Università Ca' Foscari Venezia

Computer Vision

Point features

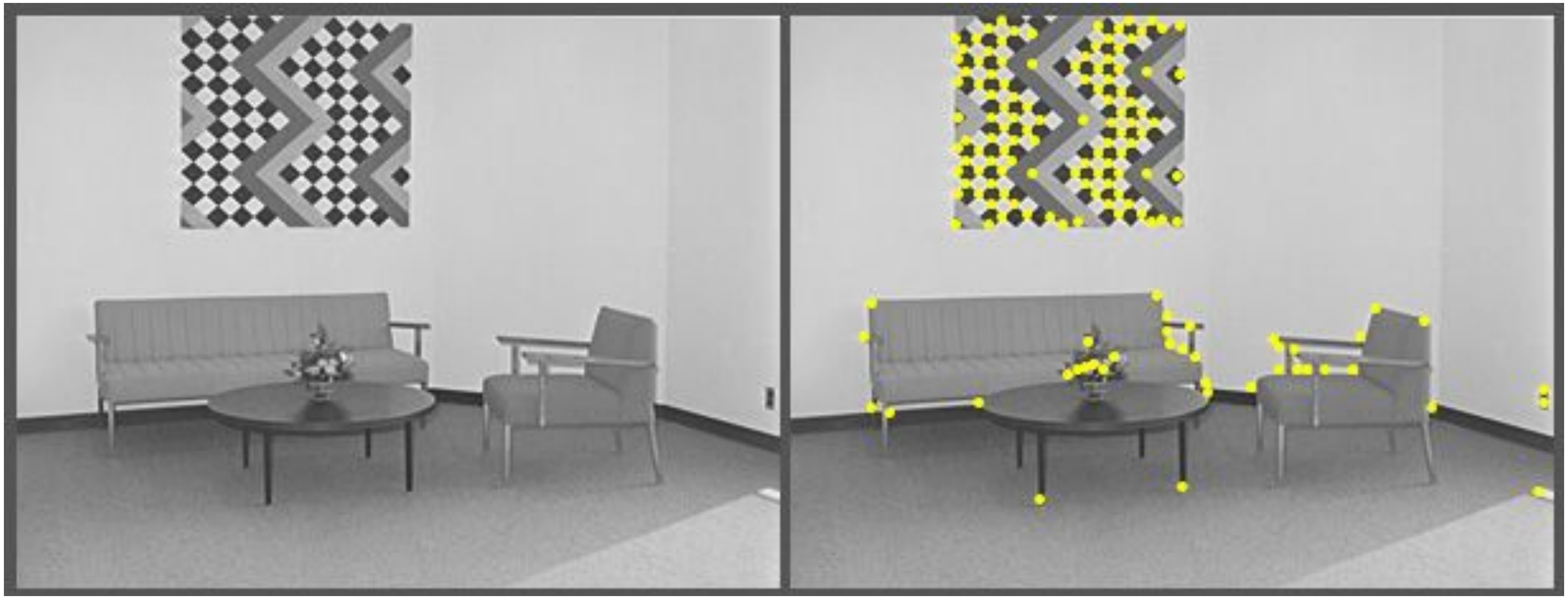
Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco
DAIS, Ca' Foscari University of Venice
Academic year 2017/2018

Università Ca' Foscari Venezia

Corners

Edge detectors perform poorly at corners.

Corners provide repeatable points for matching, so are worth detecting!

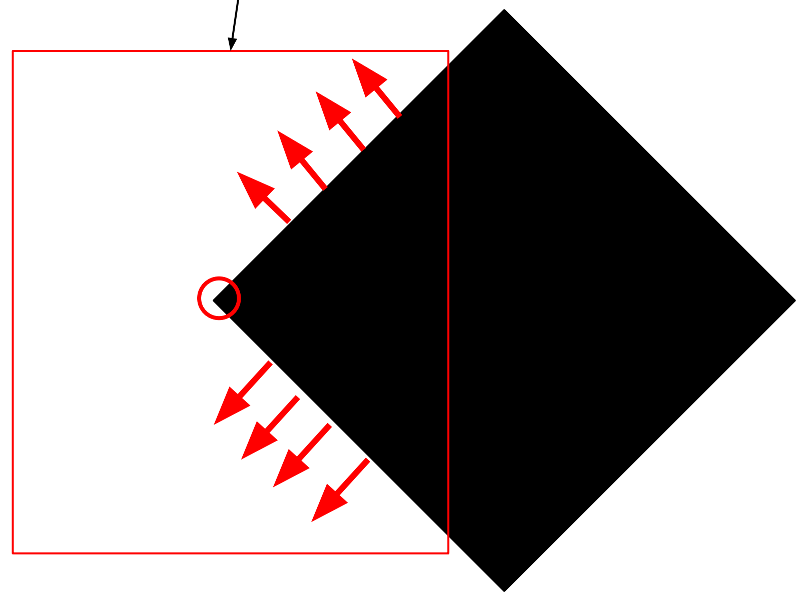


Università
Ca' Foscari
Venezia

Corners

How to find a corner? General idea:

- Exactly at a corner, gradient is ill defined.
- However, in the region around a corner, gradient has two or more different well-defined vectors.



Università Ca' Foscari Venezia

Corners and gradient

Similarly to edges, a corner point exhibit strong rapid changes in the image intensities.

For a small region around a point x_0, we can consider the Taylor expansion of the image function I(x,y) and express the change of intensity as function of the image gradient and a displacement vector h:

$$I(\mathbf{x_0} + \mathbf{h}) \approx I(x_0) + \nabla I(\mathbf{x}_0)^T h$$

$$I(\mathbf{x_0} + \mathbf{h}) - I(x_0) \approx \nabla I(\mathbf{x}_0)^T h$$

Corners and gradient

$$I(\mathbf{x_0}+\mathbf{h}) - I(x_0) \approx \nabla I(\mathbf{x}_0)^T h$$

We are not interested to the sign of this variation (gradient can have any orientation) but only to its magnitude. So we can compute the square of it:

$$\left(I(\mathbf{x_0}+\mathbf{h}) - I(x_0)\right)^2 \approx \mathbf{h}^T \nabla I(\mathbf{x}_0) \nabla I(\mathbf{x}_0)^T \mathbf{h}$$

