Local Invariant Features

Luigi Di Stefano, Samuele Salti

luigi.distefano@unibo.it, samuele.salti@unibo.it

Computer Vision Laboratory (**CVLab**)
Department of Computer Science and Engineering (**DISI**)
Alma Mater Studiorum - Università degli Studi di Bologna

Correspondences are Key

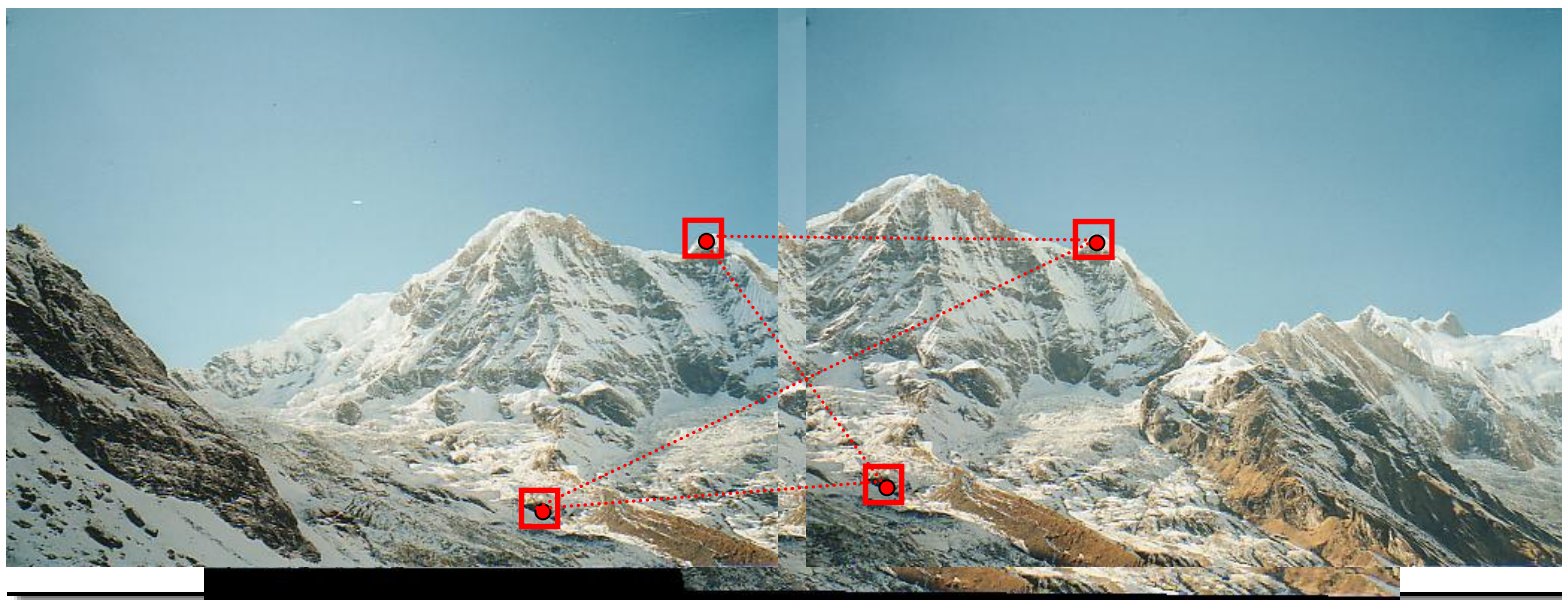- **A great variety of Computer Vision problems can be dealt with by finding**

<p style="text-align:center">**corresponding points**</p>

  between two (or more) images of a scene.

- **Corresponding (aka homologous) points: image points which are the projection of the same 3D point in different views of the scene.**

- **Establishing correspondences may be difficult as homologous points may look different in the different views, e.g. due to viewpoint variations and/or lighting changes.**

Samuele Salti, Luigi Di Stefano

Examplar Task: Mosaicing

- **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" to recognize salient points.**
  - **Compare descriptions to tell apart salient points.**

Samuele Salti, Luigi Di Stefano

Examplar Task: Mosaicing

- **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" to recognize salient points.**
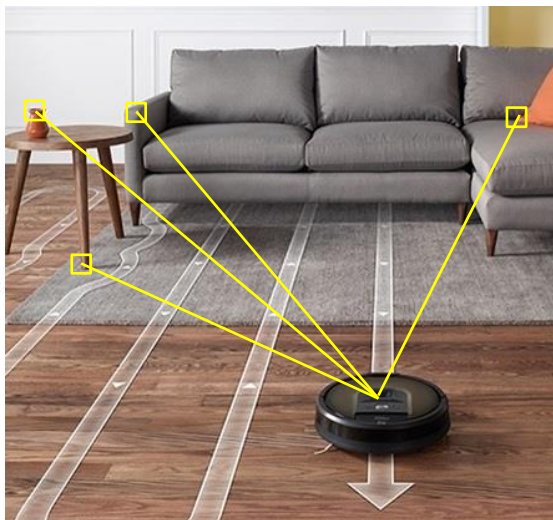   - **Compare descriptions to tell apart salient points.**

Samuele Salti, Luigi Di Stefano

Other Examples…

Object Detection

SLAM (Simultaneous Localizazion and Mapping)

Camera Tracking, e.g. for Markerless Augmented Reality

3D Reconstruction

e.g. for Robot Navigation

Visual Search

*Samuele Salti, Luigi Di Stefano*
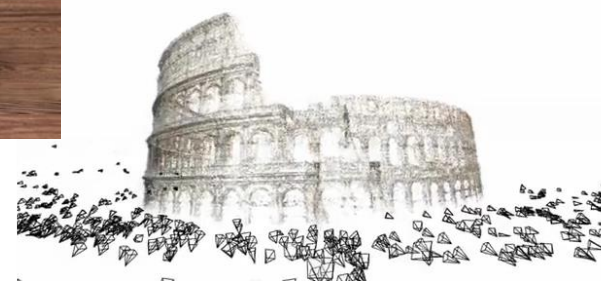
The Local Invariant Features Paradigm

The task of establishing correspondences is split into 3 successive steps:

- **Detection** of salient points (aka <u>keypoints</u>, interest points, features ...).
- **Description** - Computation of a suitable descriptor based on a neighbourhood around a keypoint.
- **Matching** descriptors between images.

and should be **invariant** (robust) to the many tranformations that may relate images.

Samuele Salti, Luigi Di Stefano
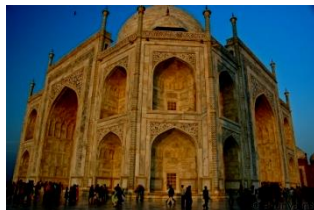
# Properties of good detectors/descriptors

- **Detector**
  - **Repeatability**: it should find the same keypoints in different views of the scene despite the tranformations undergone by the images.
  - **Interestingness/Saliency:** it should find keypoints surrounded by informative patterns of intensities, which would render them amenable to be told apart by the matching process.

- **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 possibile, 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).

*Samuele Salti, Luigi Di Stefano*

Performance of the matching process
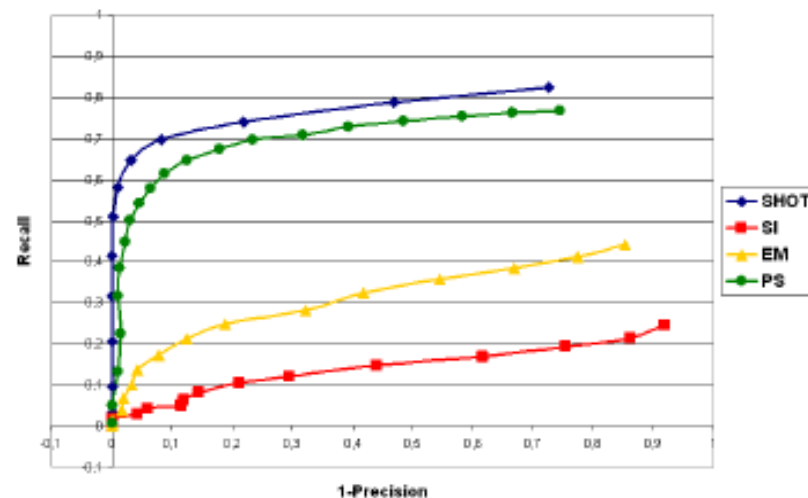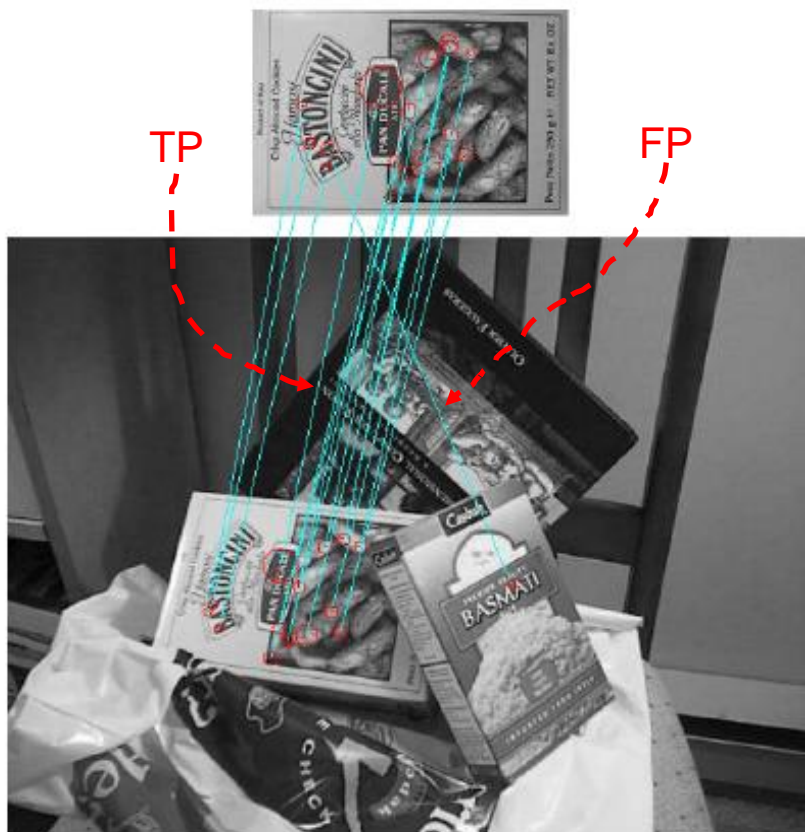
As long as one tries to gather more matches these become less precise. Comparison across the whole range of operating points is achieved by *Precision (1-Precision) – Recall curves*.

$$\text{Recall} = \frac{TP}{P}; \qquad \text{Precision} = \frac{TP}{TP + FP}$$

Samuele Salti, Luigi Di Stefano

Interest Points vs. Edges

- **Edge pixels can be hardly told apart as they look very similar along the direction perpendicular to the gradient.**
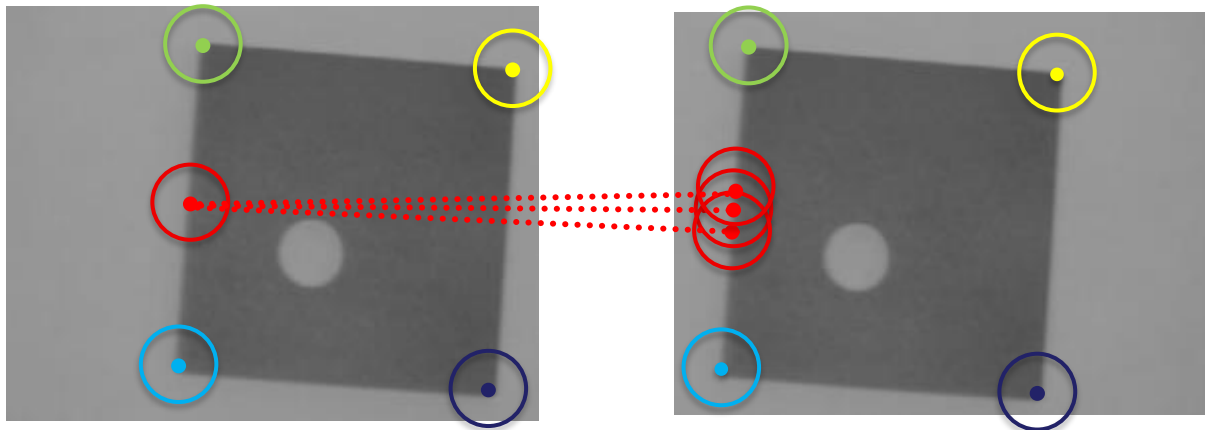
- **Pixels exhibiting a large variation along all directions are more amenable to establishing reliable correspondences.**
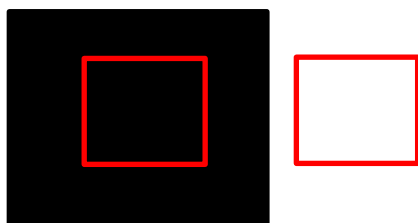
Samuele Salti, Luigi Di Stefano

