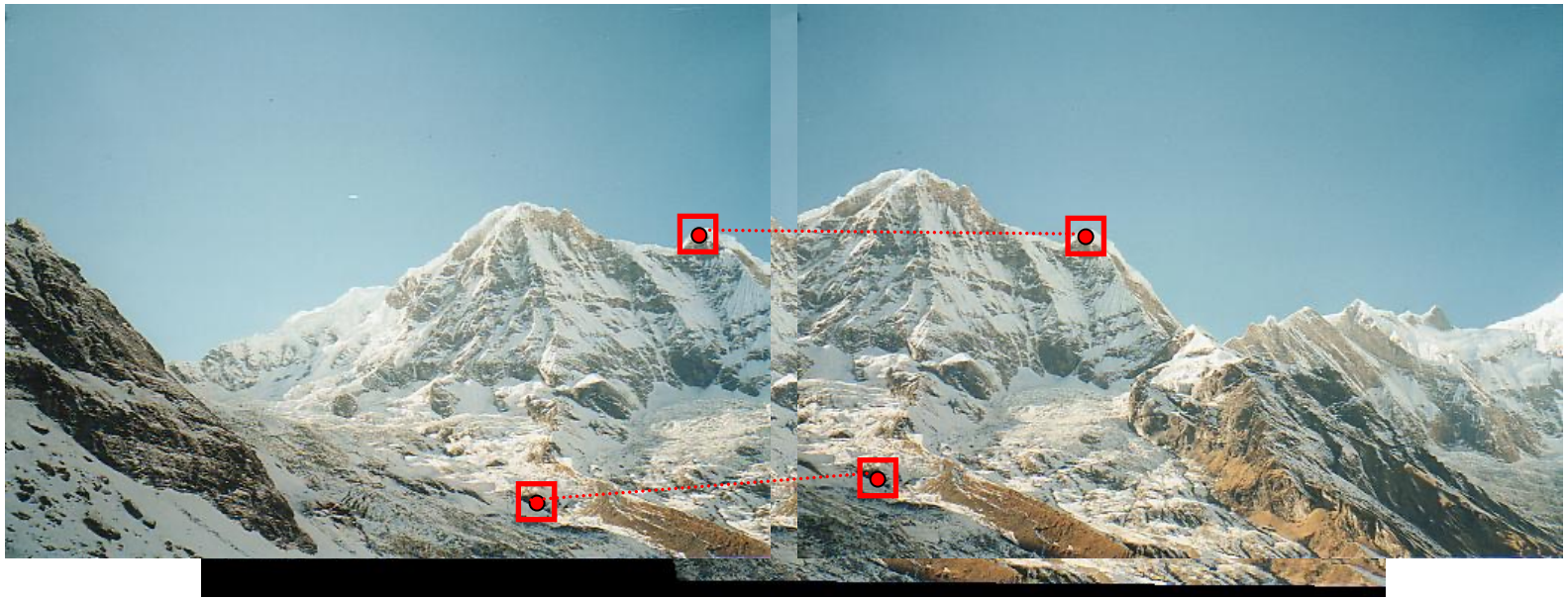
interesting

  - Find "salient points" independently in the two images
  - Compute a local "description" for each salient point
  - Compare descriptions to find matching pairs

maybe with euclidean distance (descriptors are vectors)

just intensity is not enough, we need a deeper description

the descriptor is a vector of numbers, which are properties/features computed from the neighbors of pixels



then we find correspondences computing the euclidean distance between these vectors

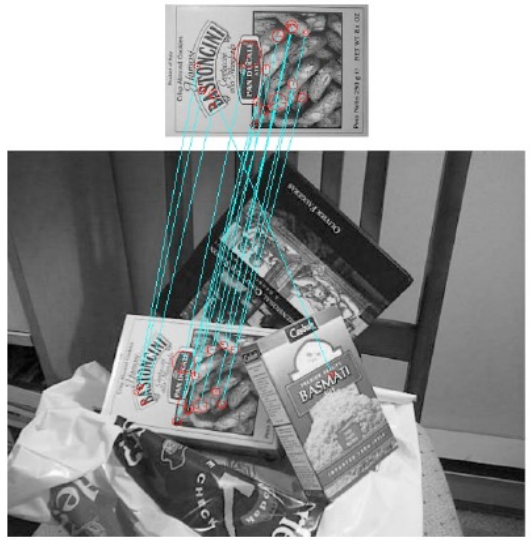
we will use salient points as "anchor points" to stitch the images

Local Invariant Feature

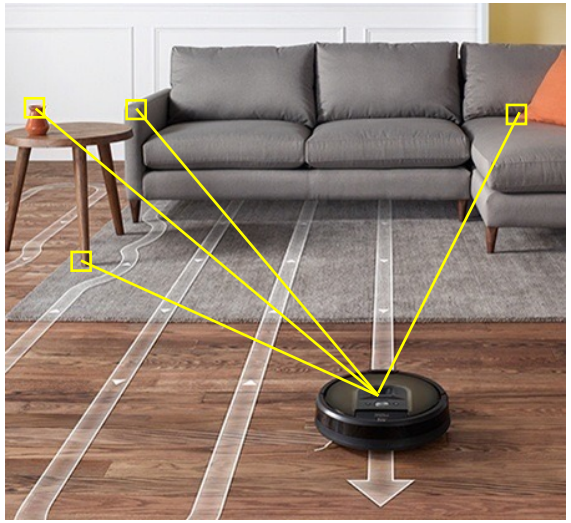
we need to extract information, namely features

- 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

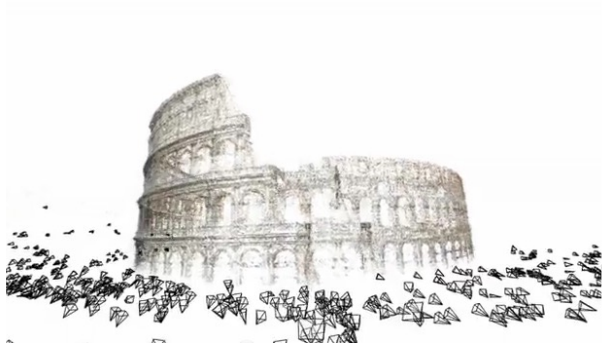


Robot Navigation



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

scaling, viewpoint changes, rotations, changes of intensity, ...

Taj Mahal

match

Properties of good detectors/descriptors

- Detector of salient points
  - Repeatability: it should find the same keypoints in different views of the scene despite the transformations undergone by the images repeatability of patterns in both images
  - Saliency: it should find keypoints surrounded by informative patterns (good for making them discriminative for the matching) information, like the contours of the objects and the neighborhood should be informative enough
- Descriptor of the (detected) salient points
  - 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 avoid features affected by noise
  - Compactness: the description should be as concise as possible, to minimize memory occupancy and allow for efficient matching

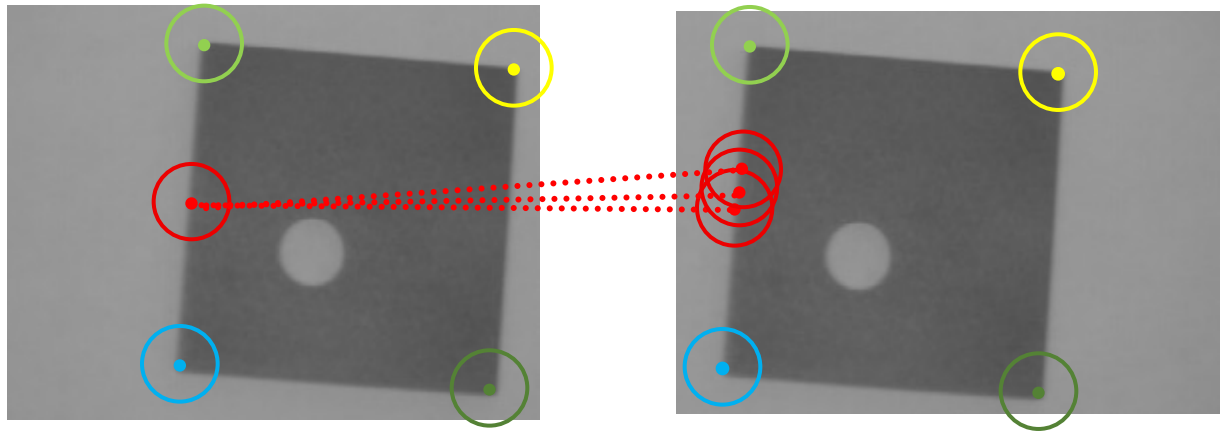
- Speed is desirable for both, and in particular for detectors, which need to be run on the whole image (while decriptors are computed at keypoints only)

Interest Points vs. Edges

edges aren't the best choice for interesting salient points in an image

corners, quite the opposite, are a very good choice for saliency

- 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



we cannot take the gradient as distinction factor (in order to search for a match) because it is ambiguous: for all of that red points the gradient is the same

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

the first feature we search in order to find a correspondance are CORNERS

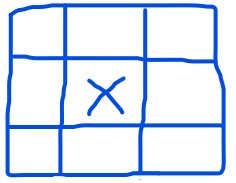
property of corner: a lot of changes in intensity around a corner, in all directions

Moravec Interest Point Detector

CORNER DETECTOR

it identifies these regions by analyzing how much the local image patch changes when shifted slightly in different directions, focusing on finding locations where all shifts result in high SSD values, suggesting significant intensity changes in multiple directions.

window/patch dimension
3x3, 5x5, 7x7, ...



1) The image is typically divided into a grid of small windows (patches) of fixed size. Each window represents a local area around a potential interest point.

2) For each window, the detector considers its pixel values. The window is then virtually shifted by a small amount in different directions: it essentially means analyzing the local neighborhood around the potential corner point

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

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

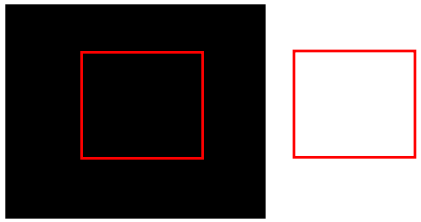
3) The sum of these squared differences (SSD) is computed for each shift direction. This represents the total change in intensity within the window after the shift.

4) The rationale is that if the minimum SSD is high, then shifting the window in any direction significantly alters the local image content, indicating a corner-like structure where intensity changes drastically in multiple directions.

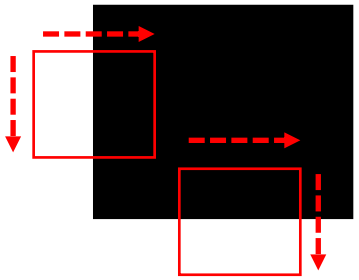
uniform region: no change in all directions

edge: no change along the edge direction

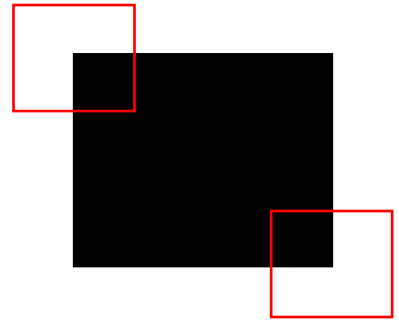
corner: significant change in all directions.



C: cornerness
C is small



C is small



C is high above a certain threshold

- After computing the cornerness => threshold and then NMS

it's the simplest solution, but it has some limitations: it is invariant only to a few transformations

H. Moravec. “Rover visual obstacle avoidance”, International Joint Conference on Artificial Intelligence, 1981

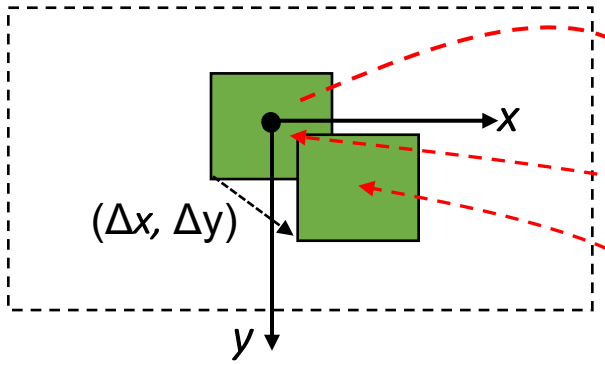
C. Harris, M. Stephens. “A combined corner and edge detector”, *Alvey Vision Conference*, 1988

Harris Corner Detector

it's the most useful detector, mostly used for camera calibration

harris extends moravec's formulation in a continuous way, providing a more advanced approach, building upon the Moravec idea by considering the change in SSD with respect to direction more directly, leading to improved robustness and corner localization.

- 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 (Δx, Δy): it doesn't consider anymore discrete shifts analyzing its local neighborhood
  - 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



$$w(x,y) = \begin{cases} 1 \rightarrow \blacksquare \\ 0 \rightarrow \square \end{cases}$$

it controls the size of the neighborhood and weights

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

weight function

This function essentially calculates the SSD within a window after shifting it by (delta_x, delta_y).