Corners and gradient

To be more resilient to noise, we can compute this intensity difference by averaging over a region Ω_{x_0} centered at x_0:

$$\left(I(x_0+h)-I(x_0)\right)^2 \approx \sum_{x\in\Omega_{x_0}} w(x-x_0)\left(I(x+h)-I(x)\right)^2$$

this is usually a Gaussian windowing function

Corners and gradient

Considering the Taylor expansion we have seen before, we have:

$$\left(I(x_0+h)-I(x_0)\right)^2 \approx \underbrace{\sum_{x\in\Omega_{x_0}} w(x-x_0)h^T\nabla I(x)\nabla I(x)^T h}_{E(x_0)}$$

Since h does not depend to x, we can move it out from the summation

Corners and gradient

$$E(x_0) = h^T \left(\sum_{x \in \Omega_{x_0}} w(x - x_0) \nabla I(x) \nabla I(x)^T \right) h$$

$$C = \sum_{x \in \Omega_{x_0}} w(x - x_0) \begin{bmatrix} I_u^2(x) & I_u(x) I_v(x) \\ I_v(x) I_u(x) & I_v^2(x) \end{bmatrix}$$

$E(x_0)$ can then be written as: $E(x_0) = h^T C h$

And the summation can be moved inside the matrix:

$$C = \begin{bmatrix} \sum_{x \in \Omega_{x_0}} w(x - x_0) I_u^2(x) & \sum_{x \in \Omega_{x_0}} w(x - x_0) I_u(x) I_v(x) \\ \sum_{x \in \Omega_{x_0}} w(x - x_0) I_v(x) I_u(x) & \sum_{x \in \Omega_{x_0}} w(x - x_0) I_v^2(x) \end{bmatrix}$$

Second moment matrix

C form the second-moment matrix (we discard the weights for clarity)

$$C = \begin{bmatrix} \sum I_u^2 & \sum I_u I_v \\ \sum I_v I_u & \sum I_v^2 \end{bmatrix}$$

1. Depends on the first-order derivatives
2. Symmetric
3. Each element is obtained as a sum over a small region around a point x_0

Simple case

First, consider the following ideal case:

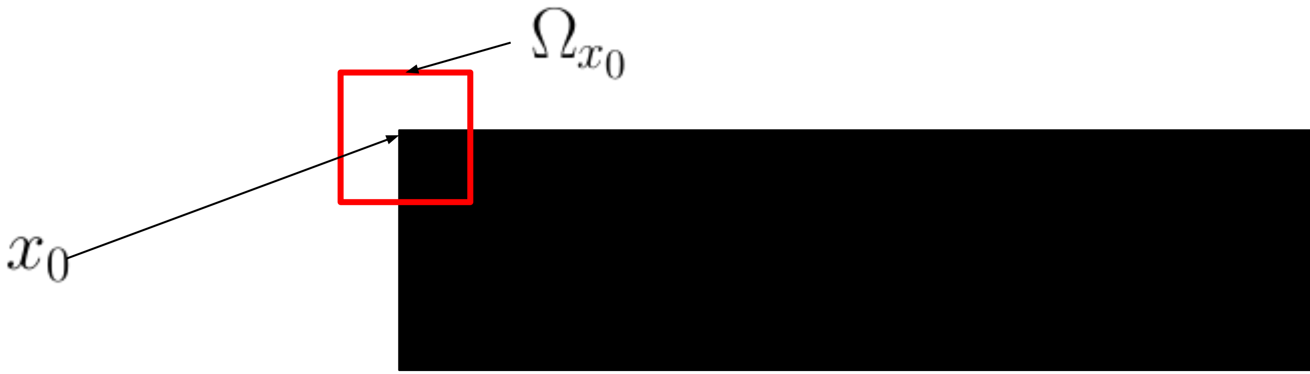


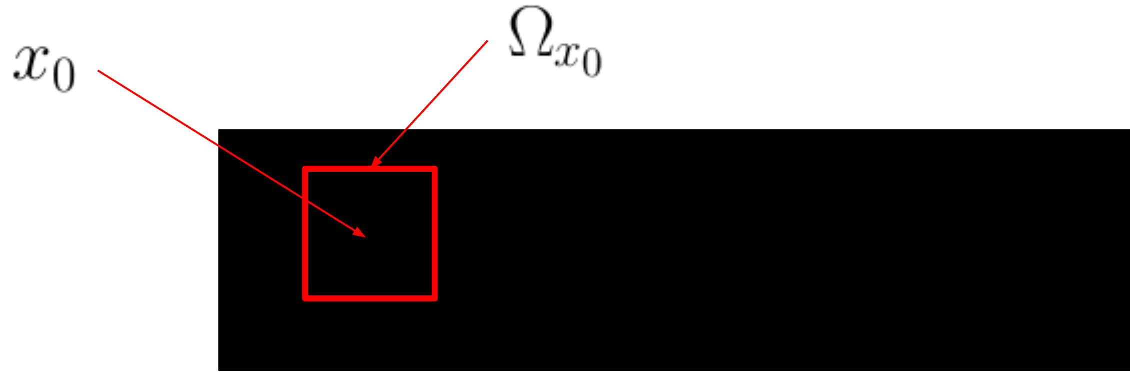
Image intensity changes either in x or y direction, but not both

$$C = \begin{bmatrix} \sum I_u^2 & \sum I_u I_v \\ \sum I_v I_u & \sum I_v^2 \end{bmatrix} = \begin{bmatrix} \lambda 1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Simple case

When x_0 is at a flat region, we expect

$$\lambda_1 = \sum I_u^2 = 0, \lambda_2 = \sum I_v^2 = 0$$

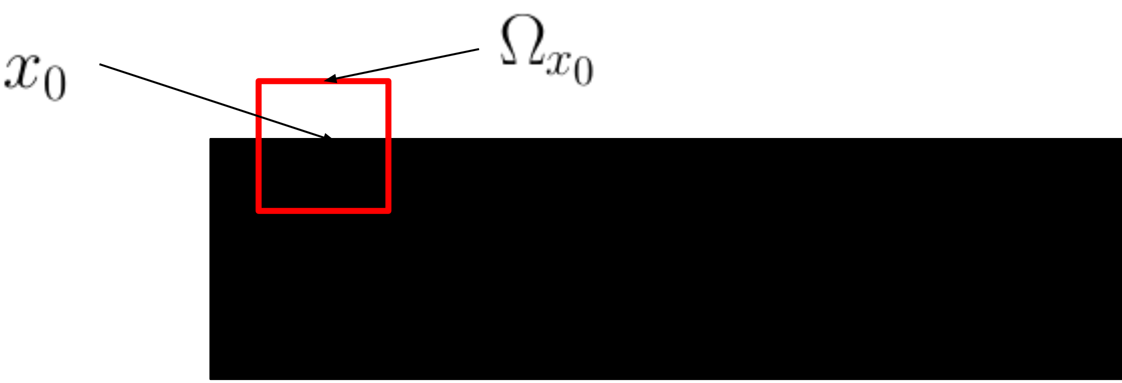


$$C = \begin{bmatrix} \sum I_u^2 & \sum I_u I_v \\ \sum I_v I_u & \sum I_v^2 \end{bmatrix} = \begin{bmatrix} \lambda 1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Simple case