Moravec Interest Point Detector

- A major early proposal was due to Moravec [1]:

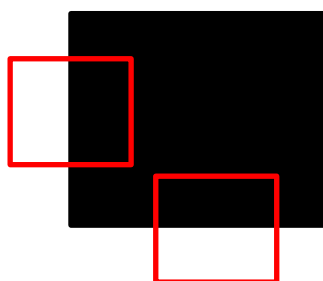$$C = min \left\{ \sum_{i,j \in w} \left(I(i+m, j+n) - I(i,j) \right)^2 \right\}; \; m,n \in \{-1,0,1\}, \neq (0,0)$$
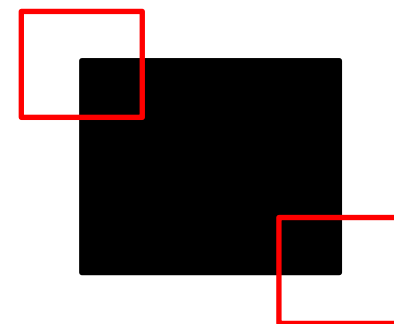
uniform region: no change in all directions

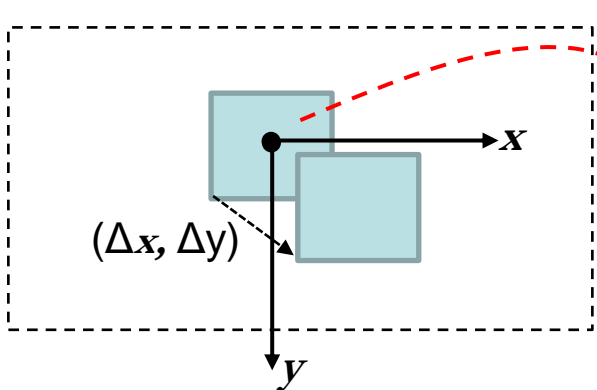**edge:** no change along the edge direction

corner: significant change in all directions.

Samuele Salti, Luigi Di Stefano

Harris Corner Detector (1)

- Harris&Stephens [2] proposed to rely on a continuous formulation of the Moravec's "error" function. Denoted as (Δx, Δy) a generic infinitesimal shift, such a function may be written as :

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

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

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

$$I\left(x+\Delta x, y+\Delta y\right) - I\left(x, y\right) \cong \frac{\partial I\left(x, y\right)}{\partial x}\Delta x + \frac{\partial I\left(x, y\right)}{\partial y}\Delta y = I_x\left(x, y\right)\Delta x + I_y\left(x, y\right)\Delta y$$
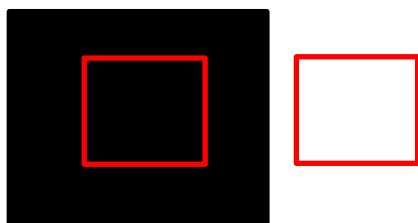
Samuele Salti, Luigi Di Stefano

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

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

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

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

$$E(\Delta x, \Delta y) = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$
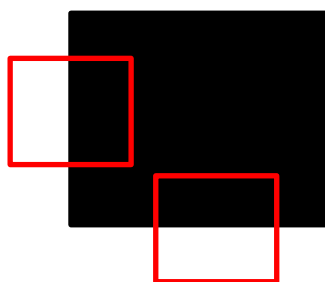
Samuele Salti, Luigi Di Stefano

M encodes the local image structure around the considered pixel. To understand why, let us hypothesize that M is a diagonal matrix:

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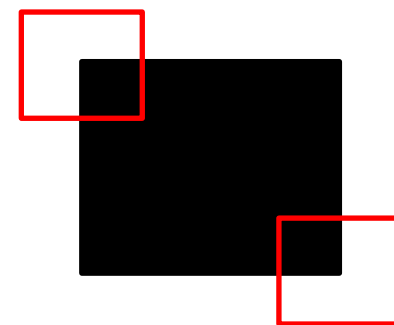$$

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

$\lambda_1 \gg \lambda_2$: **Edge**

$\lambda_1, \lambda_2 \uparrow$: **Corner**
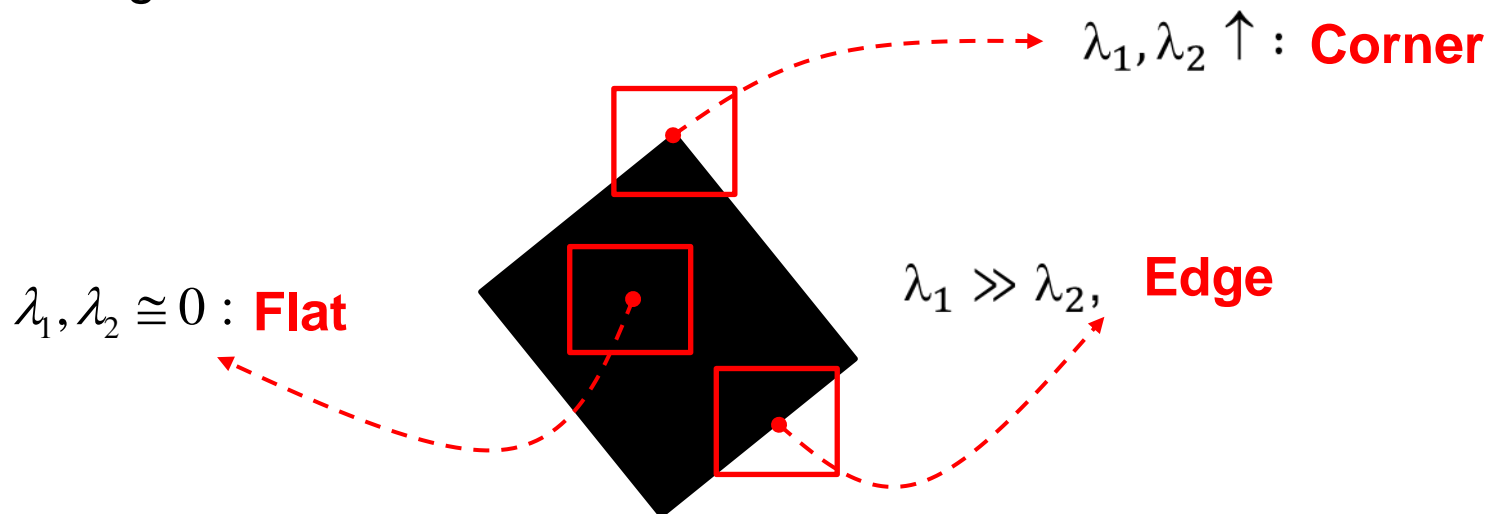

Samuele Salti, Luigi Di Stefano

Harris Corner Detector (4)

Indeed, 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 ortogonal 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.
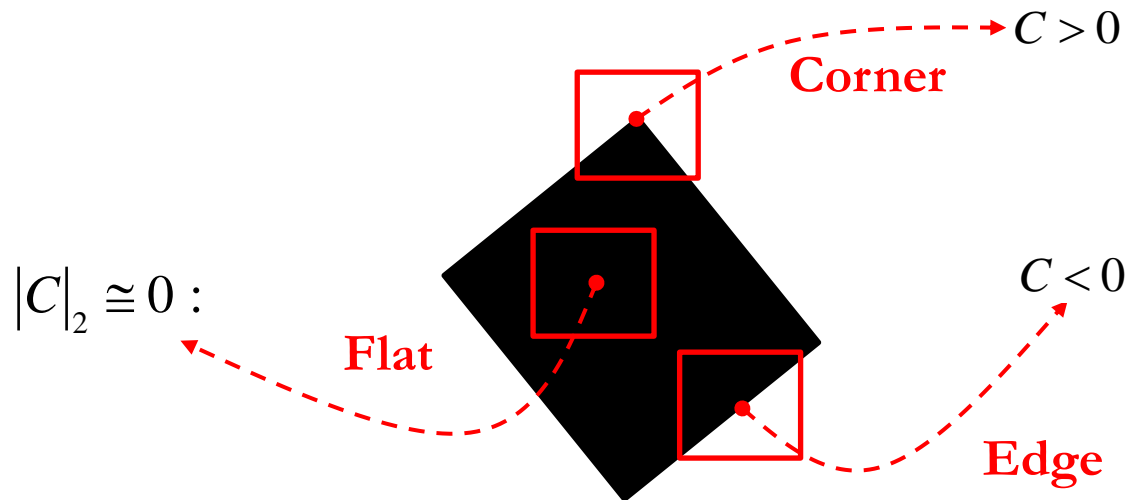

$\lambda_1, \lambda_2 \uparrow :$ **Corner**

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

$\lambda_1 \gg \lambda_2,$ **Edge**

Samuele Salti, Luigi Di Stefano

Computing the eigenvalues of M may be slow. Thus, Harris&Stephens propose to compute a more efficient "cornerness" function:

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

Analyis of the above function would show that:

$C > 0$

Corner

$|C|_2 \cong 0 :$

$C < 0$

Flat

Edge
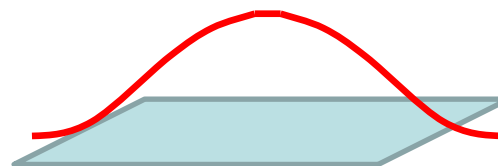
Samuele Salti, Luigi Di Stefano

Harris Corner Detector (6)

- 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.