Remember:
By taking derivatives at a specific point, the Taylor series builds a polynomial expression that captures the local behavior of the function around that point.
The Taylor expansion provides a mathematical foundation for analyzing how local intensity variations within the window change with small shifts

- Due to the shift being infinitesimal, we can deploy Taylor’s expansion of the intensity function at (x,y):

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

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

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

I_x I_y

Moravec detector is limited because it yet considers the local neighborhood but in a LIMITED way, shifting the window by a FIXED amount in specific directions

this means that it doesn't explicitly analyze the intensity variations within the window itself, but only the changes introduced by the shifts.

Therefore, Harris uses a WINDOW function w(x,y) to explicitly include the neighborhood in the analysis

Harris Corner Detector

1) The image is typically divided into a grid of small windows (patches) of fixed size. Each window represents a local area around a potential corner point

2) For each window, the detector calculates the derivatives of the image intensity in the horizontal (Ix) and vertical (Iy) directions. These derivatives capture the local change in intensity within the window. (they can be computed using finite methods like sobel filters or directly computing them)

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

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

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

3) A 2x2 matrix called the autocorrelation matrix or STRUCTURE matrix, is constructed for each window. This matrix summarizes the local intensity variations within the window around a potential corner point

- Each element of the matrix represents the sum of squared products of the image derivatives within the window

M 2x2

Sum of squared horizontal derivatives

Sum of products of horizontal and vertical derivatives

Sum of products of horizontal and vertical derivatives

Sum of squared vertical derivatives

1x1 1x2 2x1

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

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

weighted sum of derivatives around the point

N.B.

A high value in M(1,2) and M (2,1) (namely the entries of the anti-diagonal) suggests that the intensity changes in both directions are correlated, potentially indicating a corner-like structure.

Harris Corner Detector

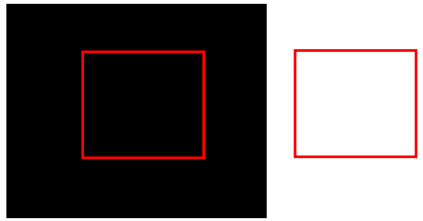
the eigenvalues represents all the information regarding that matrix (if we assume it is diagonal, the eigen values are the diagonal entries of the matrix)

- *M* encodes the local image structure around the considered pixel.
- Let us *hypothesize* that M is a diagonal matrix: $M = \begin{pmatrix} \sum_{x,y} w(x,y) I_x(x,y)^2 & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y(x,y)^2 \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$

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

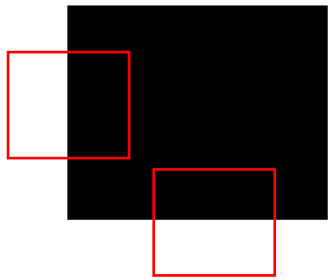
they both increase

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

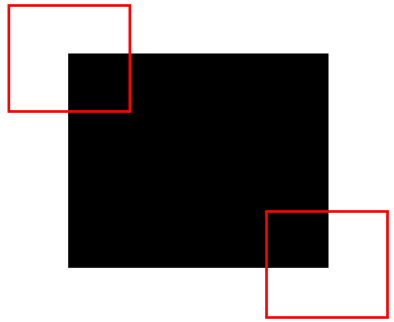


$\lambda_1 \gg \lambda_2$: Edge

I sense changes only in one direction (vert/horiz)



$\lambda_1, \lambda_2 \uparrow$: Corner



Harris Corner Detector

what if edges are diagonal?

eiegnvalues do not change despite the rotation in an image, because I can decompose the rotated image into the non-rotated image itself and a rotation matrix (namely, diagonalized with a rotation matrix)

- 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

The rotation matrix (R) captures the orientation of the principal components of the intensity variations within the window, telling us about the orientation of the dominant intensity variations.

the eigenvalues tells us about the magnitude of the intensity changes along the principal components

Diagonalization involves finding a transformation (rotation in this case) that converts the matrix into a diagonal form, where all non-zero entries lie on the main diagonal.

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

eigenvalue decomposition

- 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!

A high eigenvalue indicates a significant intensity change in that direction.

By aligning the axes with the eigenvectors, the diagonal matrix containing the eigenvalues (lambda1 and lambda2) becomes particularly informative.

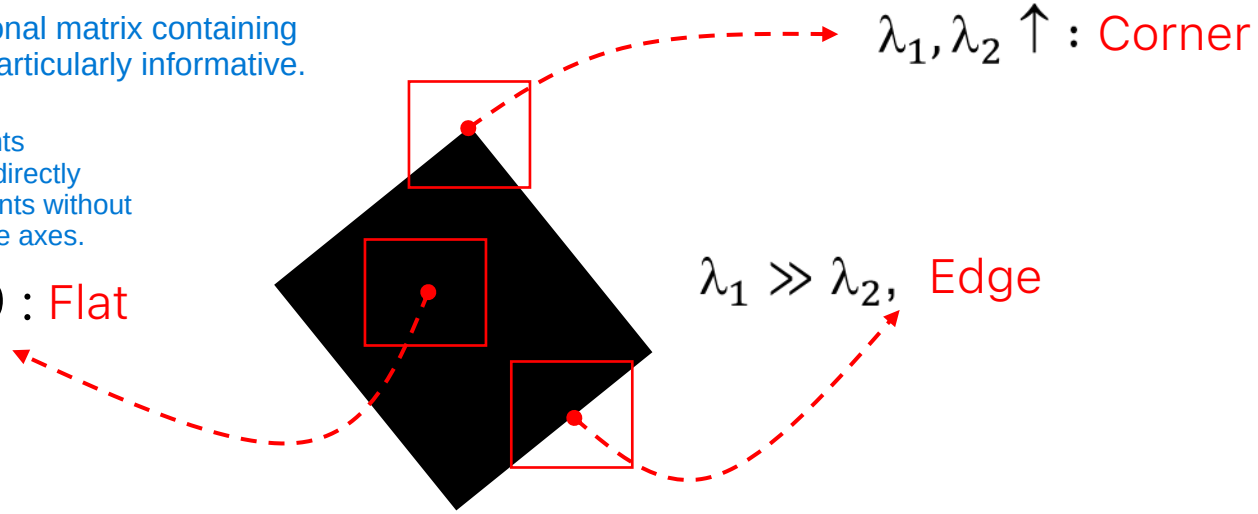
↓

Since the axes are now aligned with the principal components (directions of greatest and least variance), the eigenvalues directly represent the scaling factors along these principal components without any influence from the original (potentially misaligned) image axes.

↓

This simplifies the interpretation of the eigenvalues:

- The highest eigenvalue (lambda1) directly reflects the magnitude of intensity change along the direction of greatest variance (x' axis).
- The lowest eigenvalue (lambda2) directly reflects the magnitude of intensity change along the direction of least variance (y' axis).



Harris Corner Detector

the product is going to be smaller than the trace if the eigenvalues are very small

- You need to compute eigenvalues at each pixel => costly
- Compute a more efficient "cornerness" function:

tr, trace = sum of the diag. values

Captures the overall intensity variation within the window, regardless of the direction.

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

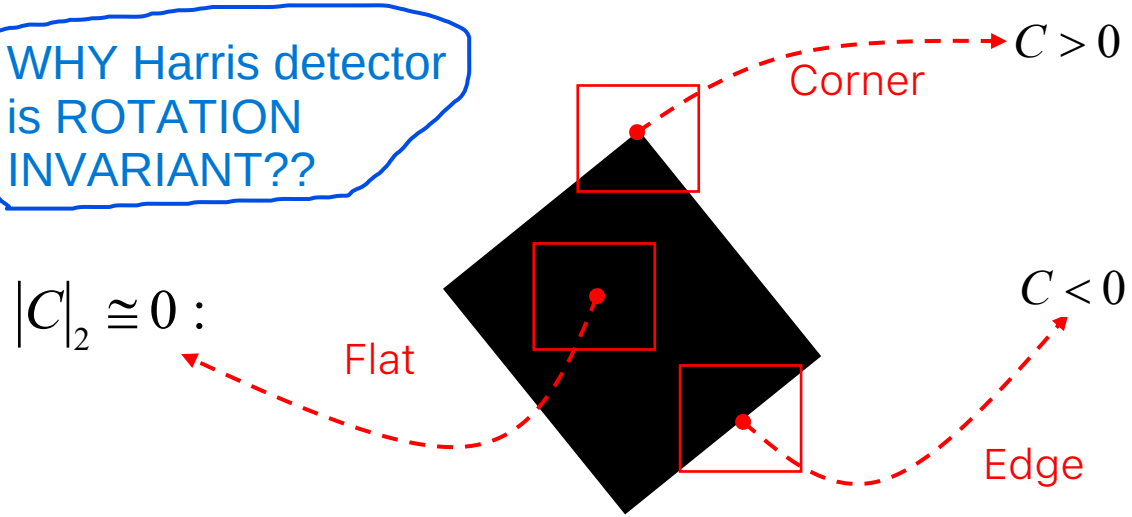
Captures the relationship between the intensity variations in horizontal and vertical directions.

k is an hyperparameter

- Analysis of the above function would show that:

Even when an image is rotated, the relationship between the intensity values of neighboring pixels within the window doesn't change significantly. This is because the local variations are happening over a small, specific region.

WHY Harris detector is ROTATION INVARIANT??



The trace and determinant capture properties of the intensity variations within the window, and these properties are not significantly affected by image rotations.

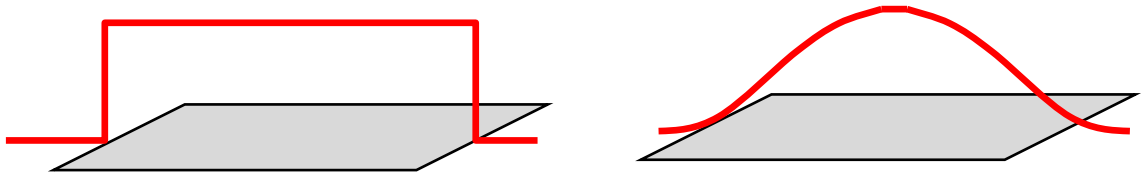
Rotations primarily affect the relative positioning of edges (horizontal vs. vertical), but the presence of significant changes in both directions (characteristic of corners) remains evident in the determinant's value, making it less sensitive to rotations.

Harris Corner Detector

because computing eigenvalues and the matrix is heavy, harris approximates with a cornerness function

- The Harris corner detection algorithm can thus be summarized as follows:
  1. Compute C at each pixel C is the cornerness
  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