When x_0 is at an horizontal edge, we expect

$$\lambda_1 = \sum I_u^2 = 0, \lambda_2 = \sum I_v^2 \gg 0$$



$$C = \begin{bmatrix} \sum I_u^2 & \sum I_u I_v \\ \sum I_v I_u & \sum I_v^2 \end{bmatrix} = \begin{bmatrix} \lambda 1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Università
Ca' Foscari
Venezia

Simple case

When x_0 is at a vertical edge, we expect

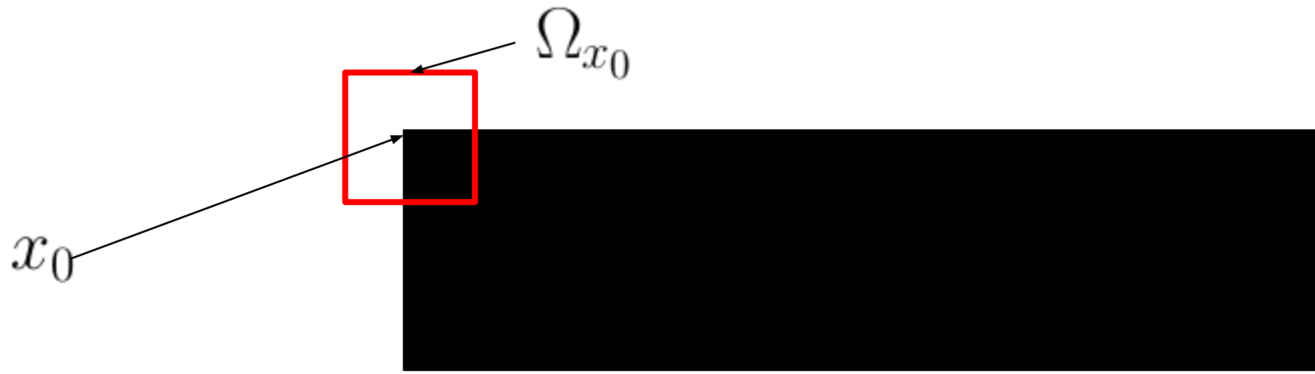
$$\lambda_1 = \sum I_u^2 \gg 0, \lambda_2 = \sum I_v^2 = 0$$

$$C = \begin{bmatrix} \sum I_u^2 & \sum I_u I_v \\ \sum I_v I_u & \sum I_v^2 \end{bmatrix} = \begin{bmatrix} \lambda 1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Università
Ca' Foscari
Venezia

Simple case

When x_0 is at a **corner**, we expect both $\sum I_u^2$ and $\sum I_v^2$ be large (ie. λ_1, λ_2 far from zero)



$$C = \begin{bmatrix} \sum I_u^2 & \sum I_u I_v \\ \sum I_v I_u & \sum I_v^2 \end{bmatrix} = \begin{bmatrix} \lambda 1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

General case

Università
Ca' Foscari
Venezia

So we can detect a corner if both λ_1, λ_2 are far from zero.

What about the general case in which $\sum I_u I_v$ are not zero?

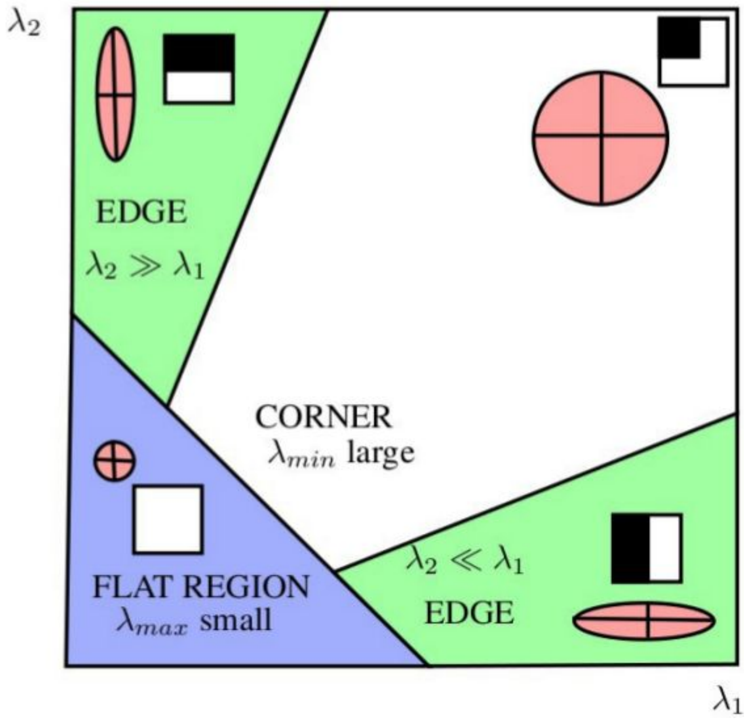
Since C is symmetric, it can be decomposed via SVD:

$$C = R \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} R^T$$

Where R is a rotation matrix and λ_1, λ_2 are the singular values of C (ie. the square-root of the eigenvalues of $C^T C$)

General case

Since the rotations do not change the magnitude of h, examining the singular values of C can tell us if x_0 is in a flat region, an edge or a corner



Harris corner detector

Analyzing the singular values of C requires the computation of SVD at each image pixel