- 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
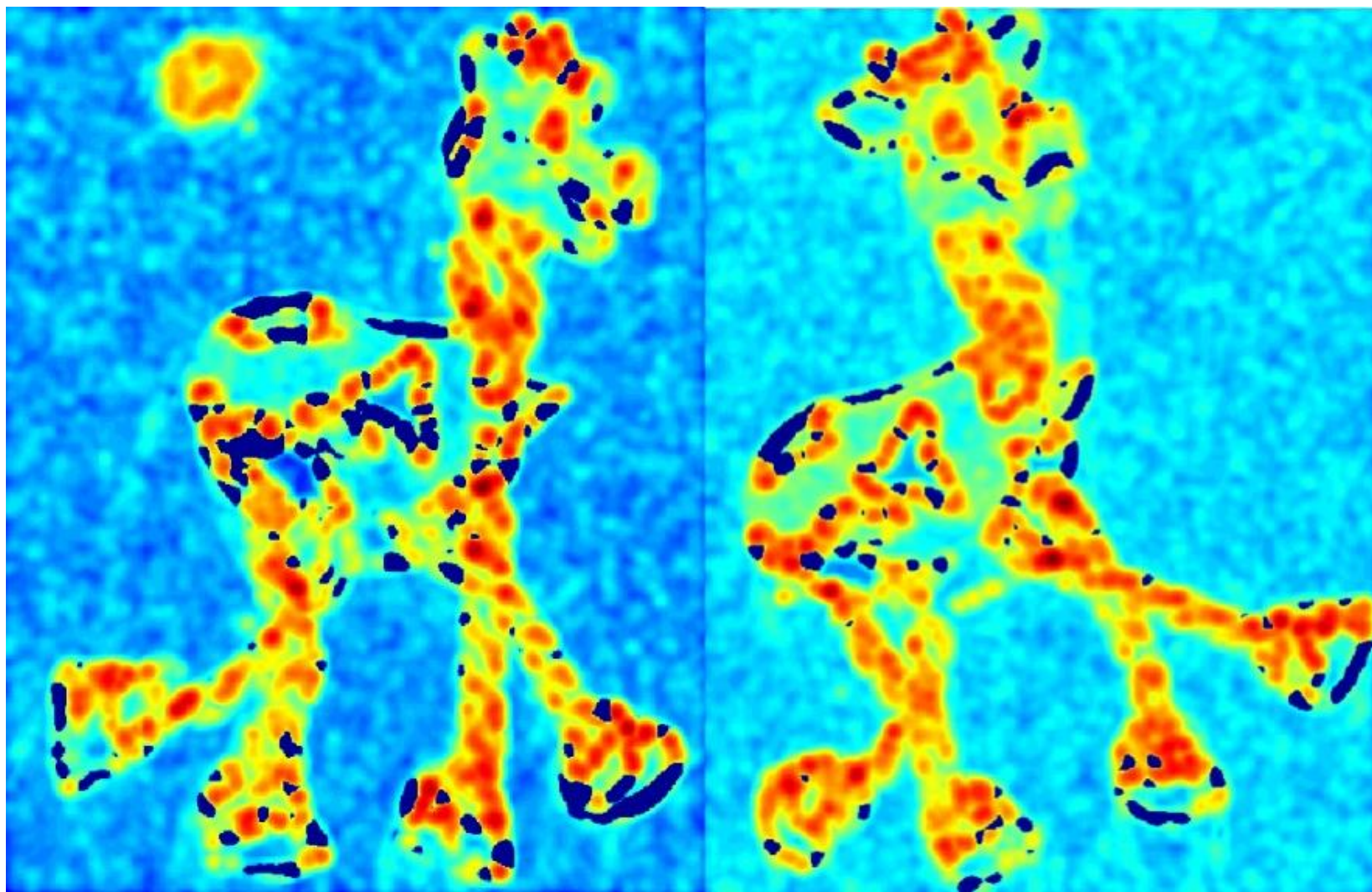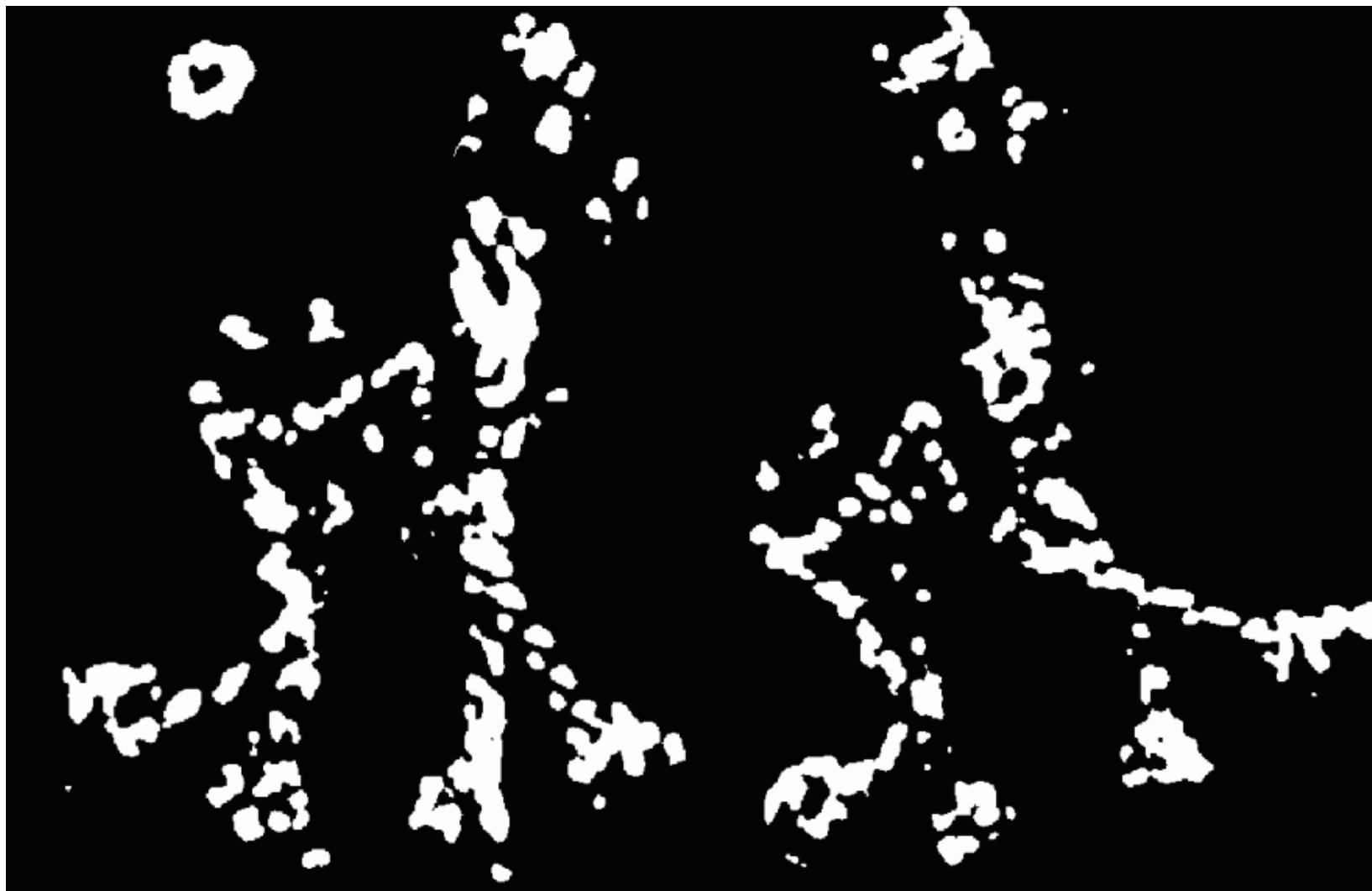
Samuele Salti, Luigi Di Stefano

Samuele Salti, Luigi Di Stefano

One Example (2)

Samuele Salti, Luigi Di Stefano

Samuele Salti, Luigi Di Stefano

Samuele Salti, Luigi Di Stefano

Samuele Salti, Luigi Di Stefano
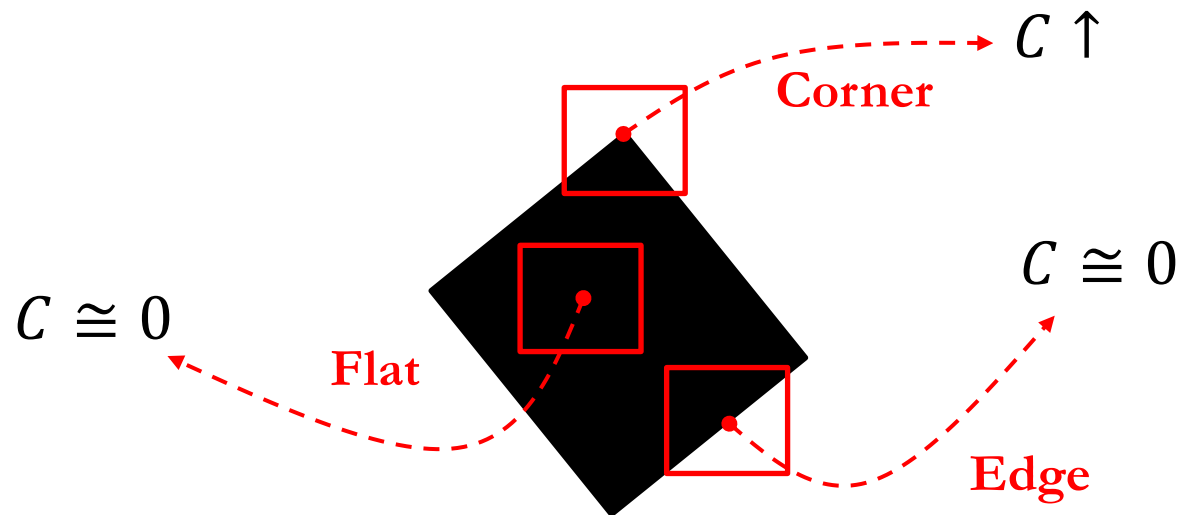
The Shi-Tomasi Corner Detector

A popular variant of the Harris Corner Detector is due to Shi and Tomasi [3] who showed that a different *cornerness* function provides better results when tracking corner points across video frames (*good features to track*):

$$C = \min(\lambda_1, \lambda_2)$$

Samuele Salti, Luigi Di Stefano

Invariance Properties (1)

- Rotation invariance? Yes, as the eigenvalues of M are invariant to a rotation of the image axes and thus so is Harris cornerness function.
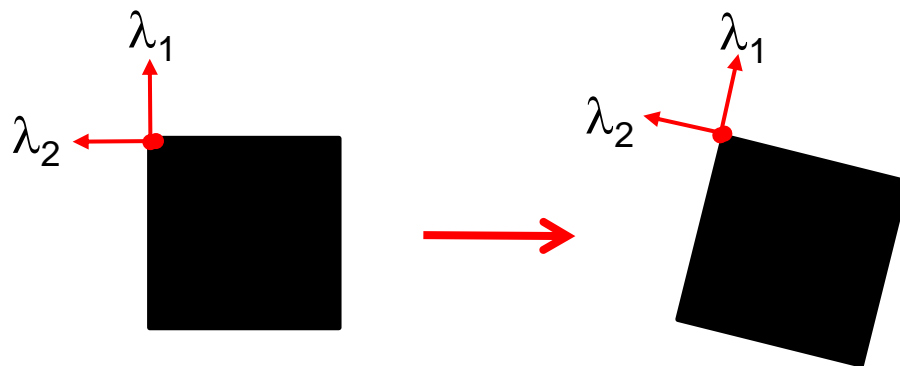
- Invariance to Intensity changes ? Limited
  - *Yes*, to an additive *bias* (*I' = I + b)* due to the use of derivatives.

  - *No*, to multiplication by a *gain* factor *(I' = a⋅I),* as derivatives get multiplied by the same factor.

Samuele Salti, Luigi Di Stefano
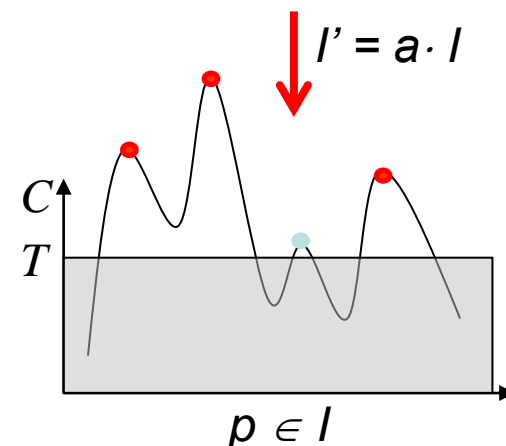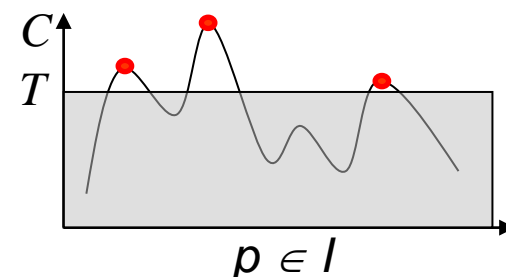
- Scale invariance?
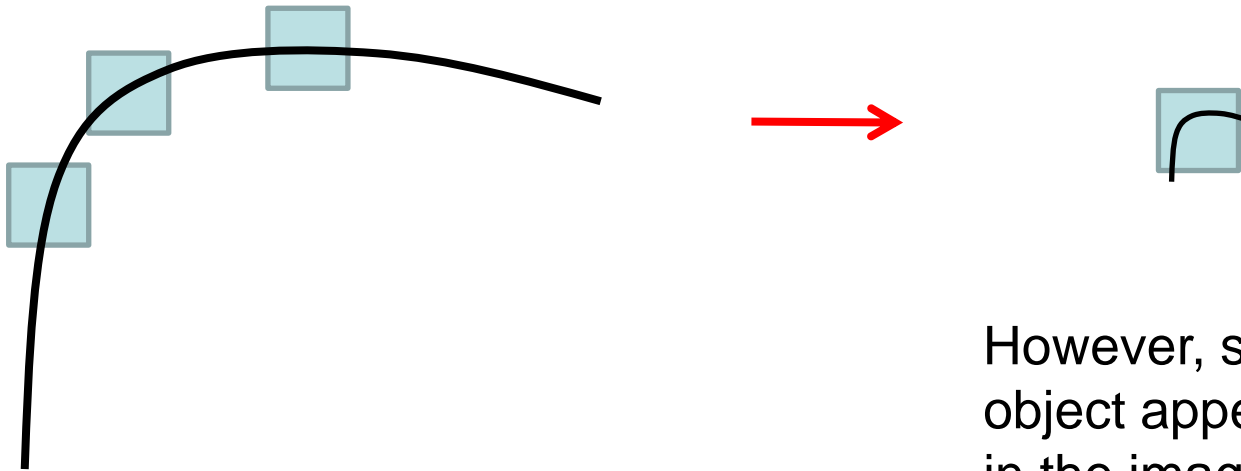
Given the chosen size of the detection window, all the points along the border are likely to be classified as **edges**.

However, 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.