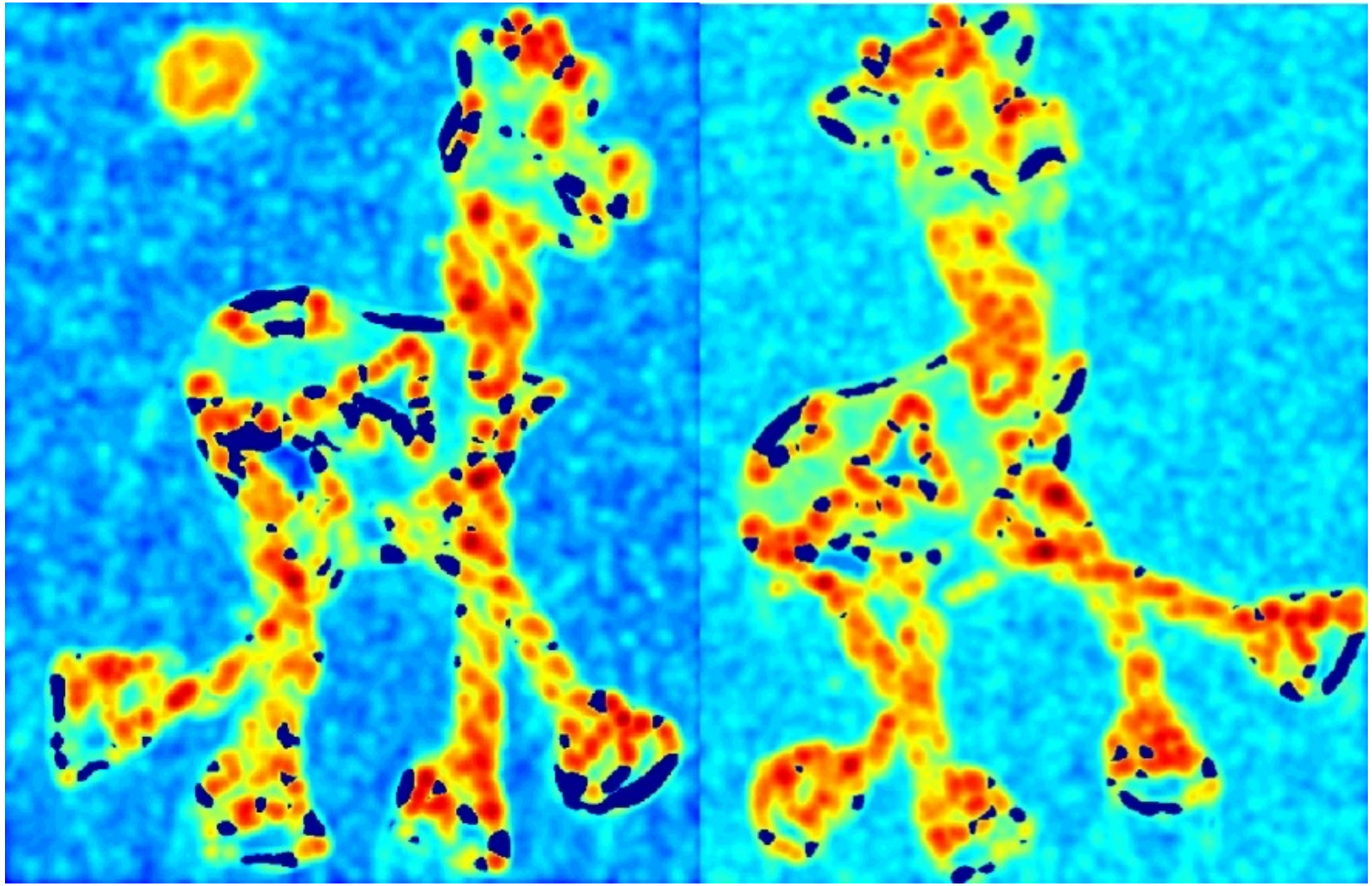
Gaussian

Harris Corner Detector

rotated and less bright

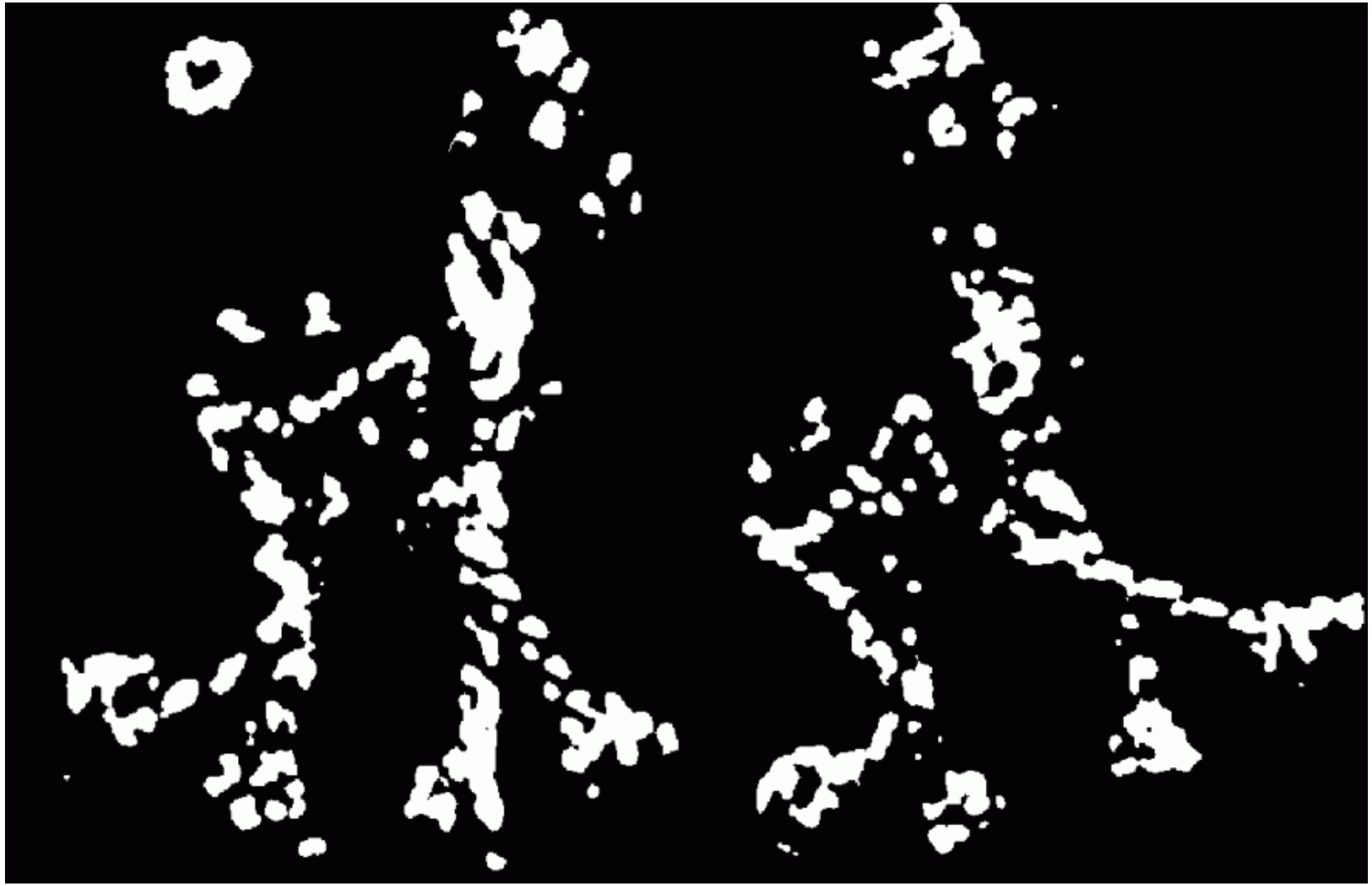
Harris Corner Detector

orange: more cornerness value



Harris Corner Detector

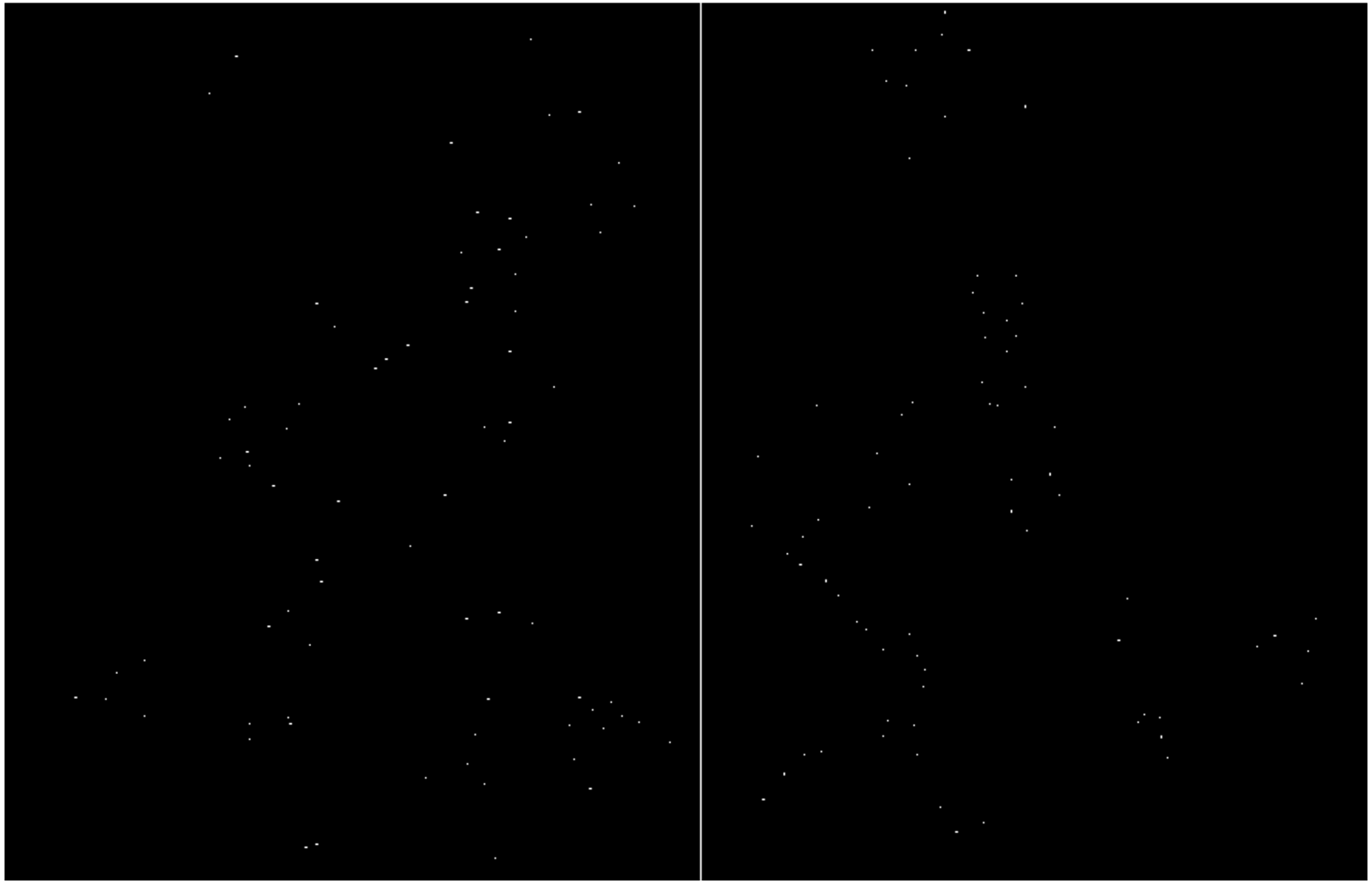
thresholding: we only keep pixels with a cornerness value above a certain threshold



Harris Corner Detector

it eliminates redundant or noisy responses, filtering out only local maxima

after NMS



Harris Corner Detector

the red ones are salient points

Next lectures…

- 13/04 Laboratory on Scale Invariant Features
- 17/04 Student survey + Last lecture for the first module (Prof. Lisanti)
- 20/04 First Lecture for the second part of the course (Prof. Salti)

we want the perfect detector, invariant to all transformations

Invariance Properties

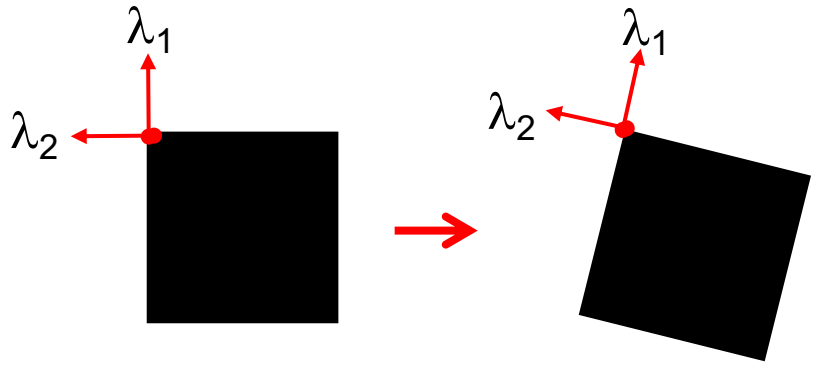
affine intensity change: $I'_{(x,y)} = \text{alpha} * I_{(x,y)} + \text{bias}$

this is the change in intensity between two images

Harris is rotation invariant, as we have already seen

but not to intensity change, only partially

- 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



In image processing, an affine transformation is a geometric transformation that preserves straight lines and parallelism but can modify lengths, angles, and areas; and this transformation is applied to the intensity value of an image. Common examples include scaling, rotation, shearing, and a combination of these.

- No invariance to an affine intendity change
  - Yes, for additive bias (I′ = I + b) due to the use of derivatives

    An additive bias primarily affects the absolute intensity values of pixels but tends to preserve the relative intensity differences between neighboring pixels within a small local region (like the window used in the Harris detector).

$$I' = I + b$$

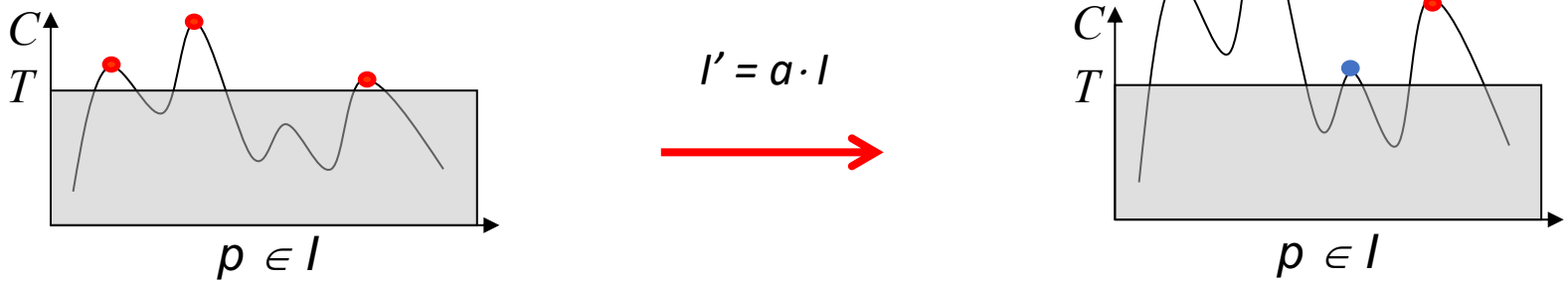
just consider the additive component of affine intendity change

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

  - No, to multiplication by a gain factor (I′ = a · I) => derivatives get multiplied by the same factor

    Scaling intensity values by a constant factor (alpha) affects both the absolute intensity values and the relative differences between neighboring pixels, alterating the values of the derivatives used in M, potentially changing the cornerness function