> This is computationally expensive in practice

Harris proposed to use the following function as a corner response:

$$R(x_0) = det(C_{x_0}) - k \; \mathrm{trace}^2(C_{x_0})$$

Where k is a constant that has to be tuned for the specific application

Università Ca' Foscari Venezia

Harris corner detector

$$R(x_0) = det(C_{x_0}) - k\ \mathrm{trace}^2(C_{x_0})$$

It can be shown that:

$$\mathrm{trace}(C_{x_0}) = \lambda_1 + \lambda_2$$

$$\det(C_{x_0}) = \lambda_1 \lambda_2$$

Therefore, $R(x_0) >> 0$ if we are on a corner, and $R(x_0) << 0$ if we are on an edge

Harris corner detector

Algorithm:

- Compute the image gradient $I_u(x, y), I_v(x, y)$
- Compute the matrix C for each pixel
  - 3 convolutions needed: $K \star I_u^2, \quad K \star I_u I_v, \quad K \star I_v^2$
  - Convolution kernel K is usually gaussian and determine the scale of the corner
- Compute the Harris response for each pixel
- Threshold the result and (optionally) perform non-maxima suppression

Università
Ca' Foscari
Venezia

Harris corner detector

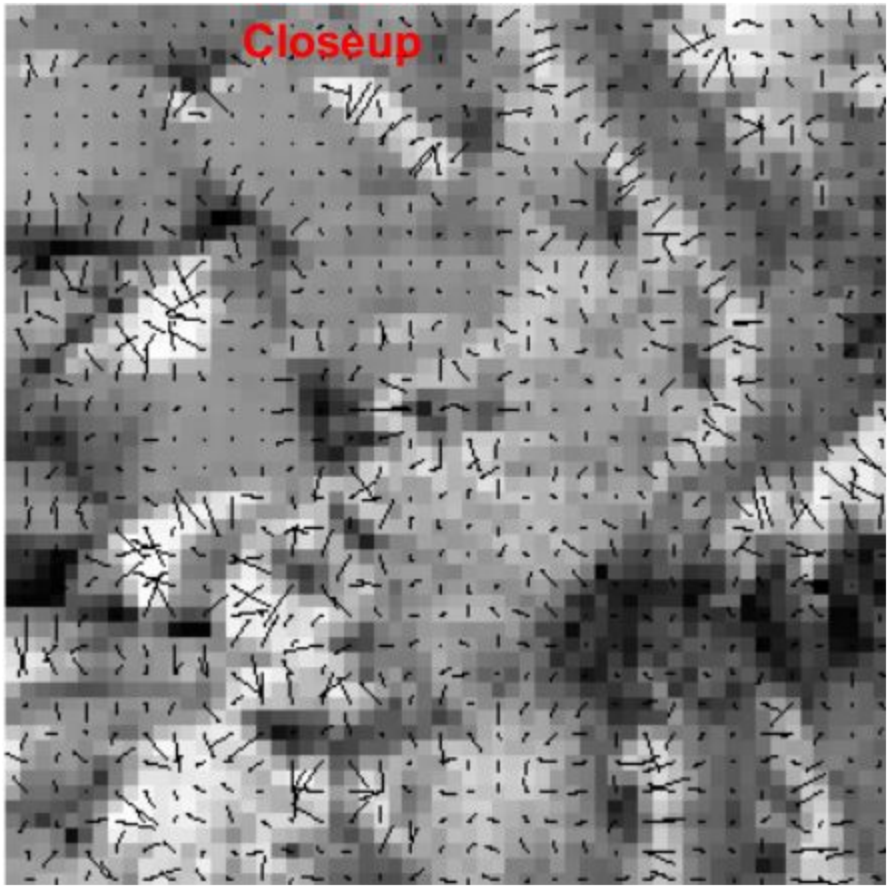
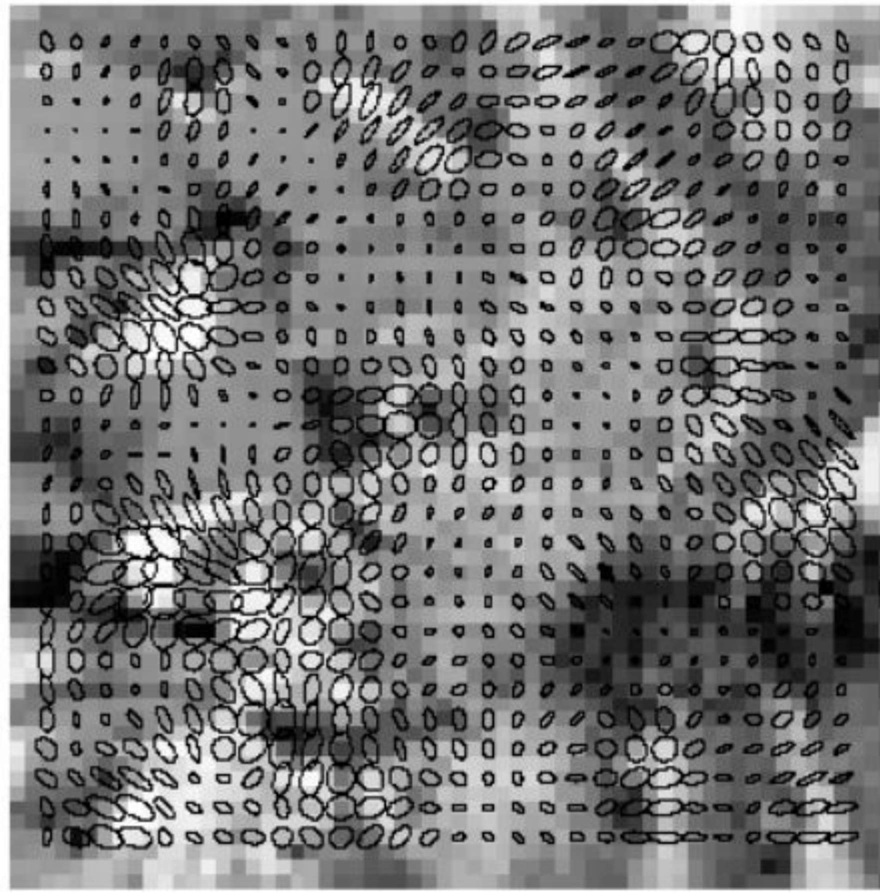