Samuele Salti, Luigi Di Stefano

Scale Invariance (1)

Typically, an image contains features at different scales, i.e. points that stand-out as interesting as long as a proper neighnourhood size is chosen to evaluate the chosen interestingness criterion.
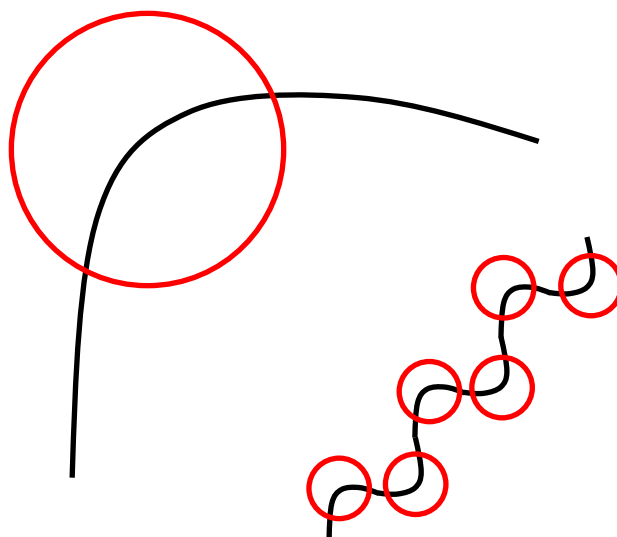
Thus, detecting all features calls for a "tool" capable of analyzing the image across the whole range of the scales deemed as relevant.

Samuele Salti, Luigi Di Stefano
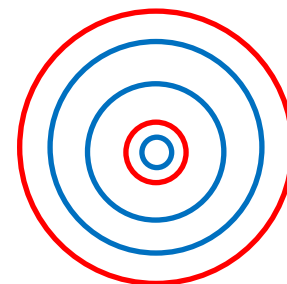
Scale Invariance (2)

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

Close

Farther

Very Far Away
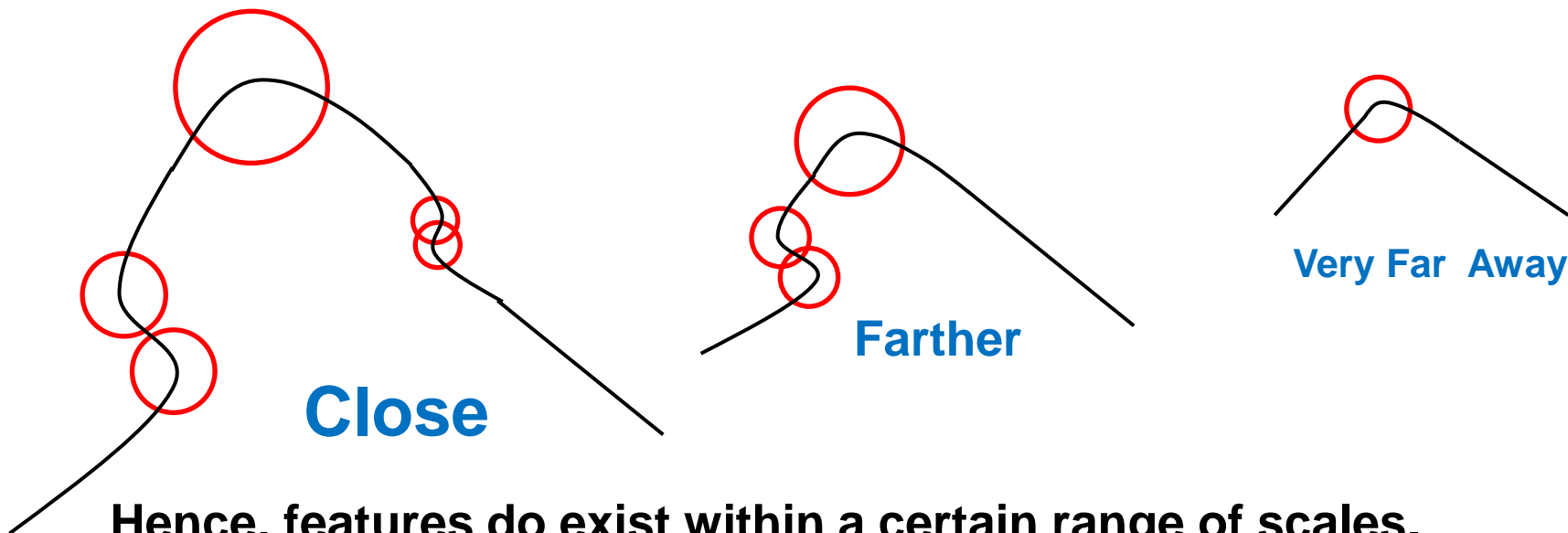
Hence, features do exist within a certain range of scales.

The previously mentioned tool would thus also allow for finding homologous features despite their showing-up at different scales in different images: the same feature would simply be detected at different steps within a multi-scale image analysis process.

Samuele Salti, Luigi Di Stefano

Scale-Space

- **Scale invariance is the main issue addressed by *second generation* local invariant features.**

- **One key finding concerns applying a fixed tool size on scaled (i.e. smaller and smaller) and increasingly *blurred* versions of the input image:**

Small Features   **Larger Features**   **Very Large Features**

- **When satisfying some specific mathematical properties, this representation is known as *Scale Space*.**

Samuele Salti, Luigi Di Stefano

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. Morever, one would not wish to create new structures while smoothing the images.

- Several researchers, such as, prominently, Witkin [4] and Koenderink [5], have studied the problem and shown that a *Scale-Space* must be realized by *Gaussian Smoothing*.

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

- Thus, a *Scale-Space* is created by repeatedly smoothing the original image with larger and larger Gaussian kernels (or, equivalently, by solving the 2D diffusion PDE over time starting from the original image).

Samuele Salti, Luigi Di Stefano

Feature Detection & Scale Selection

- **In itself, the Gaussian Scale-Space is only a tool to represent the input image at different scales. However, it neither includes any criterion to detect features nor to select their characteristic scale. As features do exist across a range of scales, the latter concept deals with establishing 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 is due to Lindberg [6], who proposed to compute suitable combinations of *scale-normalized derivatives* of the *Gaussian Scale-Space* (*normalized Gaussian derivatives*) and find their extrema.**

Samuele Salti, Luigi Di Stefano

Scale-Normalized LOG

Between Lindberg's proposed functions is the scale-normalized Laplacian of Gaussian (LOG):

$$F(x, y, \sigma) = \sigma^2 \cdot \nabla^2 L(x, y, \sigma) = \sigma^2 \cdot (\nabla^2 \ G(x, y, \sigma) * I(x, y))$$
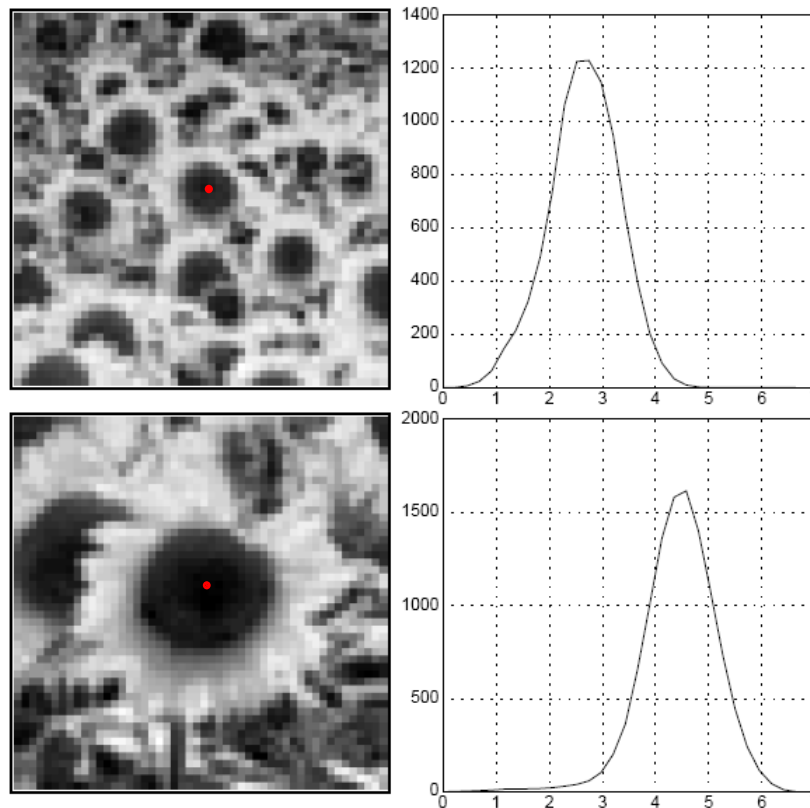
Considering the centers (red points) of the two dark blobs, it can be observed that the extremum of the scale-normalized LOG is found at a larger scale for the larger blob.

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

Samuele Salti, Luigi Di Stefano

Multi-Scale Feature Detection

Features (blob-like) and scales detected as extrema of the scale-normalized LOG.

Samuele Salti, Luigi Di Stefano
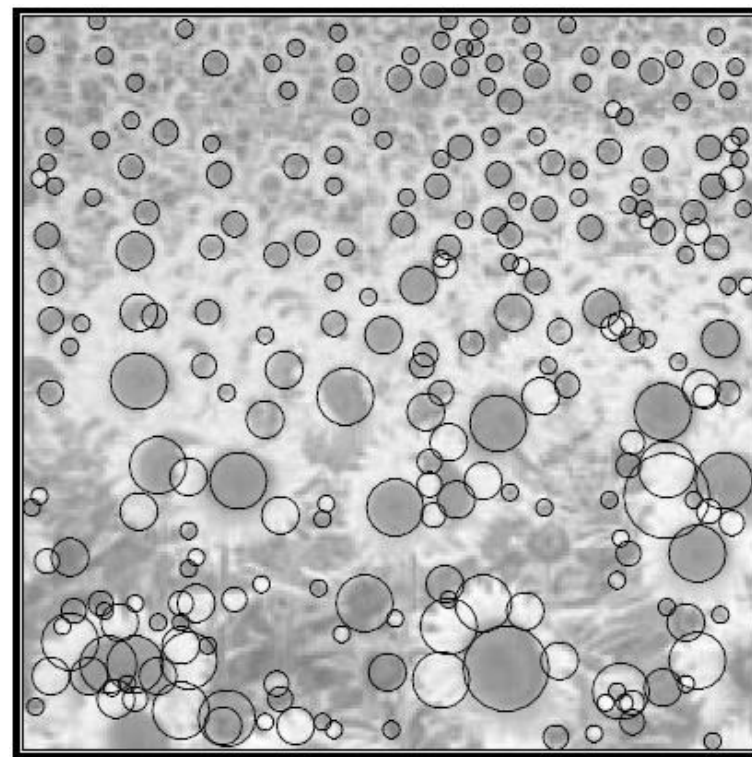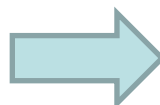
DoG Detector

- Lowe [7] proposes to detect *keypoints* by seeking for the extrema of the DoG (*Difference of Gaussian*) function across the *(x,y,σ)* domain:

$$DoG(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

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

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

- As (*k*-1) is a constant factor, it does not influence extrema location. As such, the choice of k is not critical.