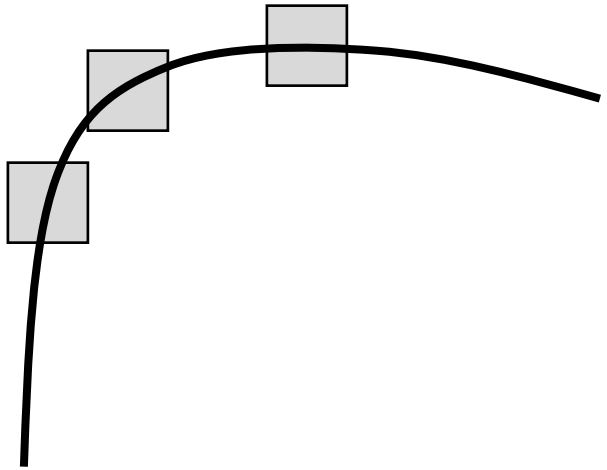
just considering the mulipl. factor alpha



Invariance Properties

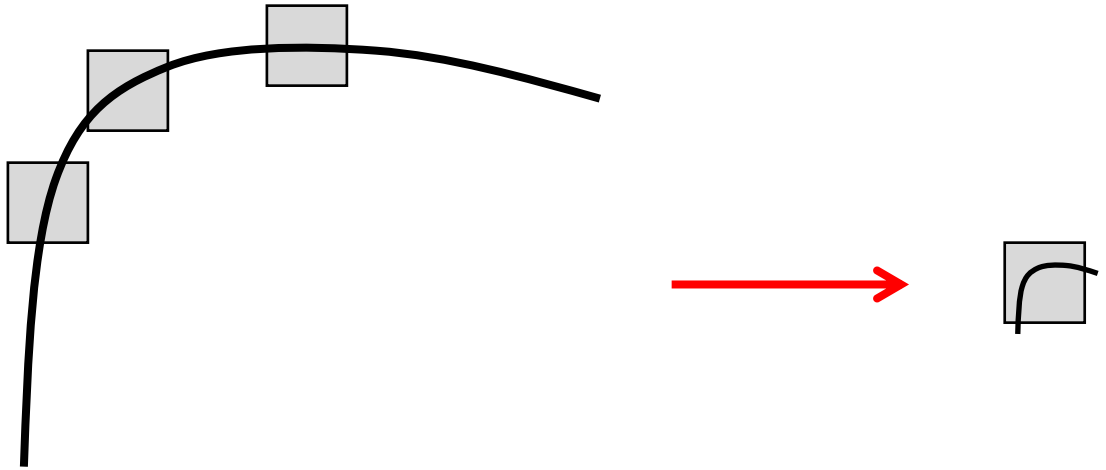
what about scaling?

- 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 problem

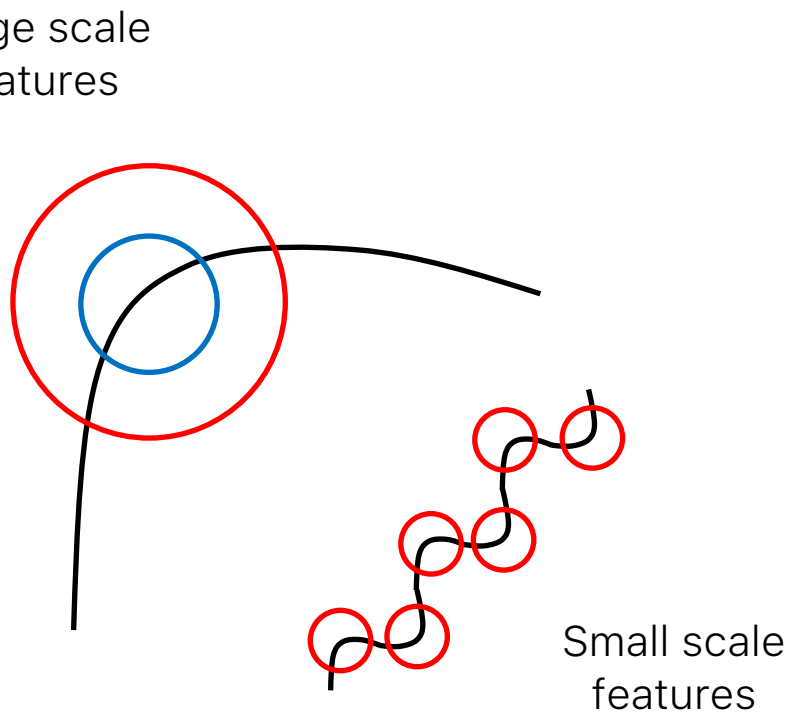
- The use of a fixed detection window size makes it impossible to repeatably detect homologous features when they appear at different scales in images

- Harris is not scale invariant

we need a multi-scale detector

Invariance Properties

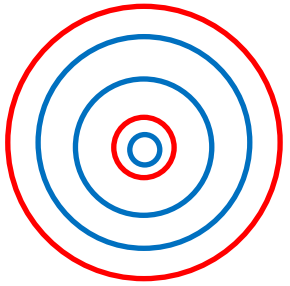
- An image contains features at different scales, i.e. points that stand-out as interesting as long as a proper neighbourhood size is chosen to evaluate the chosen interestingness criterion



- Detecting all features requires to analyze the image across the range of scales “deemed as relevant”

"considered"

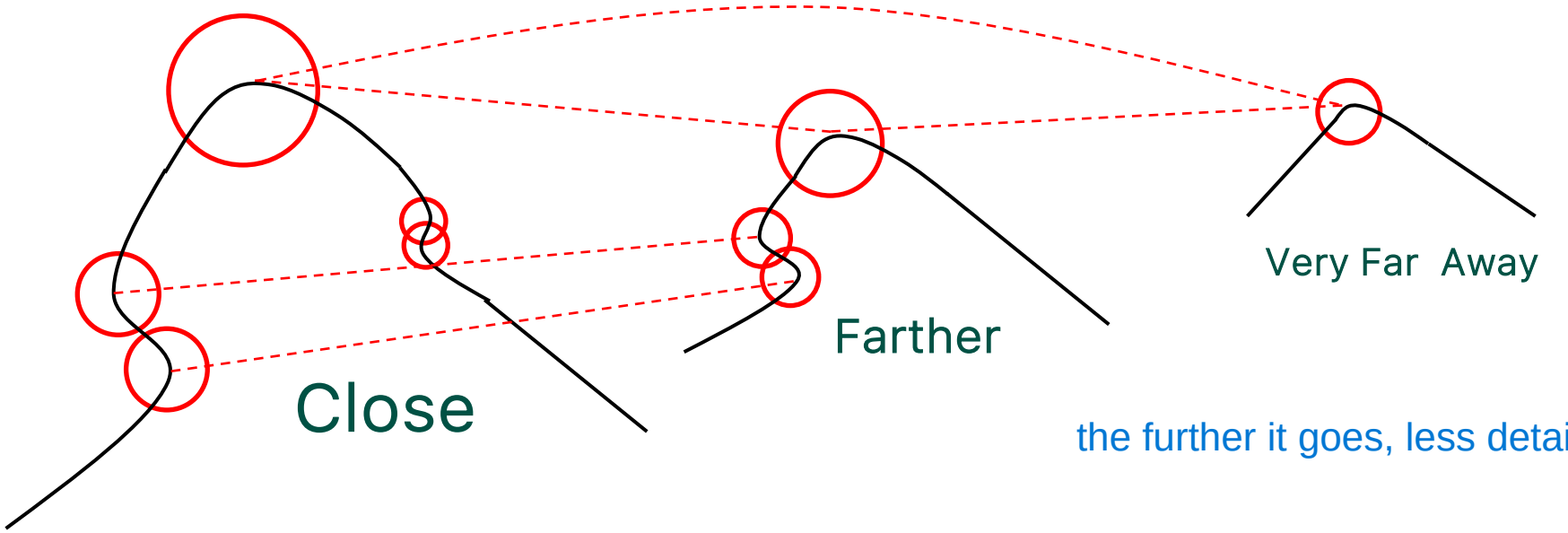
- The more features we gather the higher the chance to match across pictures



Scale Invariance

in order to match two descriptions, they have to contain the same level of details -> how this issue can be solved?
- FILTERING, namely smoothing details

- 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)



the further it goes, less details it shows!!

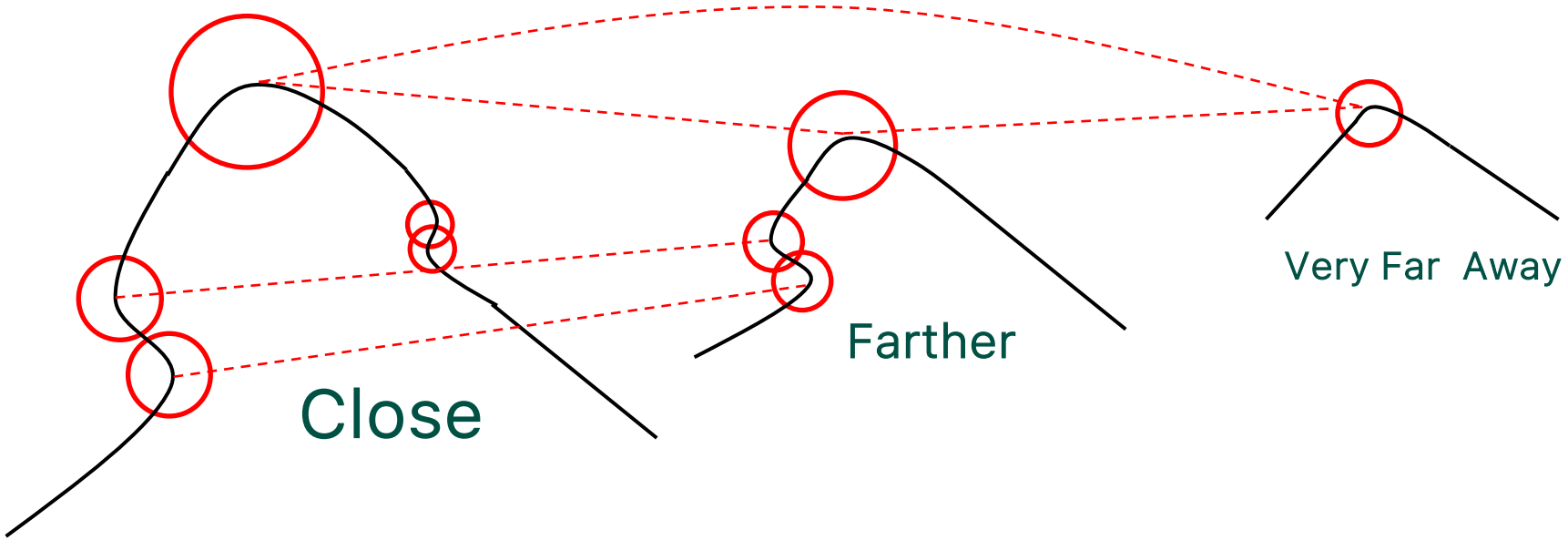
- Features do exist within a certain range of scales
  - Finding similar features despite the different scales at which they may appear in different images
  - The same feature would simply be detected at different steps within a multi-scale image analysis process

Scale Invariance

the bigger the window, the richer the level of detail

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

we do that to maintain the same level of detail in order to match descriptors of features at different scales



- 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

I keep the window fixed, but I smooth and scale the image instead

- 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

we do that (smoothing) to maintain the same level of detail in order to match descriptors of features at different scales

- 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*

this is the classical convolution between the Gaussian kernel and the image, but in this case we specify SIGMA, the level of smoothing

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

scale-space

- 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* — the most informative scale at which a particular feature should be represented
  - 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?

*a gaussian derivative is just a derivative of the gaussian smoothed image

- 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* (peaks or valleys)

  compute gaussian derivatives with Laplacian operator

  - As we filter more (higher sigma) derivatives tends to become weaker
    - to compensate Lindberg proposes to multiply/normalize derivatives by sigma (scale-normalized)

      scale-normalized derivatives are likely normalized in some way (e.g., divided by the scale parameter of the Gaussian filter). This normalization helps achieve some level of scale invariance, making the derivatives less sensitive to the absolute scale of the image.
    - We have stacked images => we look for extrema in a 3D space (pixel positions as well as scale)

extrema often indicate significant changes in intensity or image structure across different scales. By focusing on extrema, the research aims to identify locations in the image and scale-space that exhibit strong responses, suggesting the presence of informative features.

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

It is used to find regions where the intensity changes rapidly, indicating edges or blobs.

Scale-Normalized LOG

The LoG filter directly calculates the second derivative of a Gaussian-blurred image, highlighting regions with significant intensity changes across a specific scale.

this saliency function recognizes BLOBS, another relevant feature (in addition to edges and corners)

they are found by the application of the Laplacian operator

The Laplacian, denoted by nabla^2, is a mathematical operator that calculates the second derivative of an image's intensity function L(x, y) with respect to both spatial coordinates (x and y).

it essentially measures how quickly the intensity of an image changes at a particular location.