Image Gradient

Harris corner detector

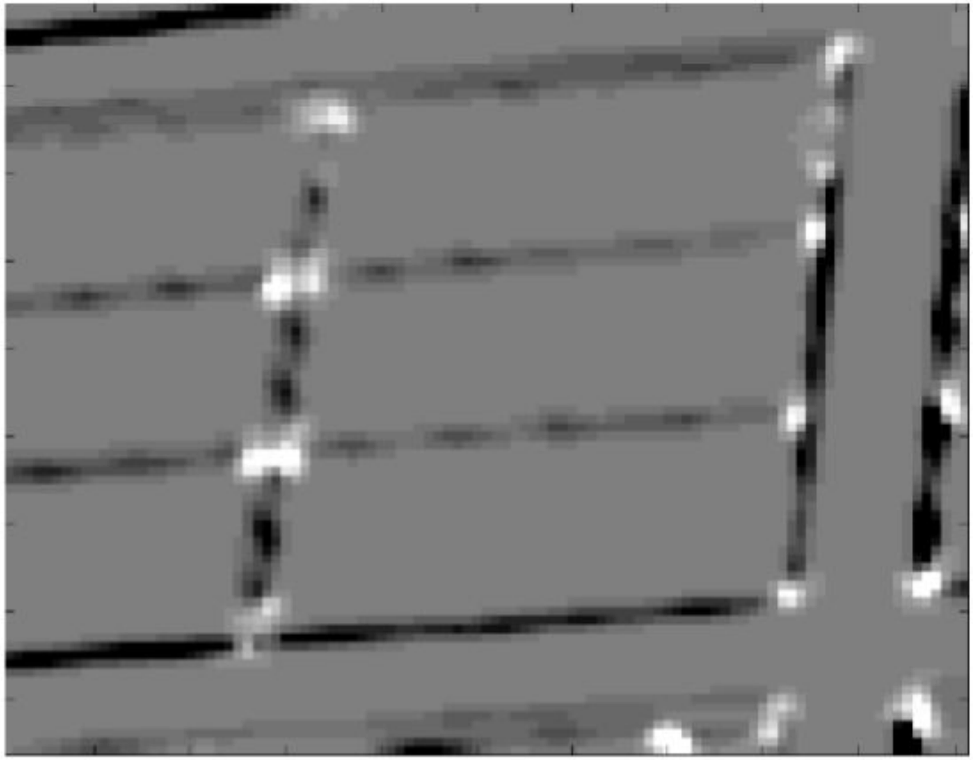


Eigenvalues plotted as ellipse axes

Università Ca' Foscari Venezia

Harris corner detector

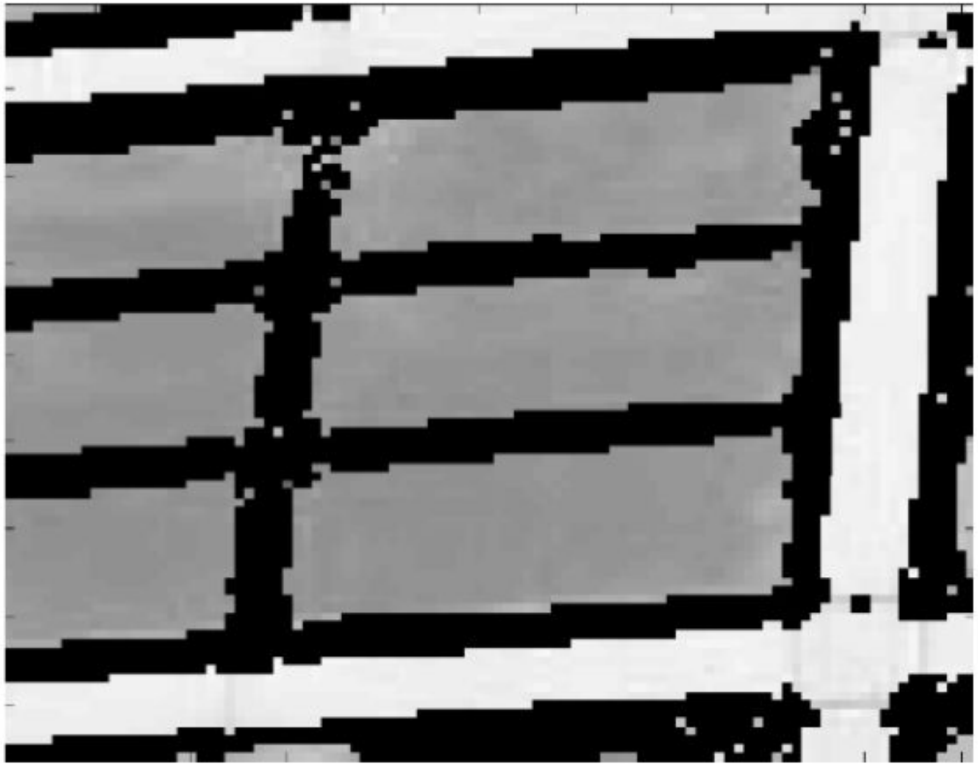
Input image



Harris response (R)

Harris corner detector

Input image

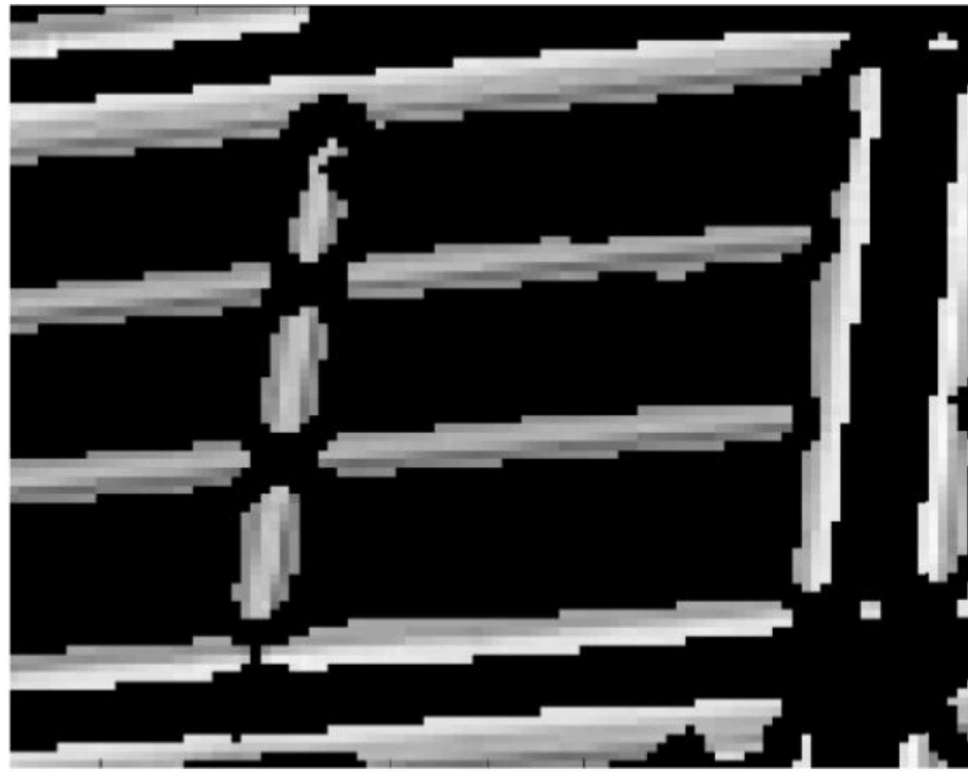


|R|<1E4
(flat regions)