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

Samuele Salti, Luigi Di Stefano
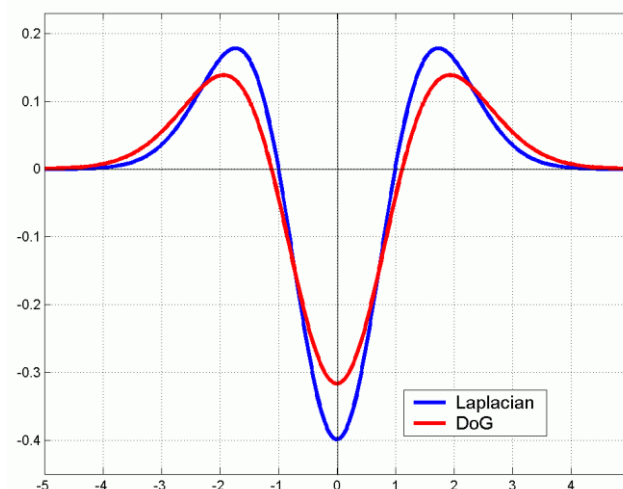
Computation of $DoG(x,y,\sigma)$

In general, in an octave we sample s scales:

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

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

2σ

$k^3\sigma$

$k^2\sigma$

$k\sigma$

σ

$(s+3)$ Gaussian filtered images per octave

$L(x,y,\sigma)$ is downsampled at ¼ resolution before computation of the next octave

Scale (next octave)

Scale (first octave)

$k^{s+1}\sigma$

σ/k

$L(x,y,\sigma)$

$DoG(x,y,\sigma)$

Extrema detection on s DoG images per octave
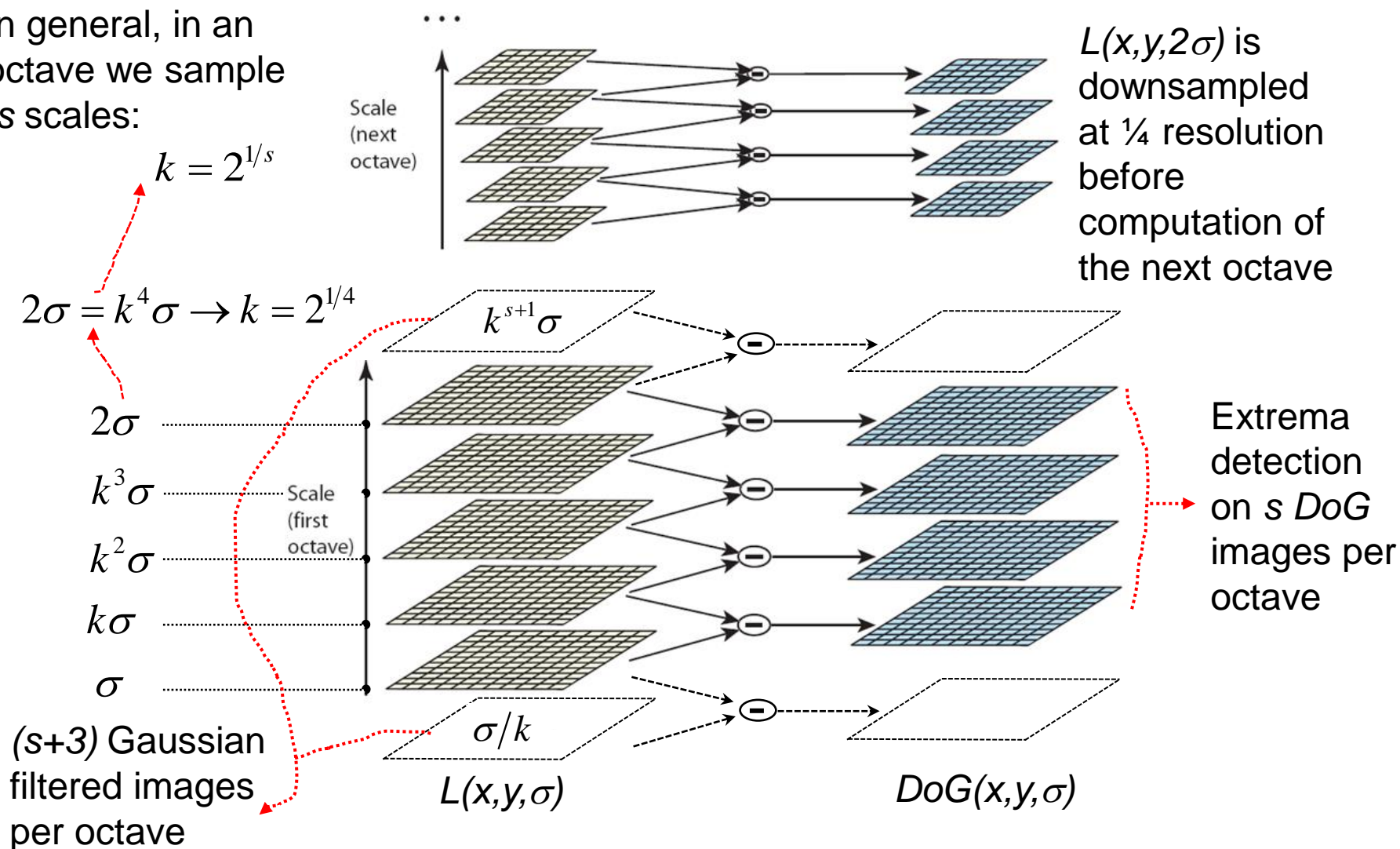
Keypoint Detection and Tuning

- **Keypoint detection:**
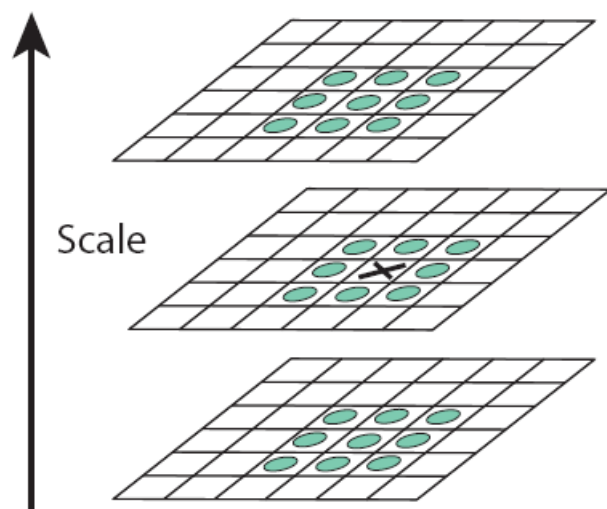
Scale

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.

S=3

- **Parameter Tuning:**

$\sigma = 1.6$ *(initial σ at each new octave)*

The input image is enlarged by a factor of 2 in both dimensions.

Samuele Salti, Luigi Di Stefano

Accurate Keypoint Localization

- To localize *keypoints* more accurately, the *DoG* function can be approximated around each extrema by its second degree Taylor expension:

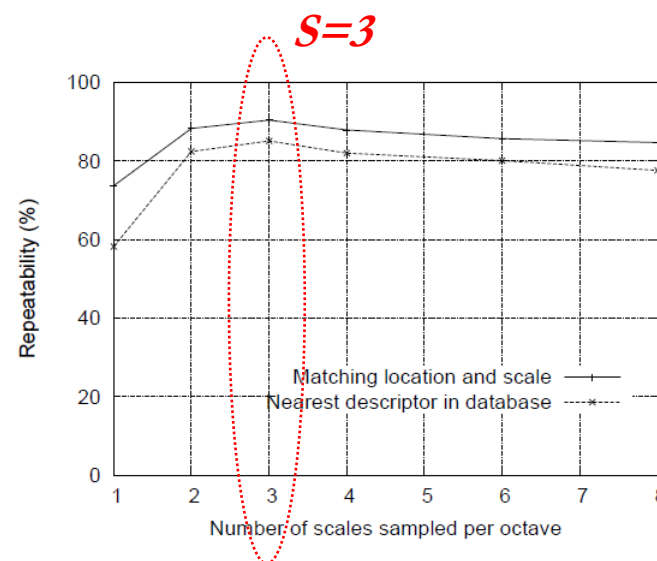$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

Displacement wrt the (x,y,σ) position of the found extremum

DoG at the found extremum

Gradient of the DoG function at the found extremum

Hessian matrix of the DoG function at the found extremum

- The new extremum (either maximum or minimum) can be localized by imposing the gradient of the approximated function to be zero:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}$$

- If the displacement $\hat{\mathbf{x}}$ turns out large (>0.5 in at least one dimension), the extremum is re-assigned to the closest discrete position and the approximation computed again. Once the displacement is small, the procedure is stopped and the found displacement provides sub-pixel interpolation wrt to the last discrete position.

Samuele Salti, Luigi Di Stefano

Pruning of Unstable *Keypoints*

- Extrema featuring weak *DoG* response turn out scarcely repeatable. They can be pruned by estimating the *DoG* at a found extremum $\hat{\mathbf{x}}$

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\,\hat{\mathbf{x}}.$$

and thresholding its magnitude: $|\,D(\hat{\mathbf{x}})\,| < \mathsf{T}_{DoG}$.

- Lowe also notices that unstable keypoints featuring a sufficiently strong *DoG* may be found along edges and devises a further pruning step based on the eigenvalues of the Hessian, which are proportional to the principal curvatures of the intensity surface. Thus, the higher is the ratio between the larger and smaller eigenvalues, *r*, the more similar to an edge rather than a blob is the considered pixel. Similarly to Harris', explicit computation of the eigenvalues may be avoided:
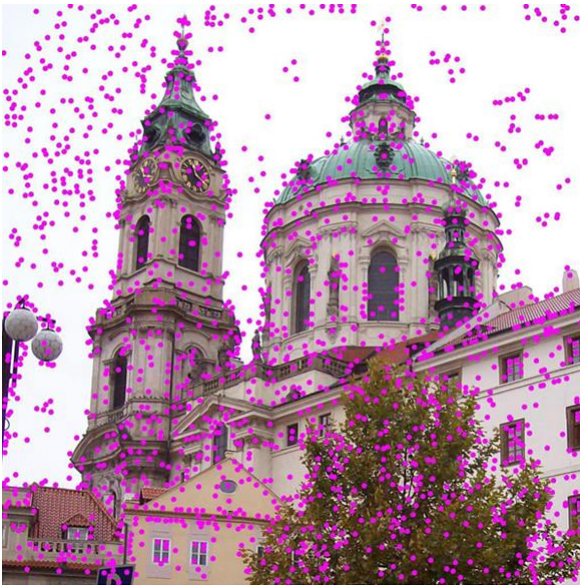
larger eigenvalue

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} > \mathsf{T}_{edge}$$

smaller eigenvalue

This increasing function of r can be thresholded to prune edges

Samuele Salti, Luigi Di Stefano

Exemplar DoG keypoints (1)
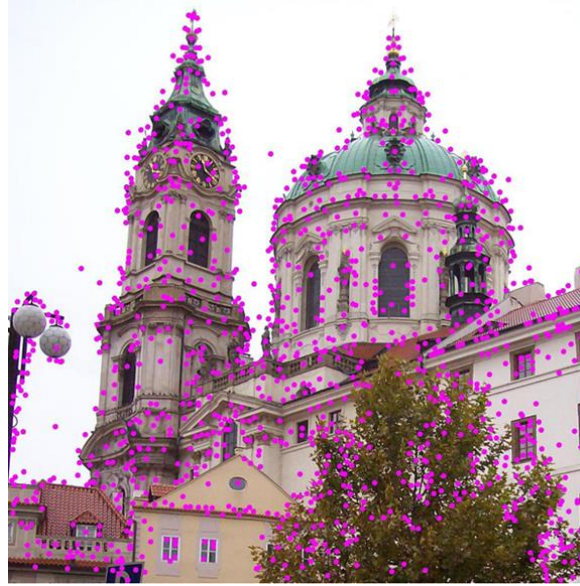
a)   b)   c)

a) **DoG extrema**
b) **Keypoints after pruning weak responses**
c) **Final keypoints after pruning those located along edges.**

Samuele Salti, Luigi Di Stefano

Exemplar DoG keypoints (2)