- Scale-normalized Laplacian of Gaussian (LOG)

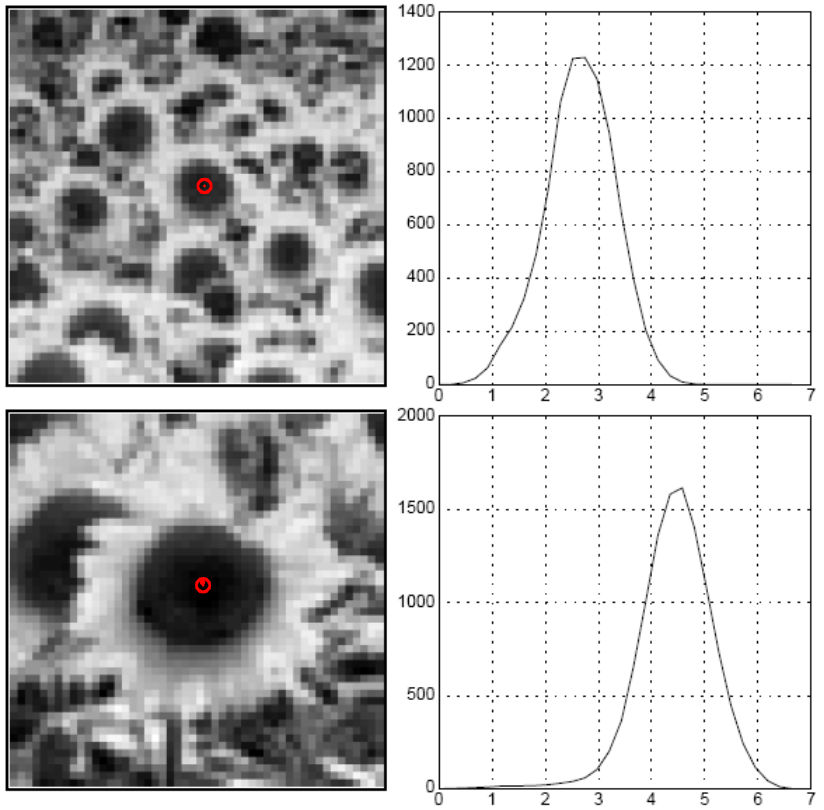
$$F(x, y, \sigma) = \sigma^2 \nabla^2 L(x, y, \sigma) = \sigma^2 (\nabla^2 G(x, y, \sigma) * I(x, y))$$

3D space

normalization

laplacian op.

- $\rightarrow (\bar{x}, \bar{y})$ $F(\bar{x}, \bar{y}, \sigma)^2$



Response of the scale normalized Laplacian (across scale)

we choose this scale because it produces the highest scale-norm. LoG for that featuture

regions with uniform intensity and different in intensity compared to their surroundings

- 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 centers of these blobs (marked as red points) will correspond to the locations where the scale-normalized LoG has extremal values.

- The ratio between the two characteristic scales is roughly the same as the ratio between the sizes (diameters) of the two blobs

The characteristic scale of a blob is the scale sigma at which the scale-normalized LoG reaches its extremum for that blob. This scale is related to the size (diameter) of the blob

- 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

To find features in the image, we need to look for extrema (maxima or minima) in the scale-normalized LoG across all dimensions

blobs detector

Multi-Scale Feature Detection

with blobs, we no longer look at local, but at extrema

for blob-like structure in an image, the intensity typically increases towards the center and then decreases outwards.

When the Laplacian is applied to such a region, the second derivative of the intensity function will be positive. This is because the intensity is initially increasing (positive first derivative) and then starts decreasing (negative second derivative).

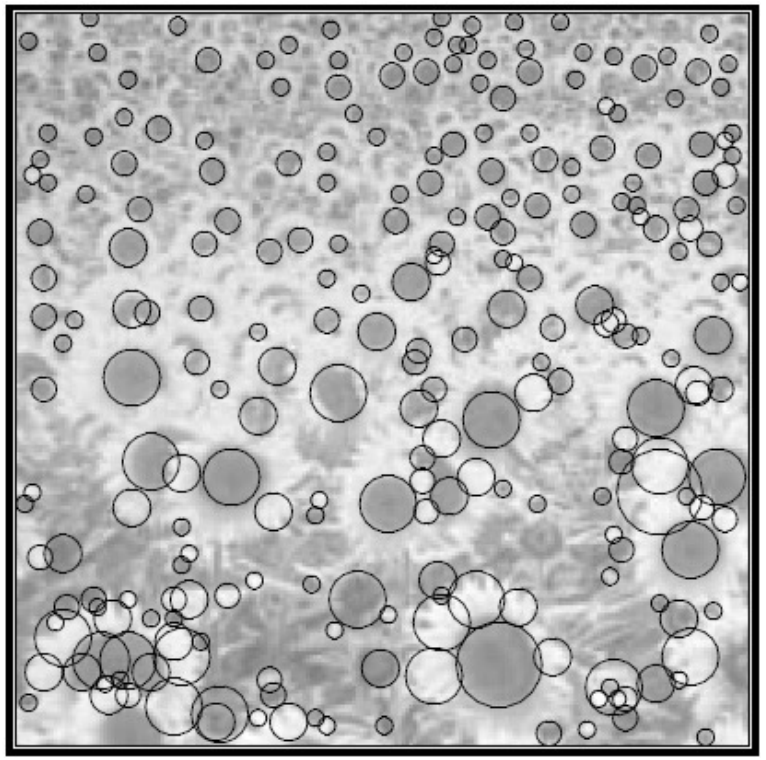
A positive Laplacian value, therefore, indicates a region where the intensity is locally "peaking," which is a characteristic of a blob.

- 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

e.g.

1 - At the center of the blob, the intensity will be highest. Moving outwards, the intensity will decrease. Mathematically, the first derivative of the intensity function at the center will be close to zero (since the intensity isn't changing much).
2 - As we move outwards, the intensity starts decreasing, resulting in a negative first derivative.
3 - The Laplacian, which is the second derivative, captures this change in the rate of change. It will be positive at the center of the blob (since the first derivative goes from positive to negative) and negative outside the blob (where the intensity continues to decrease).

using LoG we can also disambiguate between dark and light blobs



regions of different scales

another saliency function

Difference of Gaussian (DoG)

less computational effort wrt Lindberg's scale normalized LoG

The DoG operation involves subtracting two blurred versions of the image obtained using Gaussian filters with different sigma values. In essence, we are taking the difference between images at two specific scales.

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

Gaussian scale-space

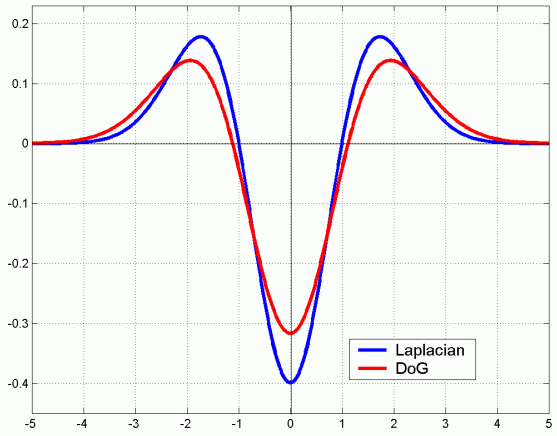
$$DoG(x,y,\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) * I(x,y) = \underbrace{L(x,y,k\sigma)}_{\text{smoothings}} - L(x,y,\sigma)$$

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

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

Scaled version of LoG

- 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)

*D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints", *IJCV*, 2004

Difference of Gaussian (DoG)

The DoG operation emphasizes regions in the resulting image where there are significant intensity changes across those scales. Edges and corners, which often contain high-frequency information, become more prominent in the DoG image compared to the original or blurred versions.

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

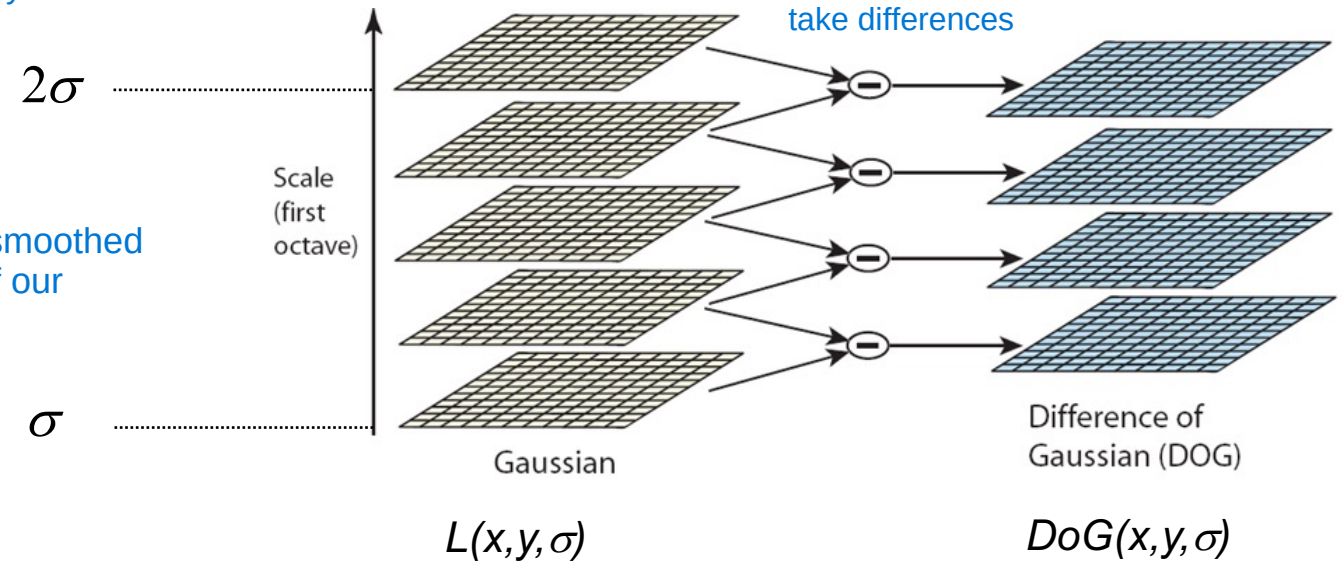
we divide the scale space into octaves

an octave is an interval when smoothing is increased by a factor of 2

our scale space at different levels of smoothing

DoG pyramid

take differences



different smoothed version of our image

how we choose the right difference ?

$L(x,y,\sigma)$

$DoG(x,y,\sigma)$

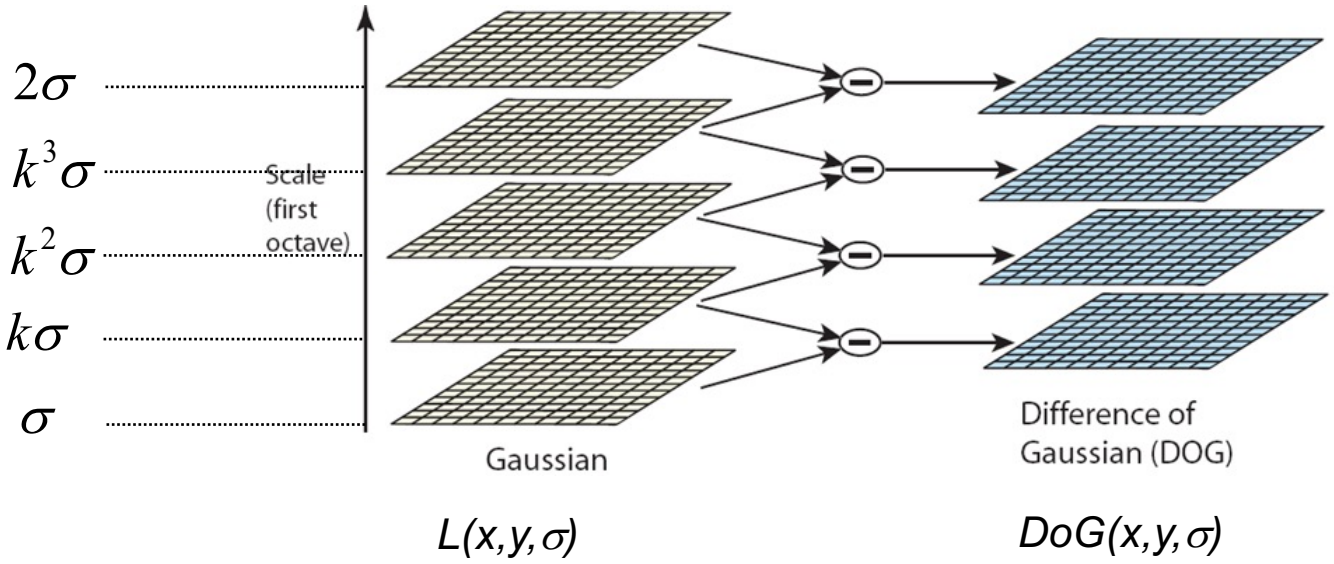
Difference of Gaussian (DoG)

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

Analyzing a single DoG image only provides information about intensity changes at that specific scale.
Since our primary goal is to identify features (like edges, corners, or blobs) that exist consistently across different scales in the image, we need to consider different layers in addition to the one under interest (the above and below ones)

We can identify features that exhibit significant intensity changes not just in one DoG layer but CONSISTENTLY across MULTIPLE SCALES. This ensures the detected features are more likely to be real and not just noise or artifacts of a specific scale.