Università
Ca' Foscari
Venezia

Harris corner detector

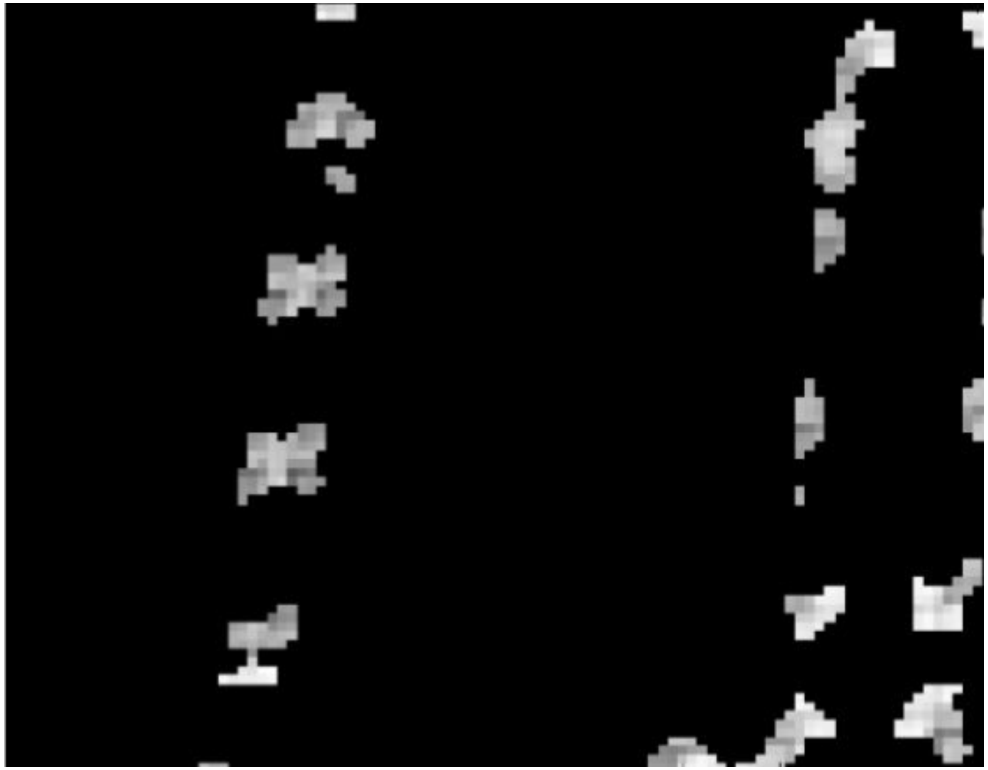
Input image



R<-1E4
(edges)

Harris corner detector

Input image



R>1E4
(corners)

More advanced features

Harris corner detector works well in practice but is not invariant to scale

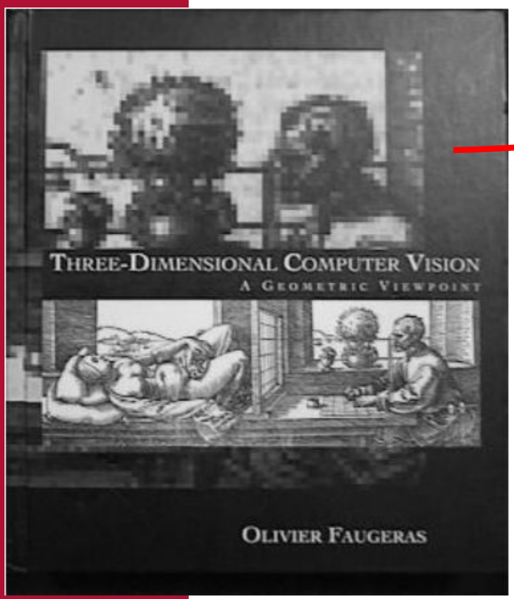
- The convolution window size affects the scale of the corner detected

To solve complex high-level computer vision problems we need more invariances and a way to distinguish and identify features

Università
Ca' Foscari
Venezia

A typical problem

Find this:



In this picture:

SIFT

Università
Ca' Foscari
Venezia

David G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

… changed the way we approach many computer vision problems!

Invariances:

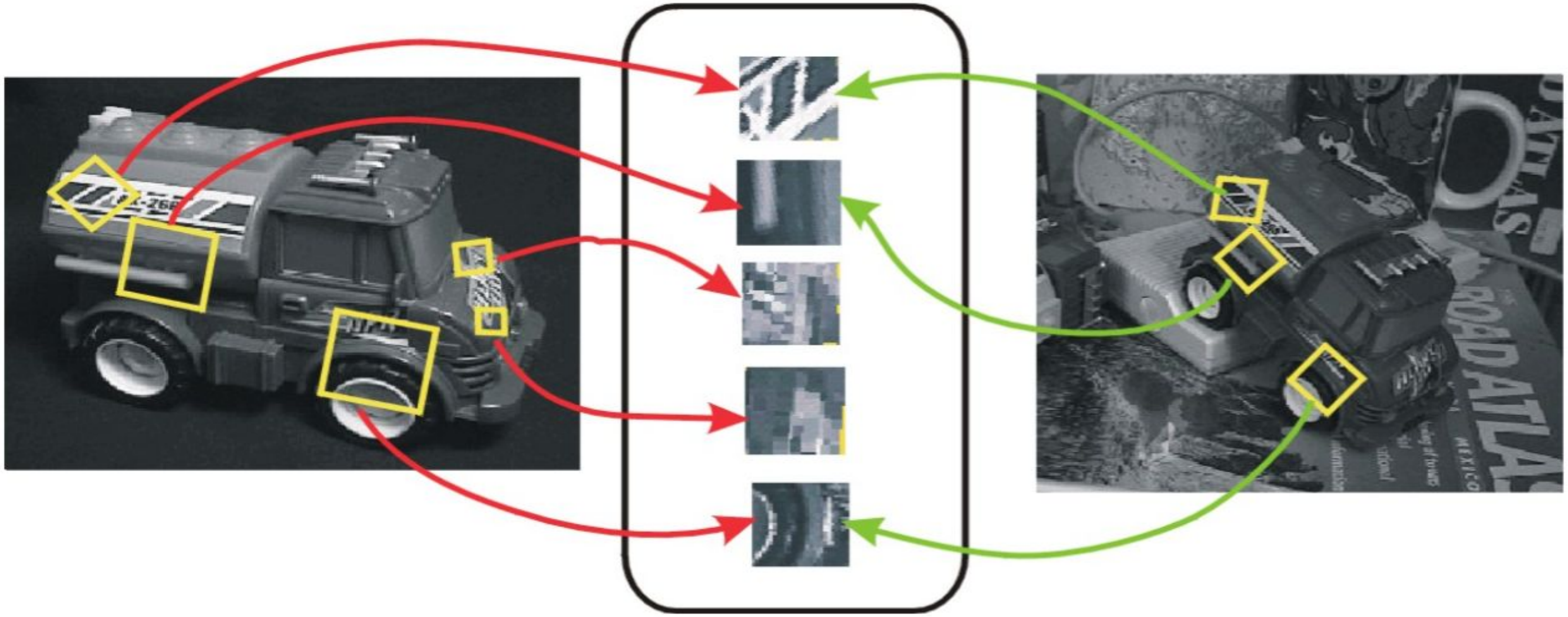
- Scaling
- Rotation
- Illumination

While still providing very good localization

Università
Ca' Foscari
Venezia

SIFT: General idea

Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Università
Ca' Foscari
Venezia

SIFT: Advantages

Locality: features are local, so robust to occlusion and clutter (no prior segmentation)

Distinctiveness: individual features can be matched to a large database of objects

Quantity: many features can be generated for even small objects

Efficiency: close to real-time performance

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

SIFT Algorithm

Keypoint Localization:

1. **Enforce invariance to scale:** Compute difference of Gaussian for may different scales; non-maximum suppression, find local maxima: keypoint candidates
2. **Localize corners:** For each maximum, fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.
3. **Eliminate edges:** Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.

SIFT Algorithm

Signature computation:

4. **Enforce invariance to orientation:** Compute orientation by finding the strongest gradient direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points upward.

5. **Compute feature signature (descriptor):** Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).

SIFT Algorithm

6. **Enforce invariance to illumination change and camera saturation:**
Normalize the descriptor to unit length to increase invariance to illumination. Then, threshold all gradients, to become invariant to camera saturation.

SIFT - Step 1

Enforce invariance to scale: Compute difference of Gaussian for may different scales; non-maximum suppression, find local maxima: keypoint candidates

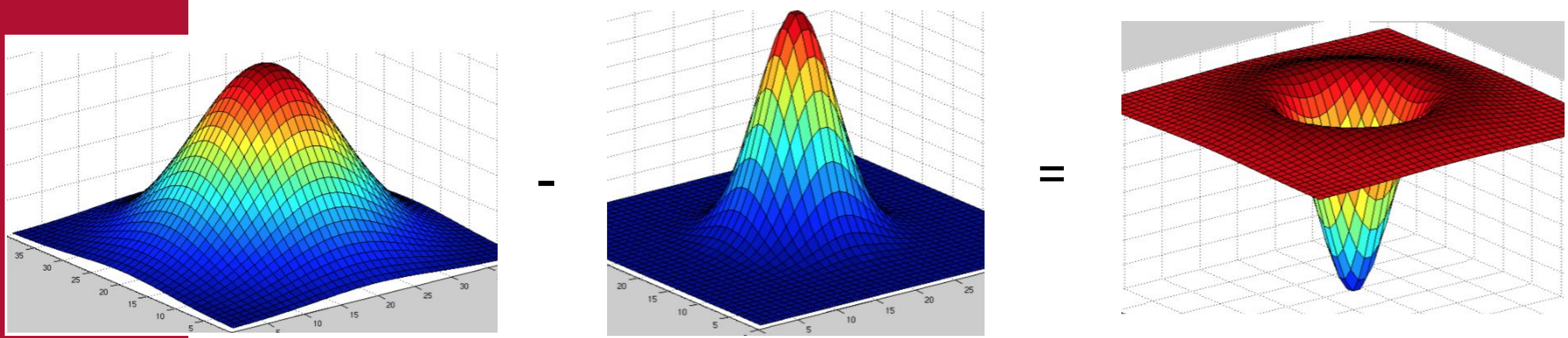
Main idea: Find corners as in Harris, but achieve scale invariance

Method:

- Convolve with Difference of Gaussians (DoG) to identify interesting image pixels
- DoG is performed at multiple resolutions and the local maximum (in space and scale is selected)