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

Samuele Salti, Luigi Di Stefano

Scale and Rotation Invariant Description

- Most keypoint detection algorithms follow an approach akin to Lowe's (i.e. finding extrema across a stack of images computed while increasing a scale parameter).
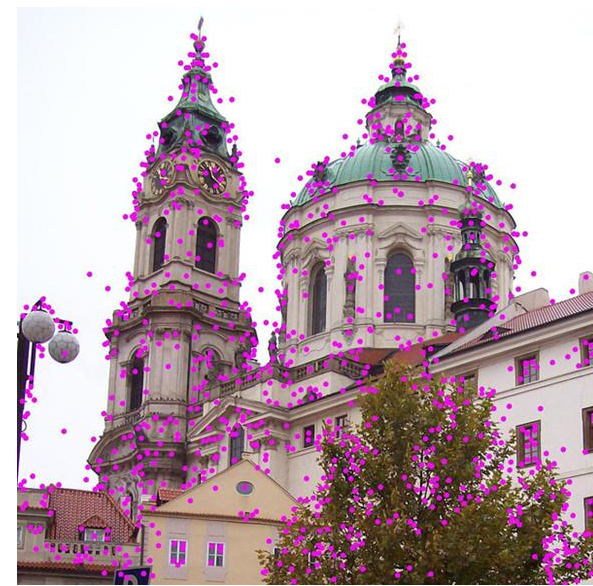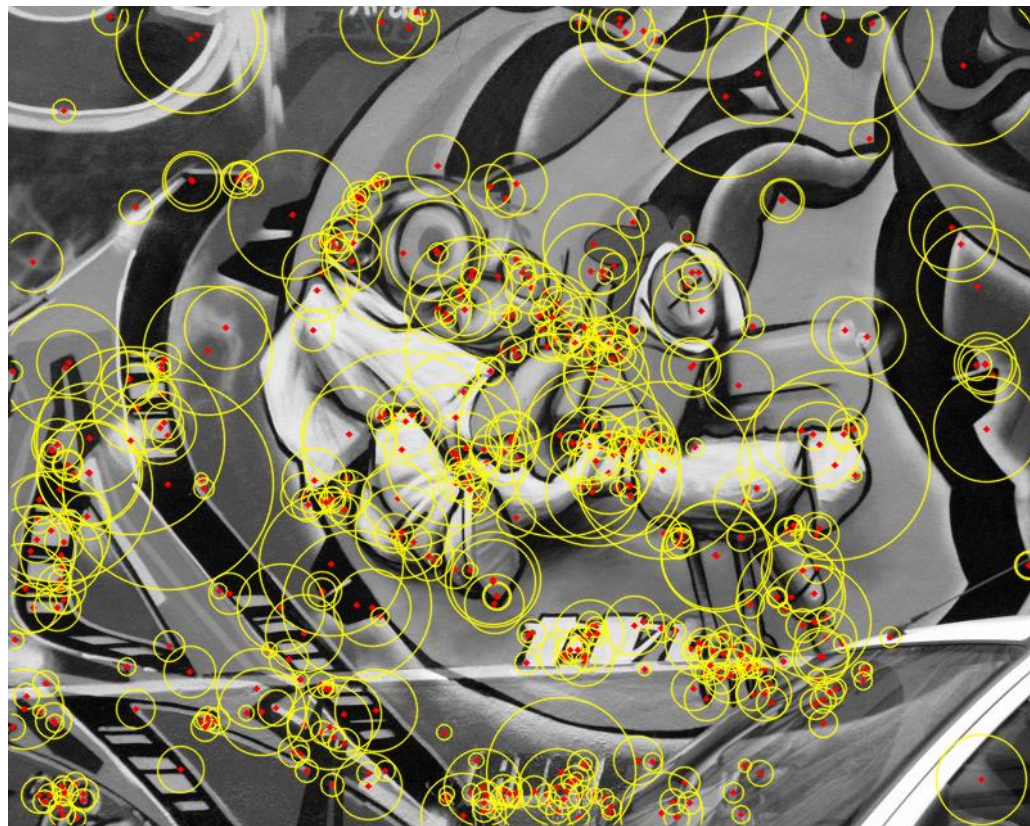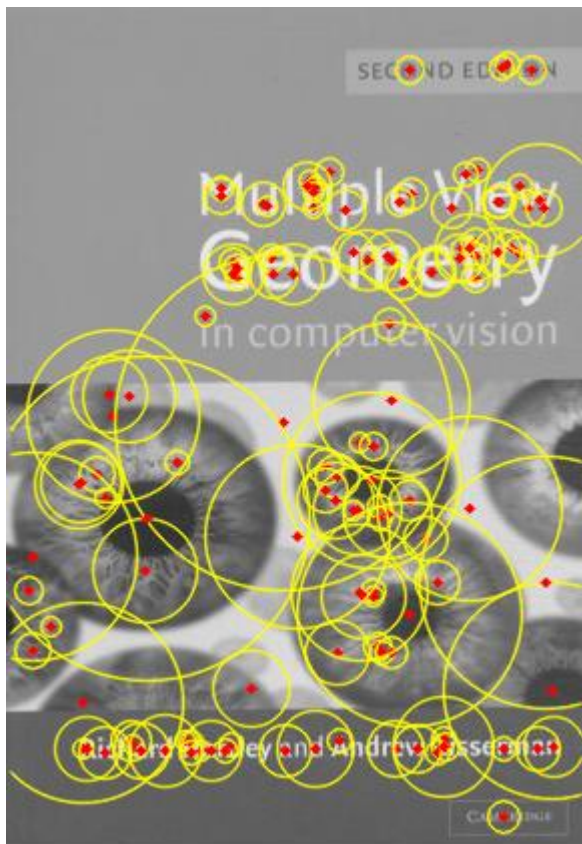
- Then, once each keypoint has been extracted, a surrounding patch is considered to compute its descriptor. A desirable property concerns such descriptor being scale and rotation invariant.

- To attain scale invariance, the patch is taken from the stack image ($L(x,y,\sigma)$ in Lowe's) that correspond to the characteristic scale.

- To attain rotation invariance, a canonical (aka characteristic) patch orientation is computed, so that the descriptor can then be computed on a canonically-oriented patch.

Samuele Salti, Luigi Di Stefano

Canonical Orientation (1)

- Lowe proposes to compute the canonical orientation of DoG keypoints as follows [7]:

  - Given the keypoint, the magnitude and orientation of the gradient are computed at each pixel of the associated Gaussian-smoothed image, *L*:

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

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

  - Then, an orientation histogram (with bin size equal to 10°) is created by accumulating the contributions of the pixels belonging to a neighborhood of the keypoint location.

  - The contribution of each such 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 denoting the scale of the keypoint.

Samuele Salti, Luigi Di Stefano

Canonical Orientation (2)

- The characteristic orientation of the keypoint is given by the highest peak of the orientation histogram.

- Moreover, other peaks higher than 0.8 of the main one would be kept as well. Accordingly, 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 ($\approx$ 15% of the keypoints), though.

$$\sigma_i = i \cdot \Delta\sigma = i \cdot (\pi/18)$$
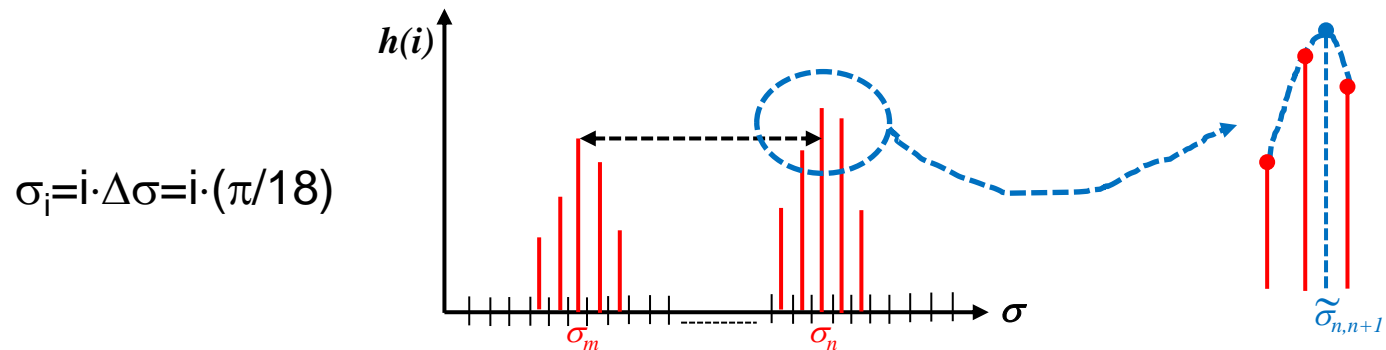
- Finally, a parabola is fit in the neighborhood of each peak to achieve a more accurate estimation of the canonical orientation. Purposely, the two adjacent bins to the found peak are considered.

Samuele Salti, Luigi Di Stefano

SIFT Descriptor (1)

The SIFT (Scale Invariant Feature Transform) descriptor is computed as follows [7]. A 16x16 <u>oriented</u> pixel grid around each keypoint is considered. This is further divided into 4x4 regions (each of size 4x4 pixels) and a gradient orientation histogram is created for each region. Each histogram has 8 bins (i.e. bin size 45°). Gradients are rotated according to the canonical orientation of the keypoint. Each pixel in the region contributes to its designated bin according to gradient magnitude as well as to a Gaussian weighting function centred at the keypoint (with σ equal to half the grid size).


Image gradients   Keypoint descriptor   => Feature vector (128)
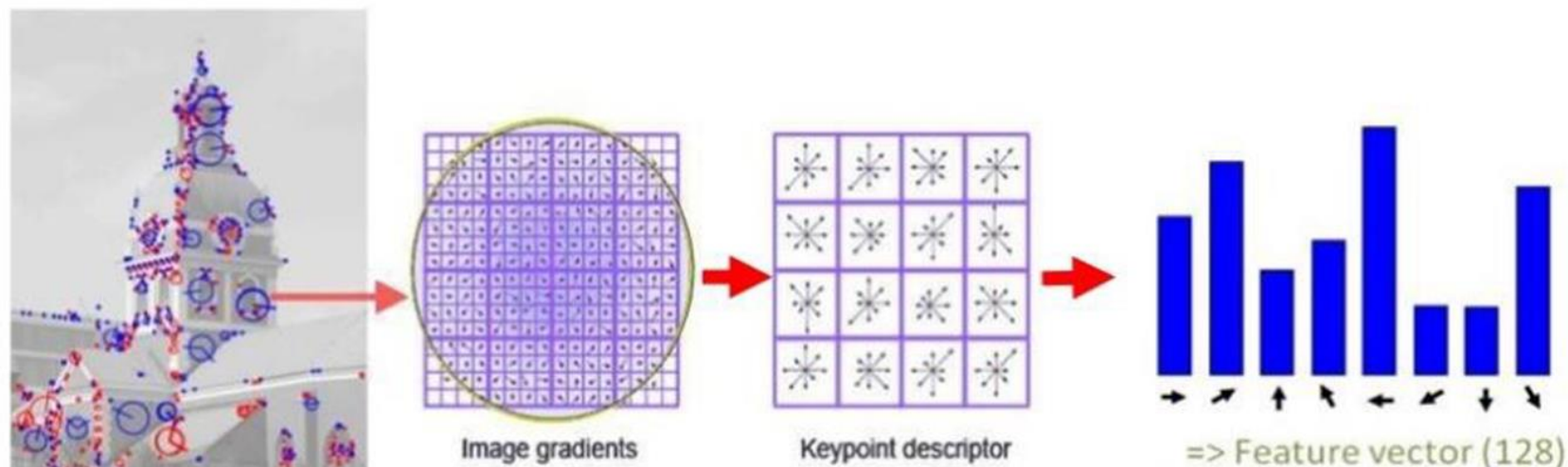
The descriptor size is given by the number of regions times the number of histogram bins per region, i.e. 4x4x8 = 128.

Samuele Salti, Luigi Di Stefano

SIFT Descriptor (2)

- 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.
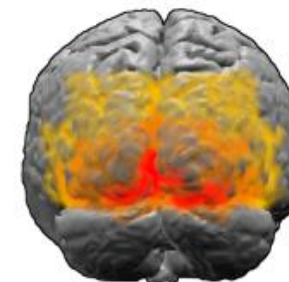
- To avoid boundary effects, a *soft* rather than *hard* assignment is employed in SIFT, whereby the contribution to two adjacent bins is weighted by the distance to the bin center:

$$\begin{cases} w_k = 1 - d_k \\ w_{k+1} = 1 - d_{k+1} \end{cases}$$
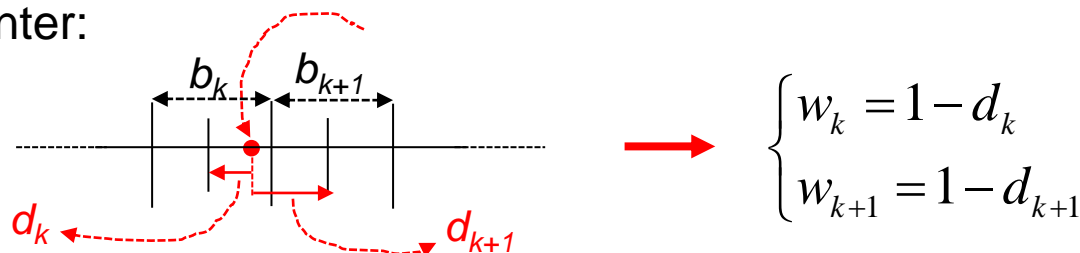
This is done within an histogram as well as between regions (the contribution is spread bilinearly between 4 adjacent regions). Hence, the overall scheme is referred to as *trilinear interpolation*.

- The descriptor is normalized to unit length to gain invariance wrt affine intensity changes. Then, to increase robustness to non-linear changes, all elements larger than 0.2 are saturated and the descriptor normalized again.

Samuele Salti, Luigi Di Stefano

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 in a metric space *M* and a query point *q* ∈ *M*, find the closest point in *S* to *q*.

- Without loss of generality, we assume that we wish to match the local features computed at run time from an image under analysis, aka target image (*T*), to those already computed from a reference image (*R*) or, equivalently, a set of reference images.

- As for each feature in *T* we look for the most similar one in *R:*

  - The features in *T* represent query points, *q.*
  - The features in *R* provide set *S.*
  - The metric space, *M*, is the descriptor space endowed by a distance function. When matching SIFT descriptors, $M=R^{128}$, the distance function being typically the Euclidean distance.

Samuele Salti, Luigi Di Stefano

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*. Generally, this is due to clutter and/or viewpoint changes.