Non-maximum suppression, which compares the DoG response of a pixel with its neighbors across scales, helps eliminate these "single-scale" extrema and identifies those that PERSIST ACROSS MULTIPLE SCALES, leading to more robust feature detection.



Features like edges and corners can appear at different sizes in an image. By looking at extrema across scales, we can detect them regardless of their absolute size in the image, leading to some level of scale invariance in feature detection.

Difference of Gaussian (DoG)

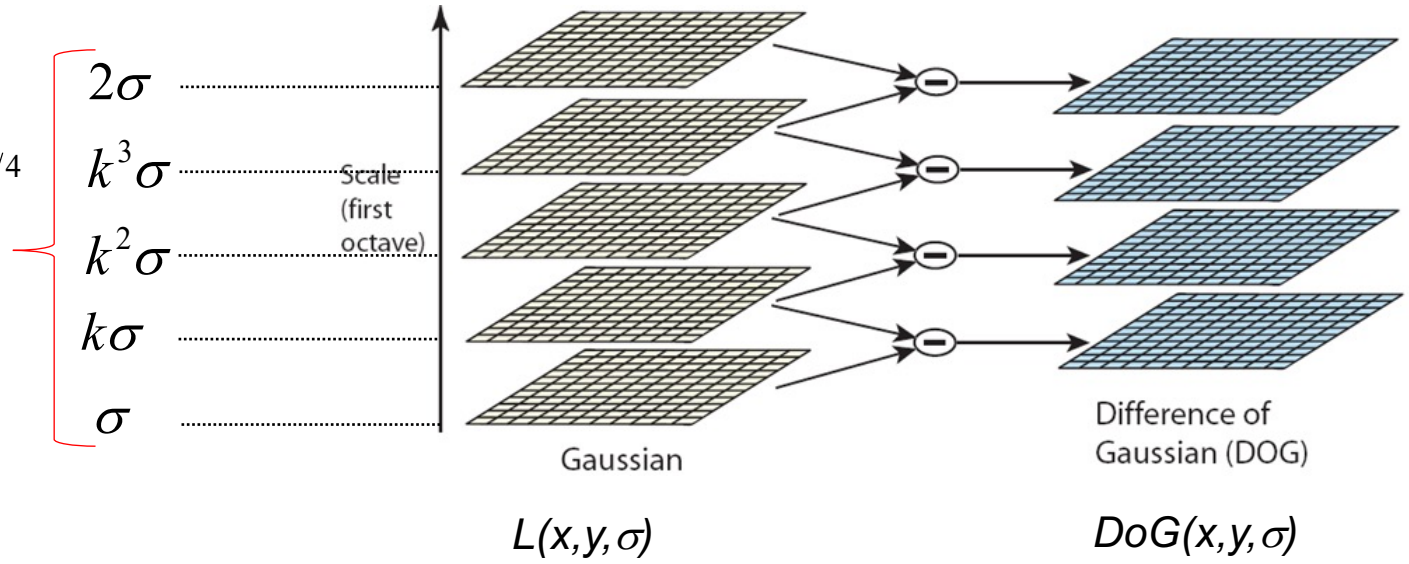
- We compute several gaussian smoothed function within an octave
  - For each pair we take the differences => than we seek for extrema in the DoG

We want to find extrema that are significant across multiple scales within the DoG pyramid.

In an octave we sample s scales:

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

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



$L(x,y,\sigma)$

$DoG(x,y,\sigma)$

Difference of Gaussian (DoG)

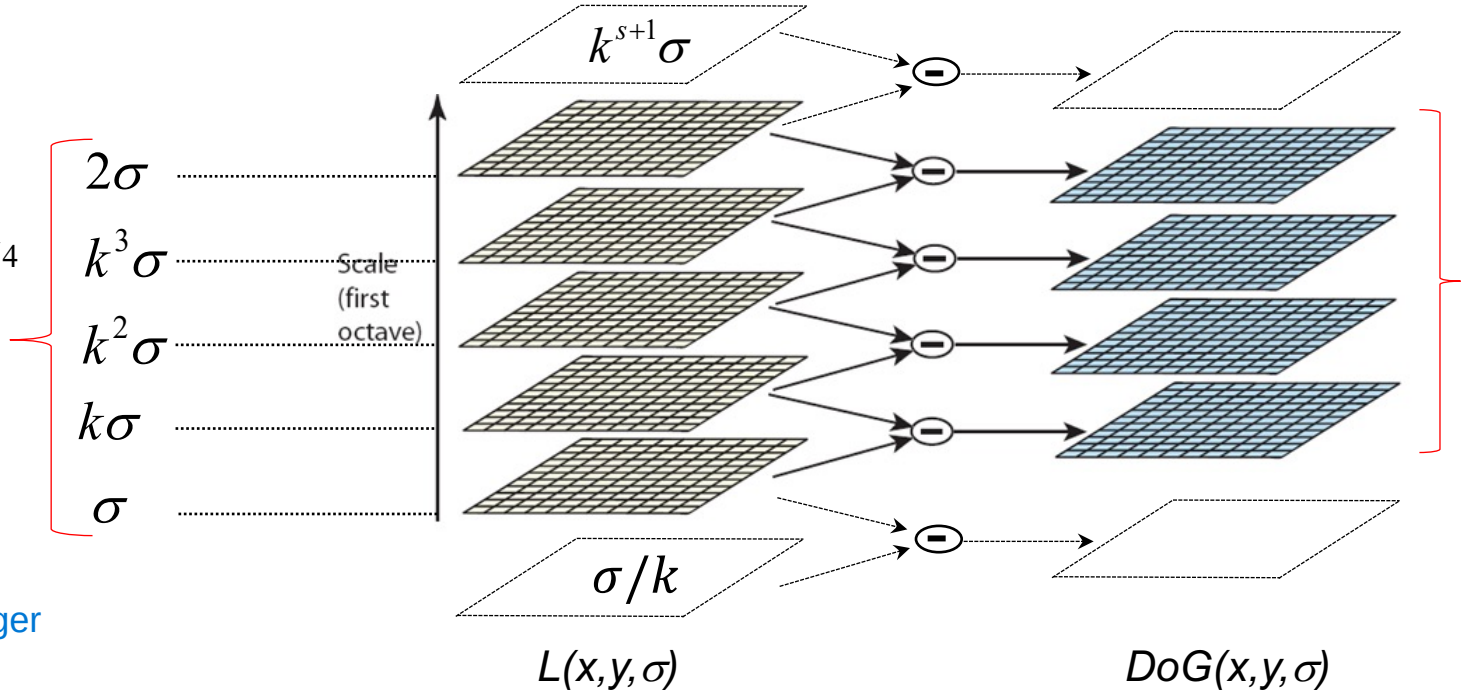
- To compute the extrema on the blue one we need two additional Gaussian filtered layer

In an octave we sample s scales:

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

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

we cannot increase sigma up to 3 sigma e.g. because it means more smoothing and bigger window



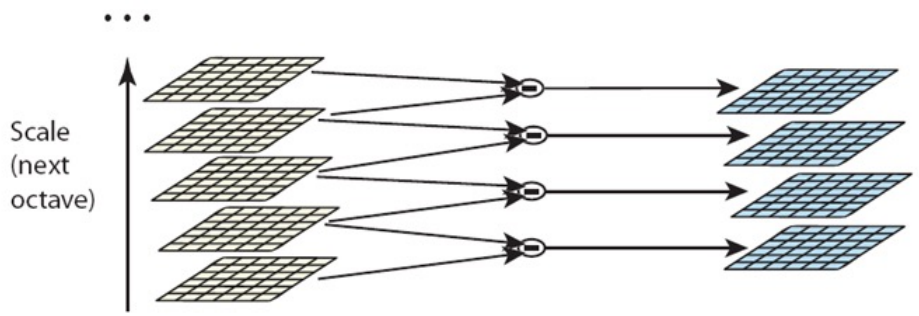
Extrema detection on s DoG images per octave

Difference of Gaussian (DoG)

after an octave is complete,

we re-do the previous process on a smaller version of the image

DOWNSAMPLING



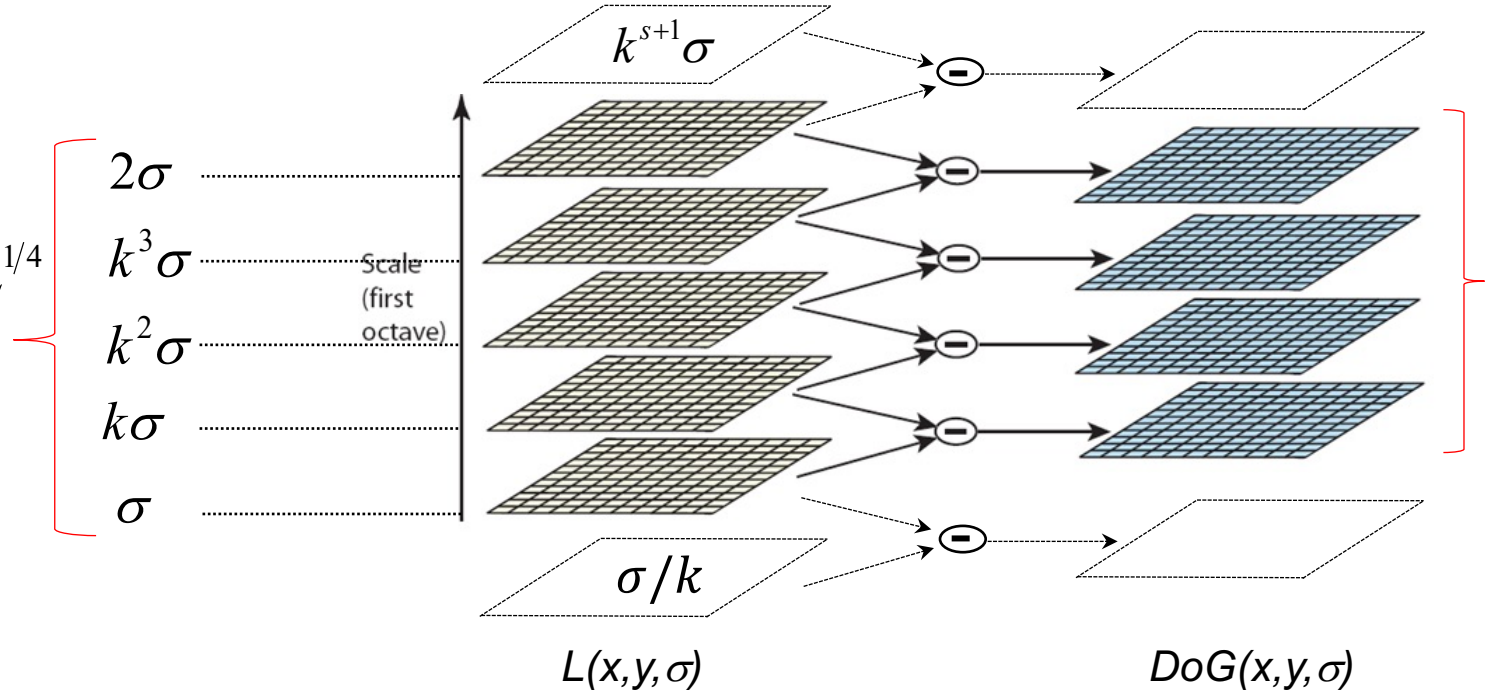
after an octave is complete

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:

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

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

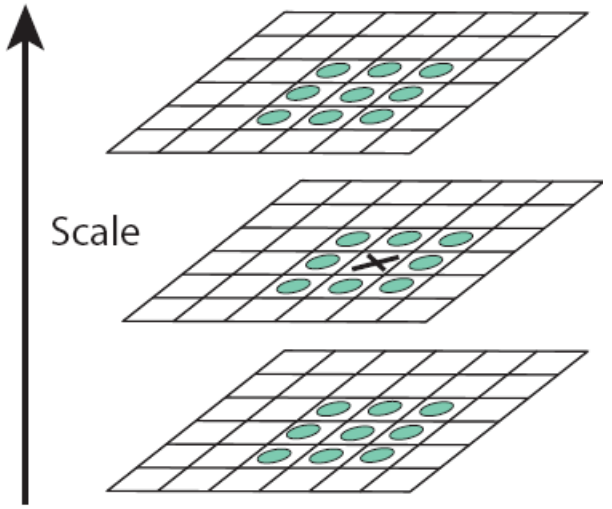


Extrema detection on s DoG images per octave

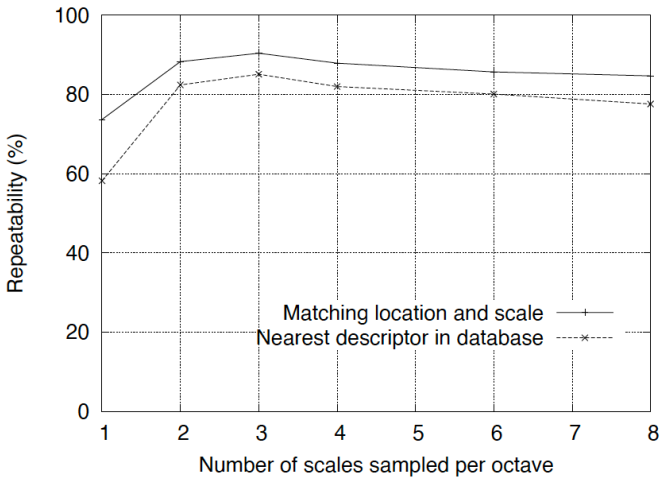
Keypoint Detection and Tuning

- 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

DoG images



- 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



1. For each pixel location in a DoG image within the pyramid, compare its DoG response with its neighboring pixels in both the same layer and adjacent layers (the layer above and below in the pyramid).
2. If the pixel's DoG response is a maximum (higher than all its neighbors) or a minimum (lower than all its neighbors) across these comparisons, it's considered a potential extremum.
3. This approach ensures that the extremum is not just a peak or valley within a single DoG image but persists across scales, indicating a more robust feature.

Exemplar DoG keypoints

tresholding on the magnitude

- 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

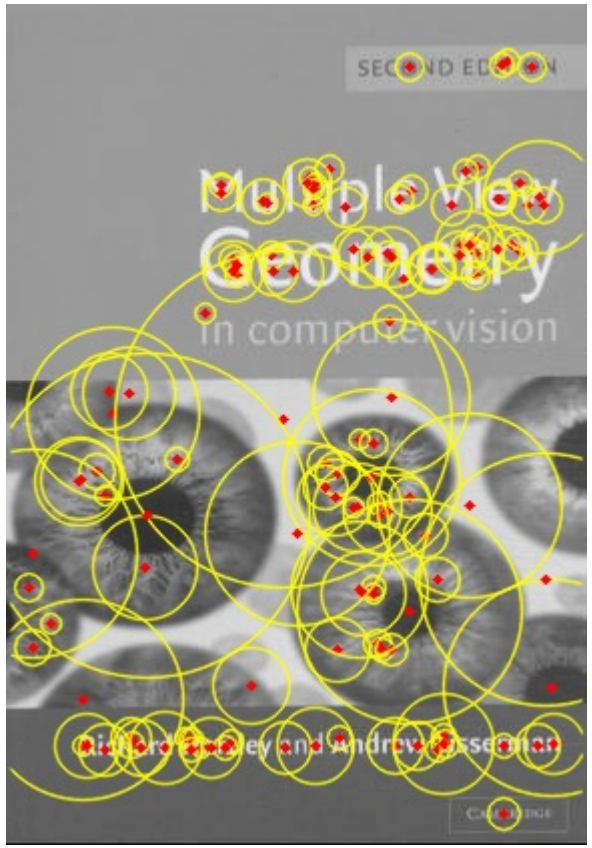
Keypoints after pruning weak responses

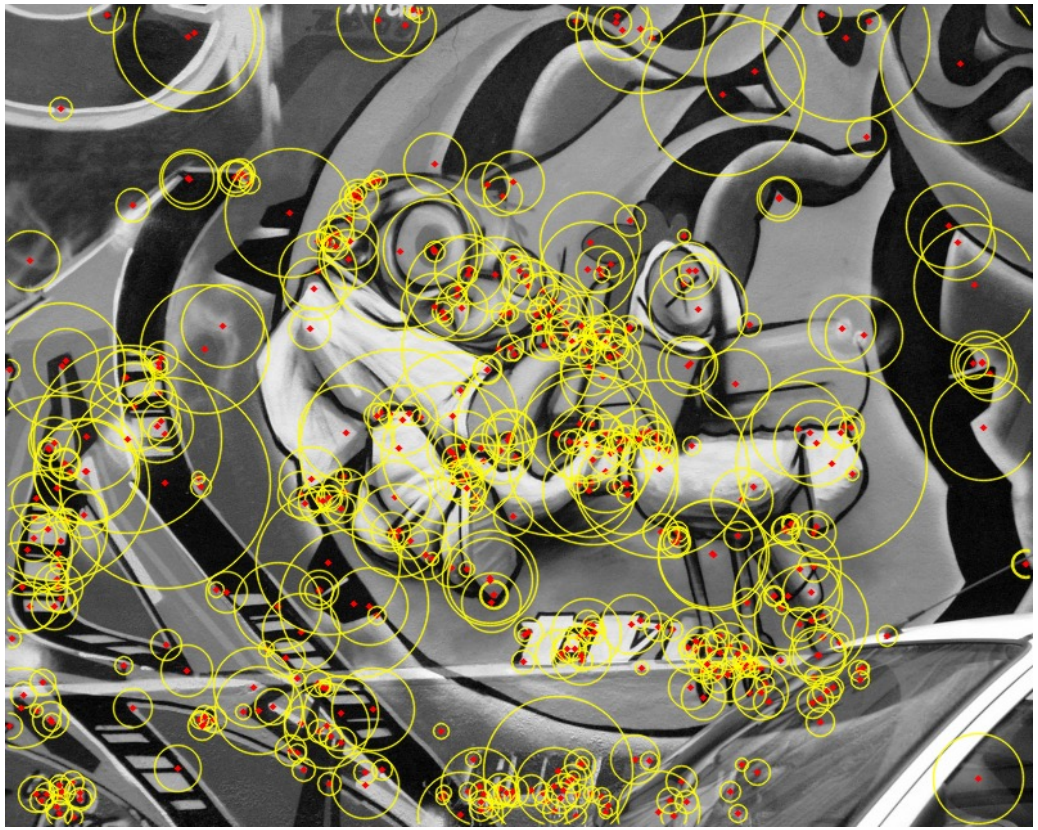
Final keypoints after pruning those located along edges

Exemplar DoG keypoints

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

because the larger is sigma, the more the detector is intended to detect large features (as blobs/circles)





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
    typical approach

- 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

The DoG response for a feature (like an edge or corner) typically reaches a peak (maximum) at a specific scale within the DoG pyramid. This peak signifies the scale where the intensity change associated with the feature is most pronounced.

By analyzing the DoG response across different scales, we can identify the layer where the peak response occurs. This layer corresponds to the scale at which the feature is most distinct and contributes most strongly to the image information.

In computer vision, feature descriptors are used to capture key properties of an image region (patch) around a detected feature (like an edge or corner). These descriptors should be informative and robust to variations like image noise or small rotations.

Exemplar DoG keypoints

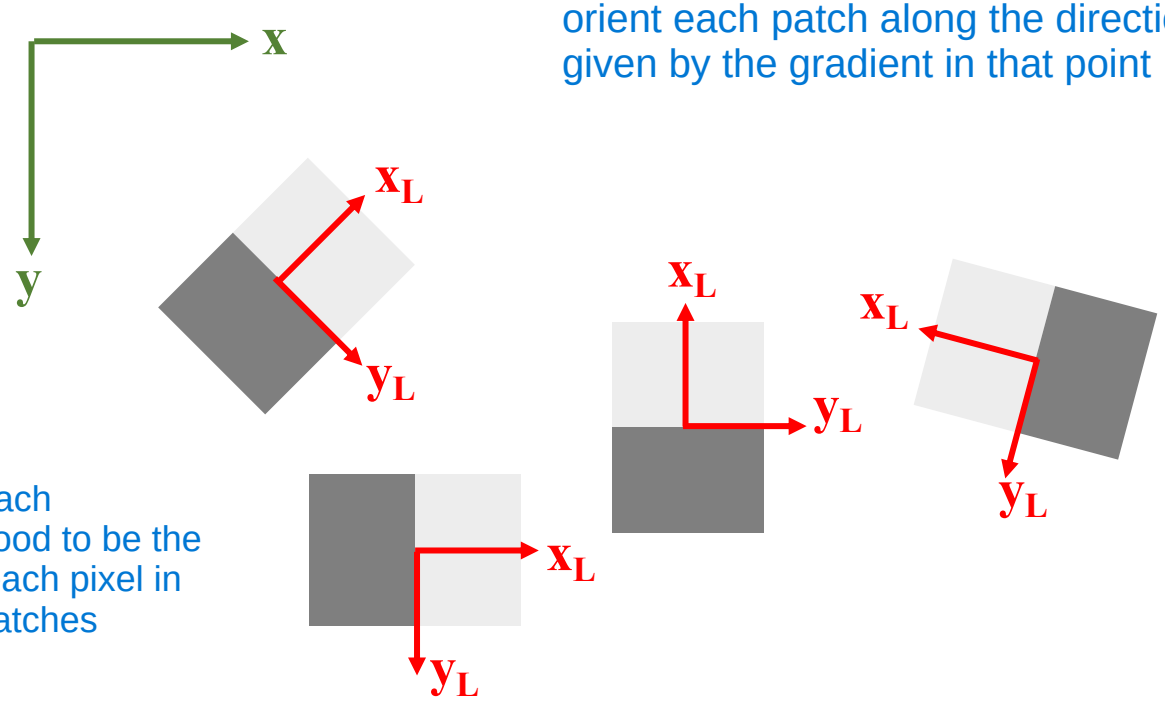
An ideal feature descriptor should be insensitive to the overall rotation of the object in the image. This is crucial for matching features across images where the object might appear at different orientations.

This direction essentially represents the feature's "natural" orientation.

- 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.

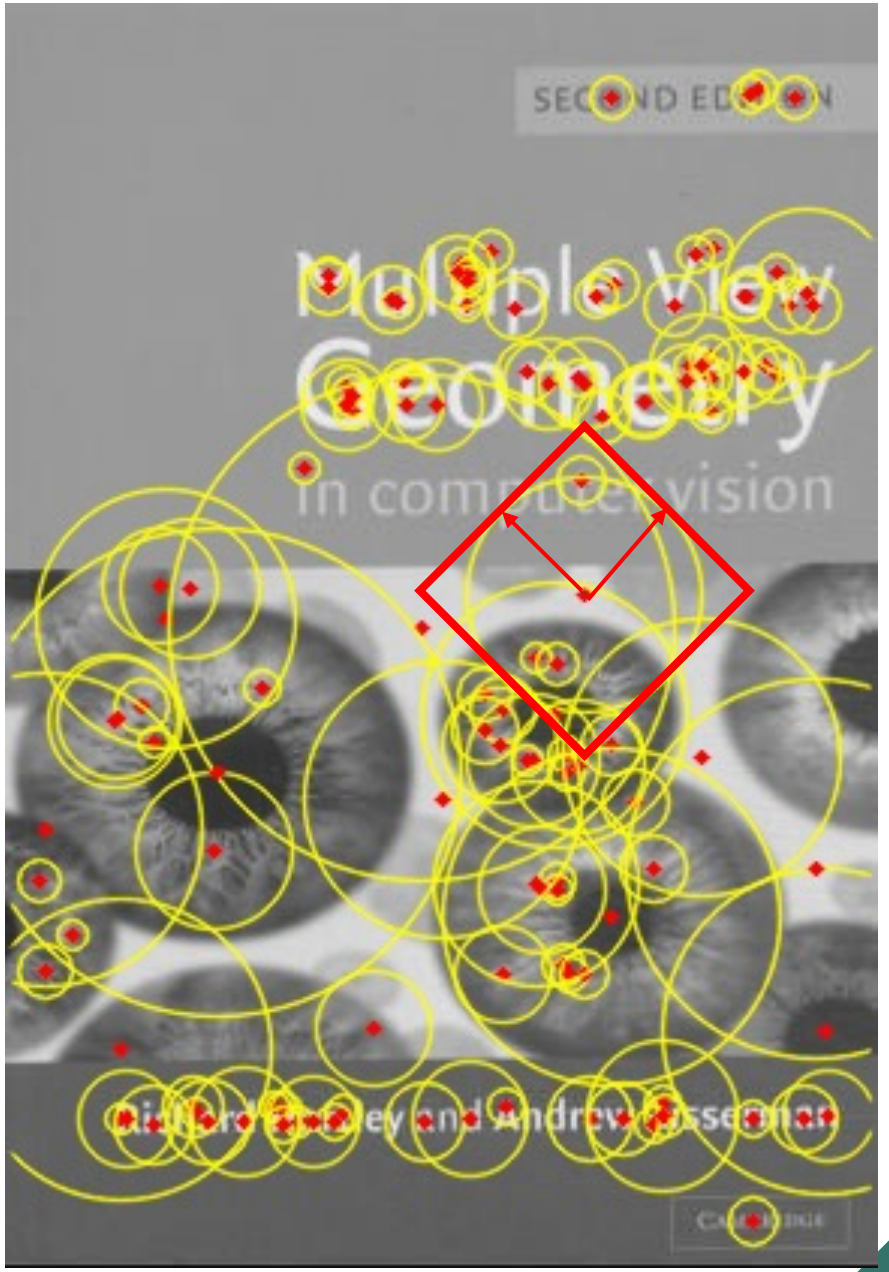
This frame essentially establishes a coordinate system specific to the feature, with axes aligned according to the canonical orientation.