- Hence, it turns out mandatory to enforce criteria to accept/reject a match found by the NN search process. Two such criteria are as follows;

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

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

Lowe [7]

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


Samuele Salti, Luigi Di Stefano
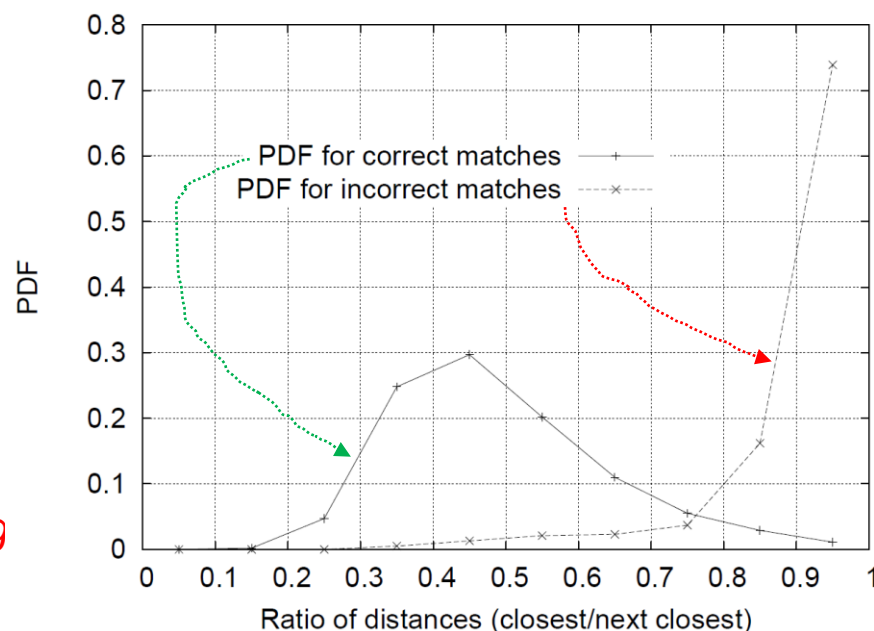
Efficient NN-Search

- Exhaustively searching for the NN of the *query* feature, *q*, has linear complexity in the size of *S*, which turns out often exceedingly slow.

- Thus, efficient *indexing techniques* borrowed from database management and information retrieval are deployed to speed-up the NN-search process.

- The main *indexing* technique deployed for feature matching is known as *k-d tree* [8]. In particular, the preferred approach is the approximate variant referred to as **BBF** (*Best Bin First*) proposed in [9]. Indeed, unlike the basic *k-d tree*, the BBF formulation is efficient also in high-dimensional spaces such as descriptor spaces (e.g., R^{128} for SIFT).

Samuele Salti, Luigi Di Stefano
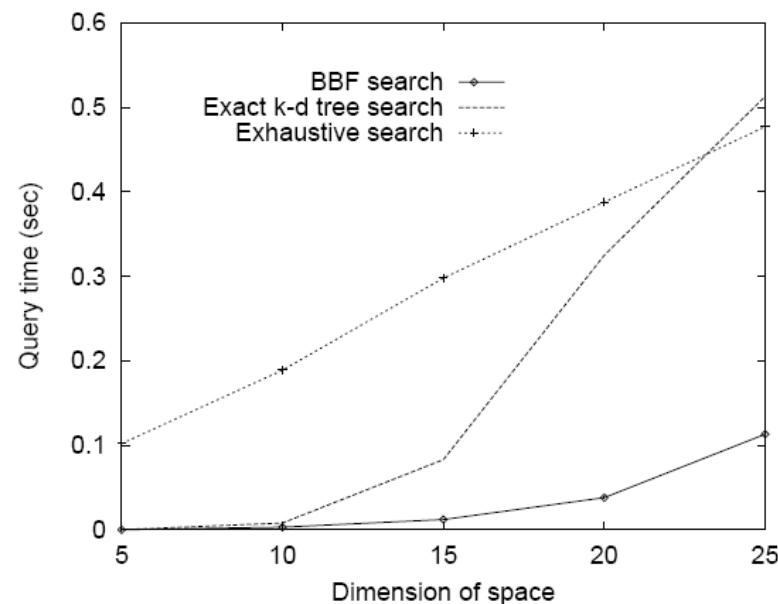
k-d Tree

- The *k-d* tree generalizes the *binary search tree* to the case of *k*-dimensional data.

- Each node defines a *split* of the data according to one of the *k* dimensions.

- During the search, the *query point, q,* traverses the tree from the *root* down to a *leaf* according to the the splits defined by nodes. The datum stored in the *leaf* is close to *q* though not guaranteed to be the NN.

- Therefore, the tree is traversed back from the *leaf* to the *root* to refine the search. During *backtracking,* only the sub-trees that may contain closer data are explored.

- If the tree is *balanced*, the initial search requires $\log_2(size(S))$ comparisons, which yields a large speed-up as long as backtracking involves visiting relatively few nodes.

d(q,s_j)

Samuele Salti, Luigi Di Stefano
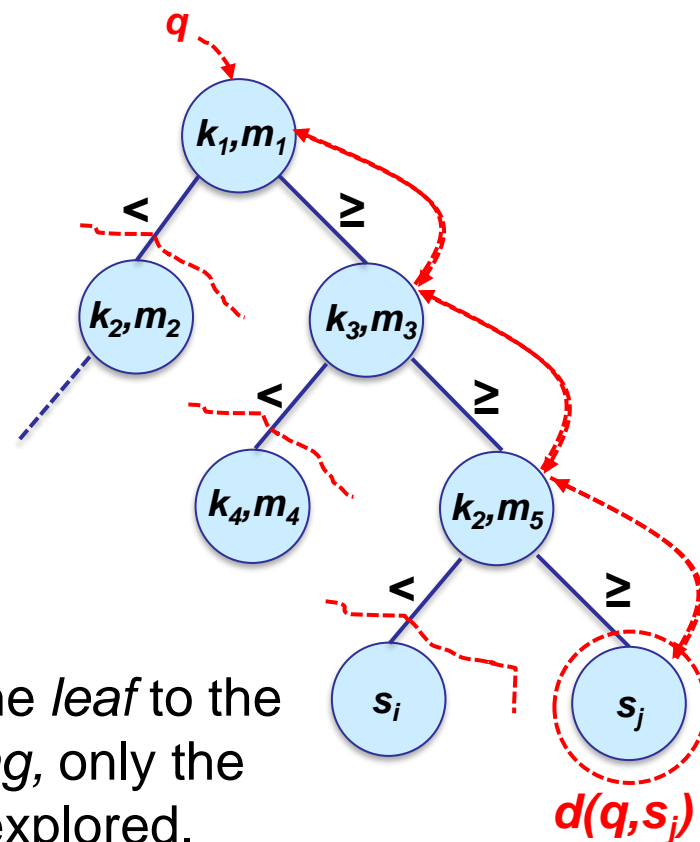
Building up the Tree

- The data are always split by choosing the dimension showing the highest variance. The datum taking the median value of such dimension defines the split.

- Thus, the root is created first. This splits the initial set of data, S, into two sub-sets, S_1 and S_2, of, roughly, the same sizes.

- The process is then applied recursively.
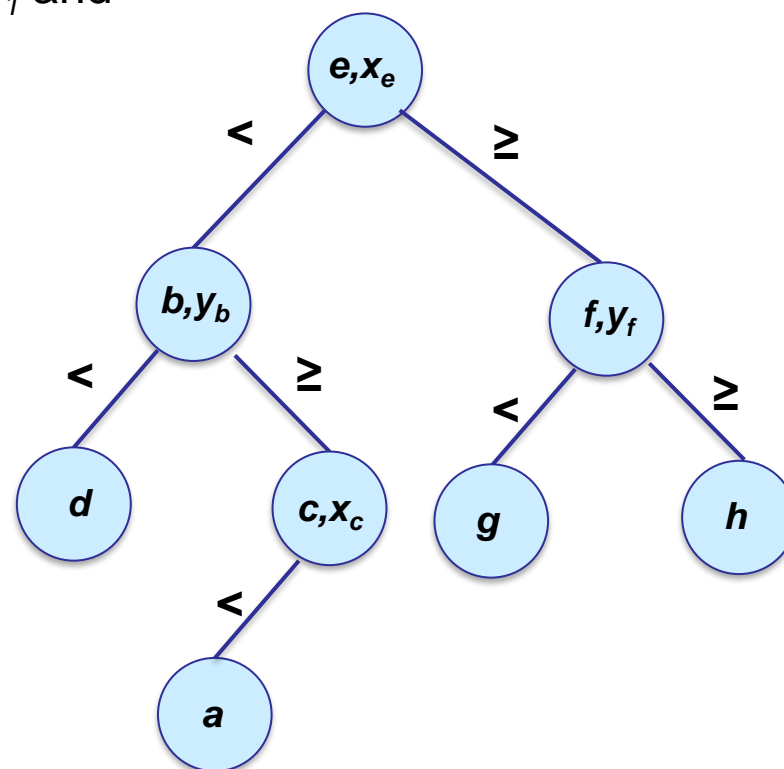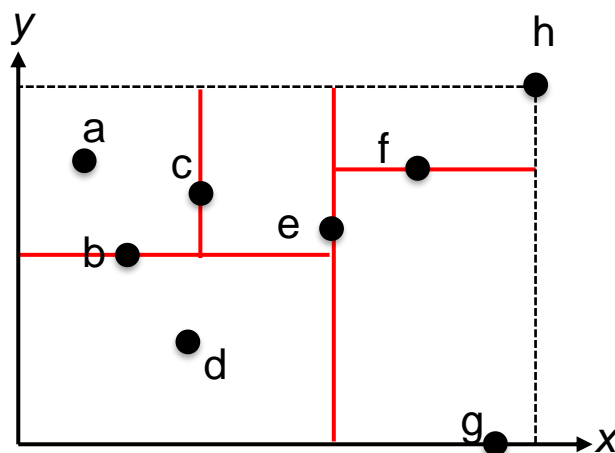
The k-d tree partitions the data space into bins adaptively: bins are smaller in higher density regions, larger in regions of lower data density.

NN Search (1)
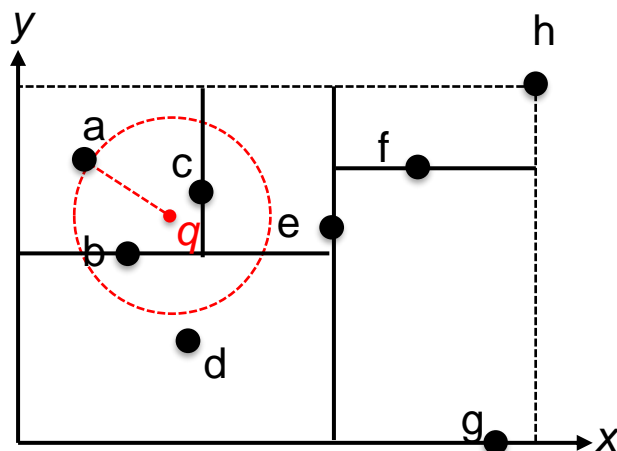
As the (hyper)sphere centred at *q* with radius *d(q,a)* does intersect the (hyper)planes separating the partition of the space associated with leaf *a* from its adjacent partitions, it may be necessary to explore these partitions (together with their parent nodes) to find the NN of *q*.

Samuele Salti, Luigi Di Stefano
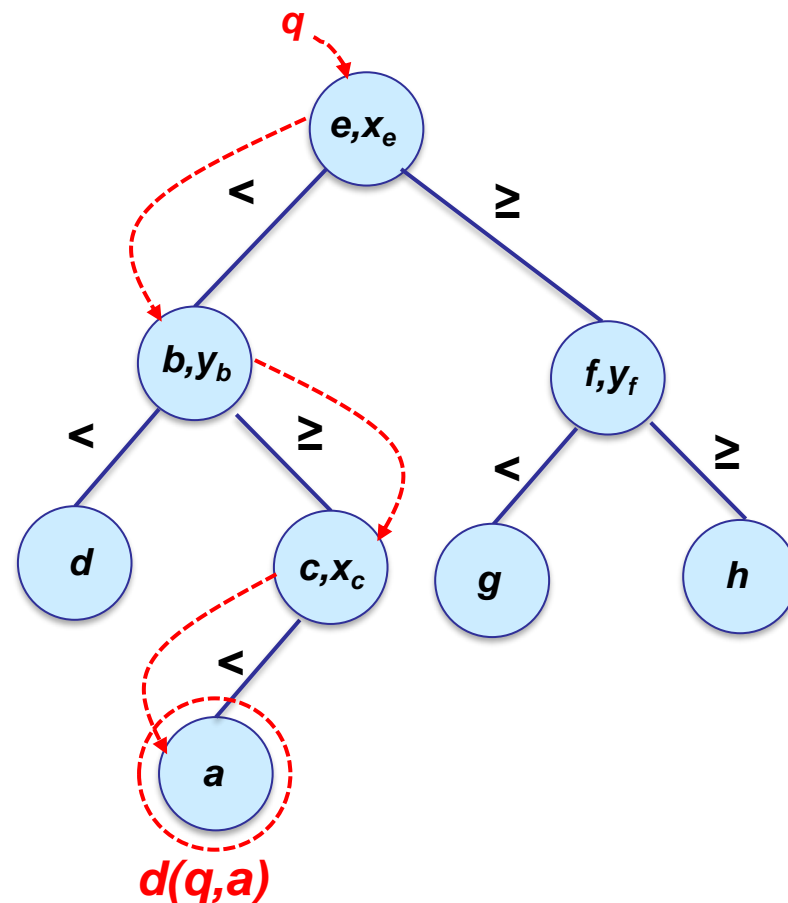
NN Search (2)

1. Given the query, q, the tree is traversed from the root to the closest leaf, denoted here as NN_{curr}, the associated distance given by $d(q, NN_{curr})$.