- When computing the descriptor, pixel coordinates are taken in the local reference frame.



orient each patch along the direction given by the gradient in that point

we want each neighborhood to be the same for each pixel in different patches



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

the canonical orientation it's usually calculated within the local neighborhood surrounding the detected keypoint representing the feature itself.

- 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 this for every keypoint

An orientation histogram is created to capture the distribution of gradient orientations within the local neighborhood around the keypoint. This essentially tells us how much "gradient energy" falls into different orientation bins.

  - 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)

use a Gaussian weighting to reduce the influence of noisy or irrelevant gradients farther away to the keypoint

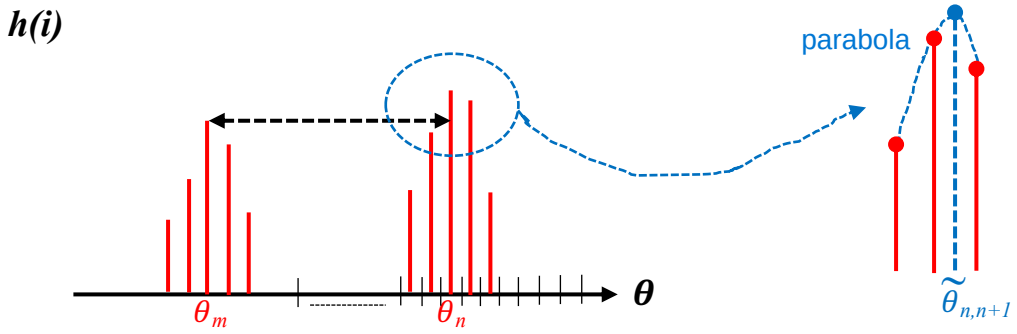
furthermore, by using a scale-dependent sigma, the weighting adapts to the feature's size.

The contribution of a pixel is based on its gradient magnitude. Pixels with stronger intensity changes (higher gradient magnitude) contribute more significantly to the histogram.

- 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



This is an ambiguous keypoint, we have two prominent directions

- 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

due to noise, discretization (limited number of bins), or other factors, the peak might not be perfectly sharp. It could be broader or have a slightly flattened top.

by fitting a parabola to the points around the peak bin in the histogram, SIFT aims to achieve a more precise estimation of the sub-peak location. This sub-peak location is considered a more accurate representation of the true dominant orientation compared to the central value of the peak bin itself.

SIFT RECAP PIPELINE →

SIFT Descriptor

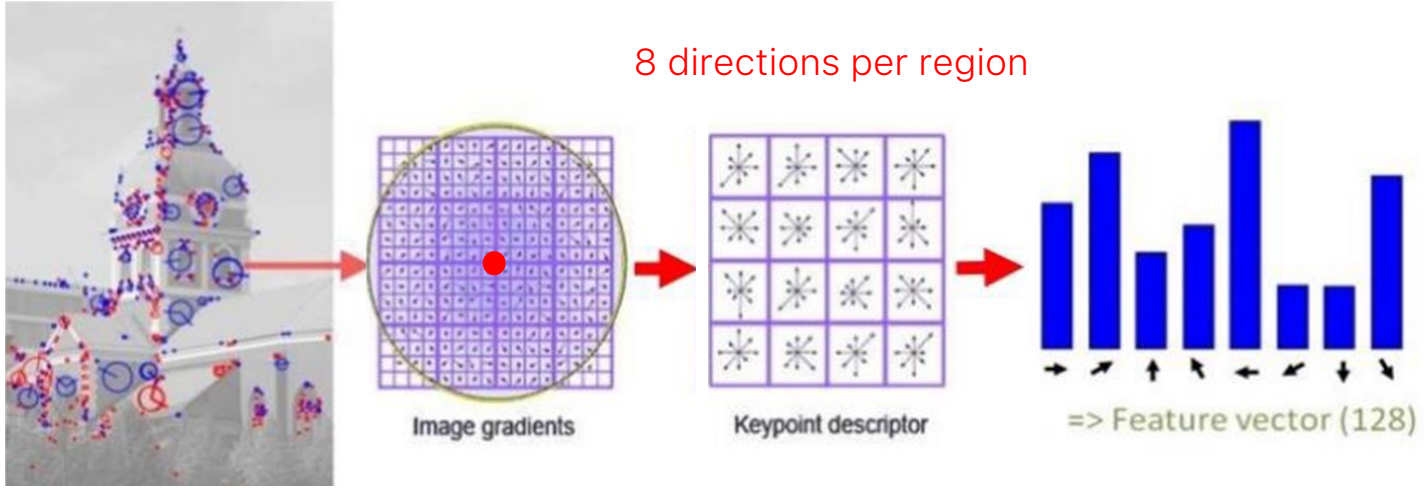
1 Scale-Space Representation: SIFT starts by creating a scale-space representation of the image. This involves applying Gaussian filters at different scales (blur levels) to the image, resulting in a pyramid-like structure.

2 DoG (Difference of Gaussian) Extrema Detection: within the scale-space pyramid, SIFT identifies potential keypoints (features) by computing the Difference of Gaussian (DoG) between adjacent scales. This highlights regions with significant intensity changes across scales.

- 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)

3 Orientation Assignment: for each keypoint, a dominant orientation is assigned. This is achieved by analyzing the gradient directions and magnitudes within a local neighborhood around the keypoint. An orientation histogram is created to capture the distribution of these gradients. A peak in the histogram corresponds to the dominant orientation. SIFT also refines this orientation with sub-pixel accuracy by fitting a parabola around the peak.

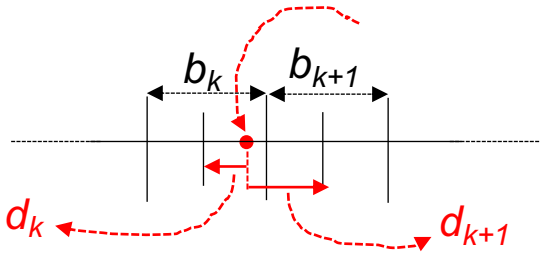
4 Feature Descriptor: finally, a feature descriptor is computed for each keypoint. This descriptor captures the keypoint's local properties in a way that is robust to variations like image noise, illumination changes, and even small rotations. SIFT uses a gradient histogram around the keypoint, weighted by a Gaussian function and oriented according to the assigned orientation.



The descriptor size is given by the number of regions times the number of histogram bins per region, i.e. 4x4x8 = 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:



$$\begin{cases} w_k = 1 - d_k \\ w_{k+1} = 1 - d_{k+1} \end{cases}$$

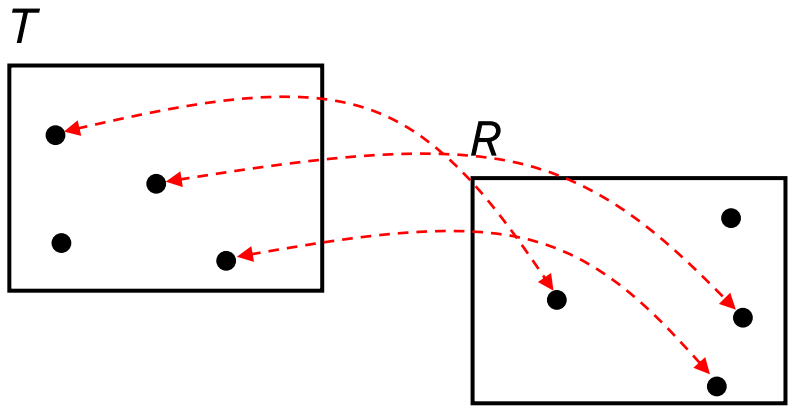
- This is done within an histogram as well as between regions (the contribution is spread bilinearly between 4 adjacent regions). Hence, the overall scheme is referred to as *trilinear interpolation*.
- The descriptor is normalized to unit length to gain invariance wrt affine intensity changes
  - To 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

how to filter out matches? tresholding

- 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)

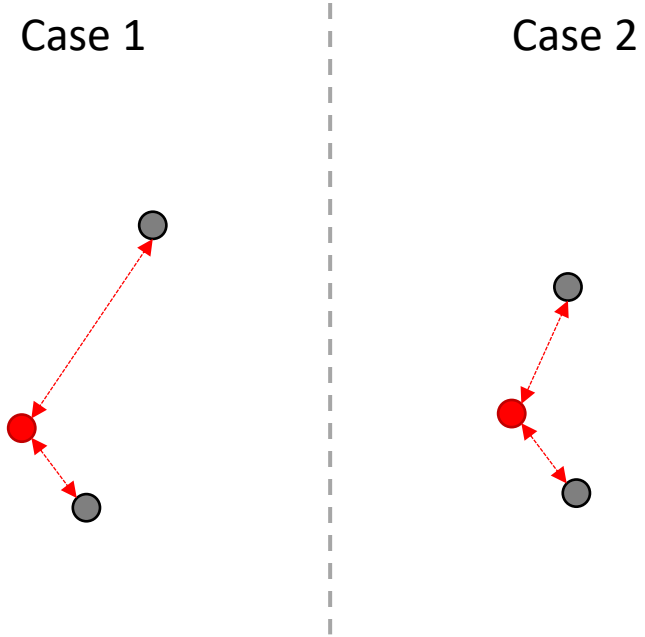
the descriptor is high-dimensional
then it's not feasible

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

look at two neighbors

the closer these distances are,
the higher the ambiguity

  - 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



Validating Matches

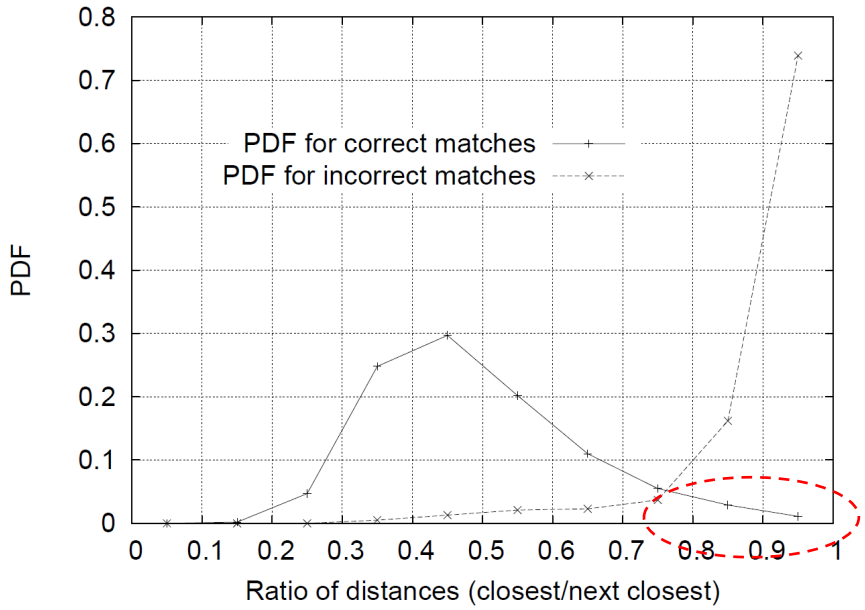
- The found NN does not necessarily provide a valid correspondence as some features in *T* may not have a corresponding feature in *R* (clutter and/or viewpoint changes)

- 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)

The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the second closest



  - 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

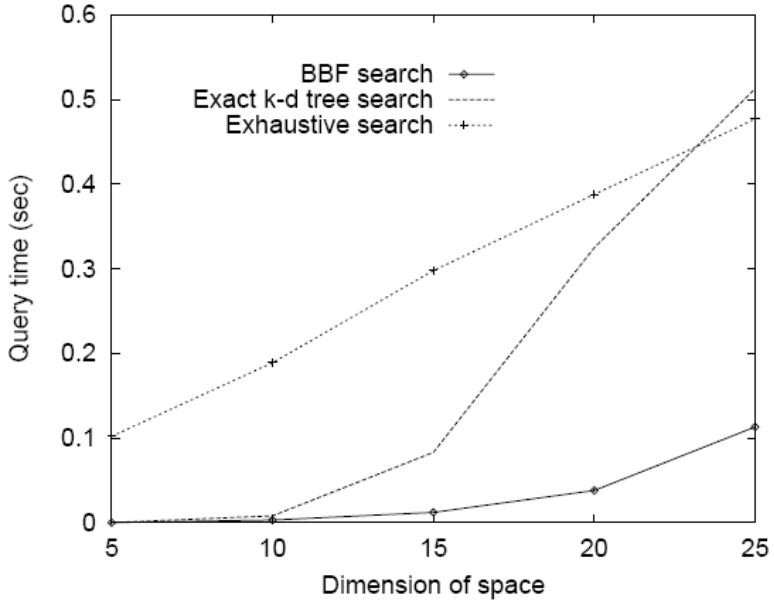
- Lowe shows that T=0.8 may allow for rejecting 90% of wrong matches while missing only 5% of those correct

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