2. Backtracking:

   - The parent node to the found leaf is considered. If its distance to q is less than $d(q, NN_{curr})$, the parent node becomes the new NN_{curr} and $d(q, NN_{curr})$ gets updated accordingly.
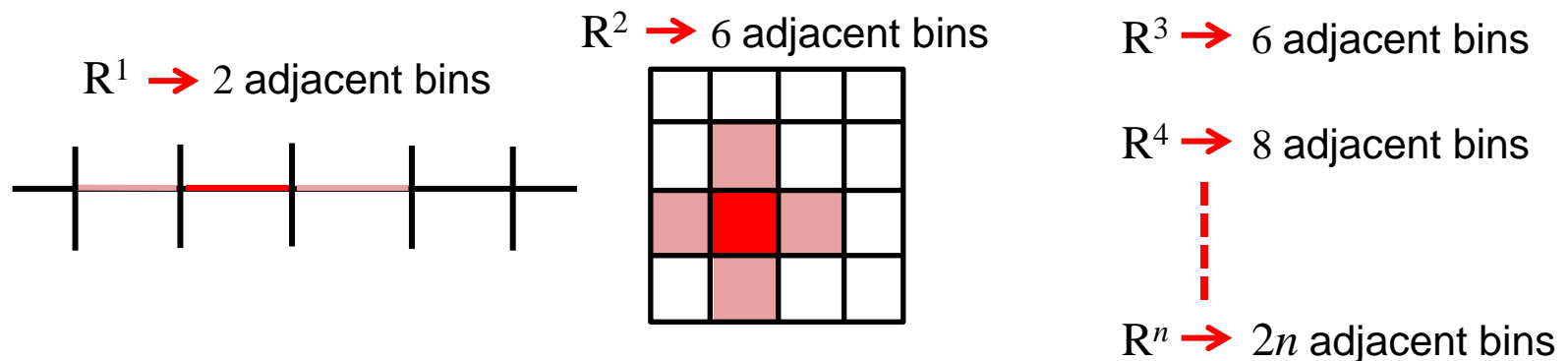
   - A check is carried out to determine whether the other half-space defined by the current parent node may contain or not a closer point. This depends on whether or not the hypersphere of radius $d(q, NN_{curr})$ centred at q intersect the splitting hyperplane.

     - Should the former be the case, the sub-tree whose root is the parent node is traversed down to the next closest leaf.

     - Conversely, the parent node to the current one is considered.

   - The process ends upon getting to the root of the tree, with NN_{curr} turning out the NN to q.

Samuele Salti, Luigi Di Stefano

The curse of dimensionality

- A *k-d tree* may be thought of as partitioning the space into "bins". During *backtracking*, the bins adjacent to that containing the found leaf might be examined to find the NN.

- However, as the dimensionality of data increases, so does the number of bins adjacent to a given one, which renders *k-d trees* inefficient (perhaps even slower than exhaustive search) with high-dimensional data.

R^1 → 2 adjacent bins

R^2 → 6 adjacent bins

R^3 → 6 adjacent bins

R^4 → 8 adjacent bins

R^n → $2n$ adjacent bins

- The issue comes from the inherent structure of the R^n space, thus it is hardly addressable.

Samuele Salti, Luigi Di Stefano

Approximate Search (Best Bin First)

Hence, Beis & Lowe in [9] have proposed an approximate search algorithm, referred to as *Best Bin First* (BBF), which deploys a *priority queue* and limits the maximum number of reachable leaves (E_{max}).

1. As with the standard algorithm, given the query, *q*, the tree is traversed down to the closest leaf, so to find the initial NN_{curr} and associated distance ($d(q, NN_{curr})$). **Moreover, though, the visited nodes are ordered in a *priority queue* according to their distance to *q*.**

2. Backtracking:

    –   The first element in the *priority queue* is extracted so to traverse its unexplored branch down to a leaf. During the process, the *priority queue* is always kept updated.

    –   The previous step is iterated until a fixed number of leaves (E_{max}) has been reached.

> The found *approximate* NN is the closest data point (NN_{curr}) after E_{max} leaves have been visited.

Samuele Salti, Luigi Di Stefano

BBF vs. *k-d* tree
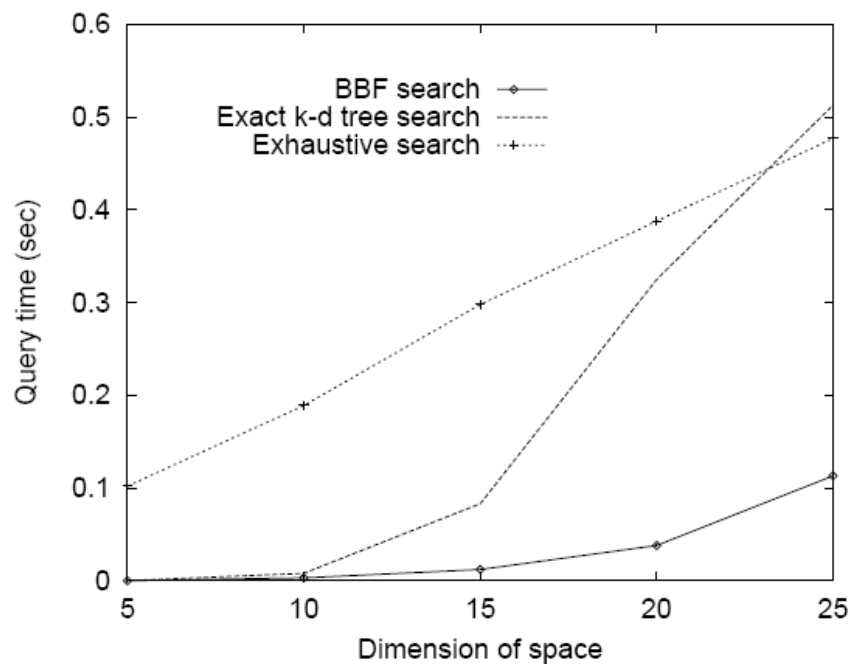
Efficiency
(30.000 uniformly
distributed data
points)

Accuracy
(E_{max}=200, n=12)

Samuele Salti, Luigi Di Stefano
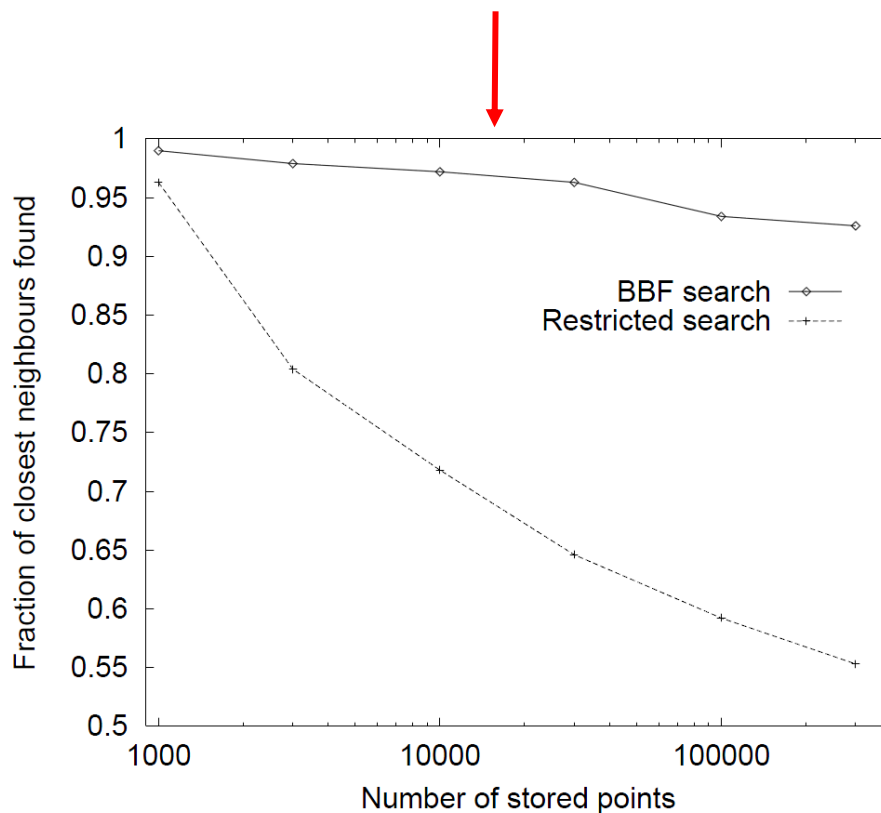
A few other major proposals

- A fast alternative to SIFT is SURF (*Speeded-Up Robust Features*) algorithm [10]. Blob-like features are detected through efficiently computable filters inspired by Lindberg's Gaussian derivatives [5]. The scale-space is achieved more efficiently by mean filtering.

- MSER (*Maximally Stable Extremal Regions*), proposed in [11], detects interest regions of arbitrary shapes, in particular approximately uniform areas either brighter or darker than their surroundings. Description of MESR region may then be carried out by SIFT [12].

- FAST (*Features from Accelerated Segment Test*) is a very efficient detector of corner-like features [13].

- 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 [14],[15],[16].

- Finally, BOLD (Bunch Of Lines Descriptor) [17] allows for deploying the local features paradigm with texture-less objects based on line segments extracted along edges.

Samuele Salti, Luigi Di Stefano

References (1)

1. H. Moravec. "Rover visual obstacle avoidance", *International Joint Conference on Artificial Intelligence*, 1981.

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

3. J. Shi, C. Tomasi (June 1994). "Good Features to Track". *IEEE Computer Vision Pattern Recognition Conference (CVPR)*, 1994.

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

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

6. T. Lindeberg. "Feature detection with automatic scale selection", *International Journal of Computer Vision*, 30(2):79–116, 1998.

7. D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision,* 20(2):91–110, 2004*.*

8. J. Friedman, J. Bentley, and R. Finkel. "An algorithm for finding best matches in logarithmic expected time", *ACM Trans. Math. Software,* 3:209–226, 1977.

9. J. Beis, D.G. Lowe. "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces", *IEEE Computer Vision Pattern Recognition Conference (CVPR)*, 1997.

References (2)

10. H. Bay, T. Tuytelaars, and L. Van Gool. "Speeded-Up Robust Features (SURF)", *Computer Vision and Image Understanding*, Volume 110, Issue 3, Pages 346–359, 2008.

11. J. Matas, O. Chum, M. Urban, and T. Pajdla. "Robust wide baseline stereo from maximally stable extremal regions", *British Machine Vision Conference (BMVC)*, 2002.

12. P.E. Forssen, D.G. Lowe "Shape Descriptors for Maximally Stable Extremal Regions ", *IEEE Conference on Computer Vision (ICCV)*, 2007.

13. E. Rosten, T. Drummond. "Machine learning for high-speed corner detection", *European Conference on Computer Vision (ECCV)*, 2006.

14. M. Calonder, V. Lepetit, C. Strecha, P. Fua. "BRIEF: Binary Robust Independent Elementary Features", *European Conference on Computer Vision (ECCV)*, 2006.

15. E. Rublee, V. Rabaud, K. Konolige, G. Bradski "ORB: an efficient alternative to SIFT or SURF", *IEEE Conference on Computer Vision (ICCV)*, 2011.

16. S. Leutenegger, M. Chli, R. Y. Siegwart "BRISK: Binary Robust Invariant Scalable Keypoints", *IEEE Conference on Computer Vision (ICCV)*, 2011.

17. F Tombari, A Franchi, L Di Stefano, "BOLD features to detect texture-less objects", *IEEE Conference on Computer Vision (ICCV)*, 2013.

Samuele Salti, Luigi Di Stefano