Difference of Gaussians

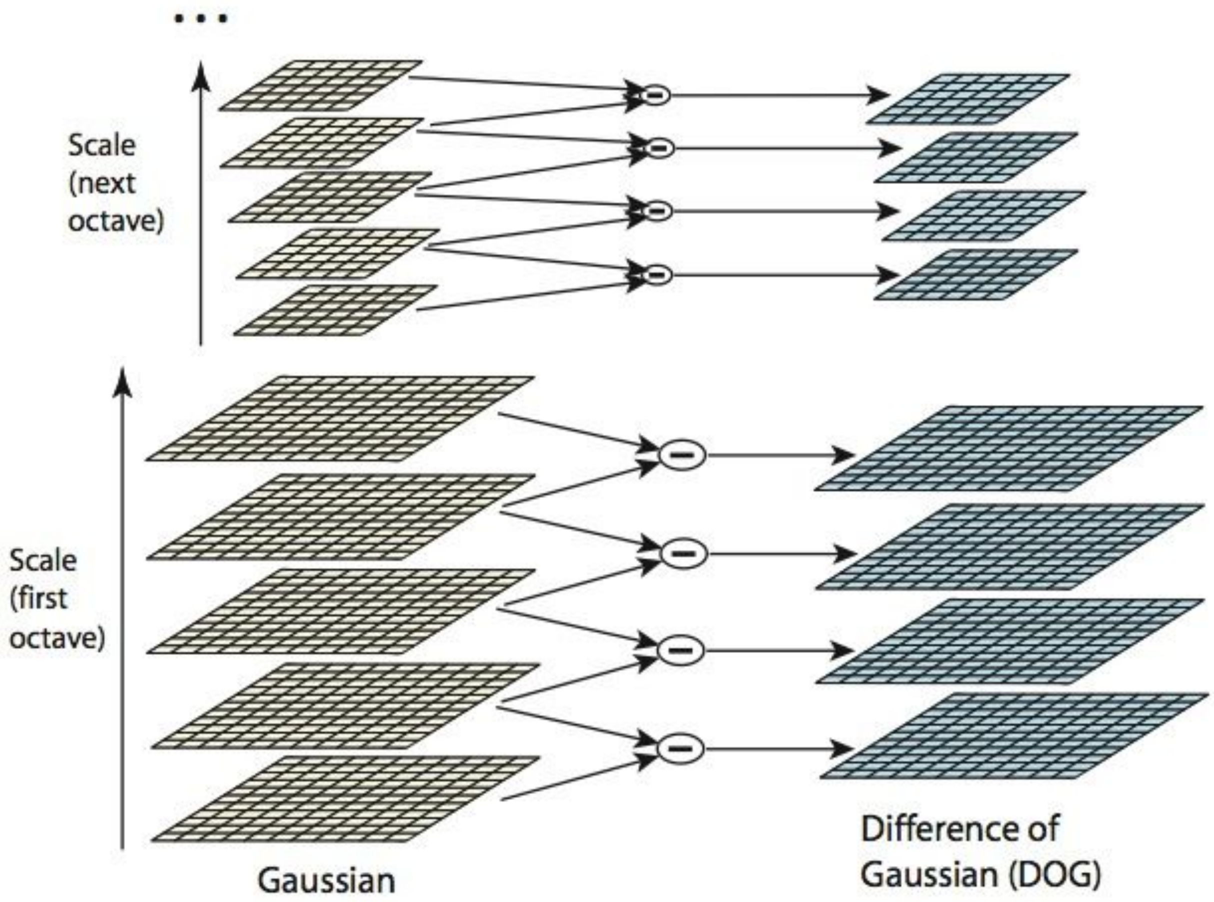
Essentially an High-pass filter which approximates well the LoG.



Why use that? We can efficiently compute the DoG at different scales using image pyramid

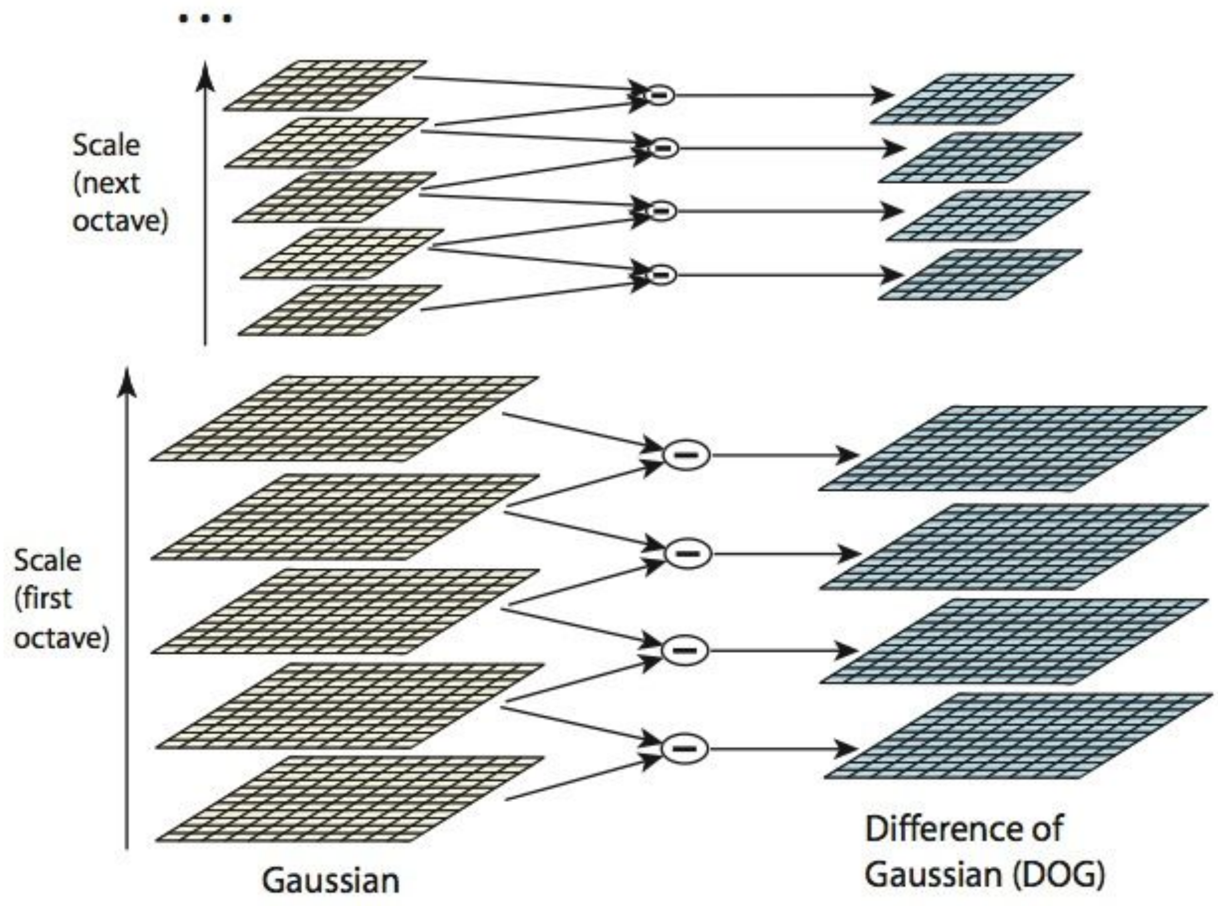
DoG & Image Pyramid

Image is scaled several times (size is halved), each scale is called an octave



DoG & Image Pyramid

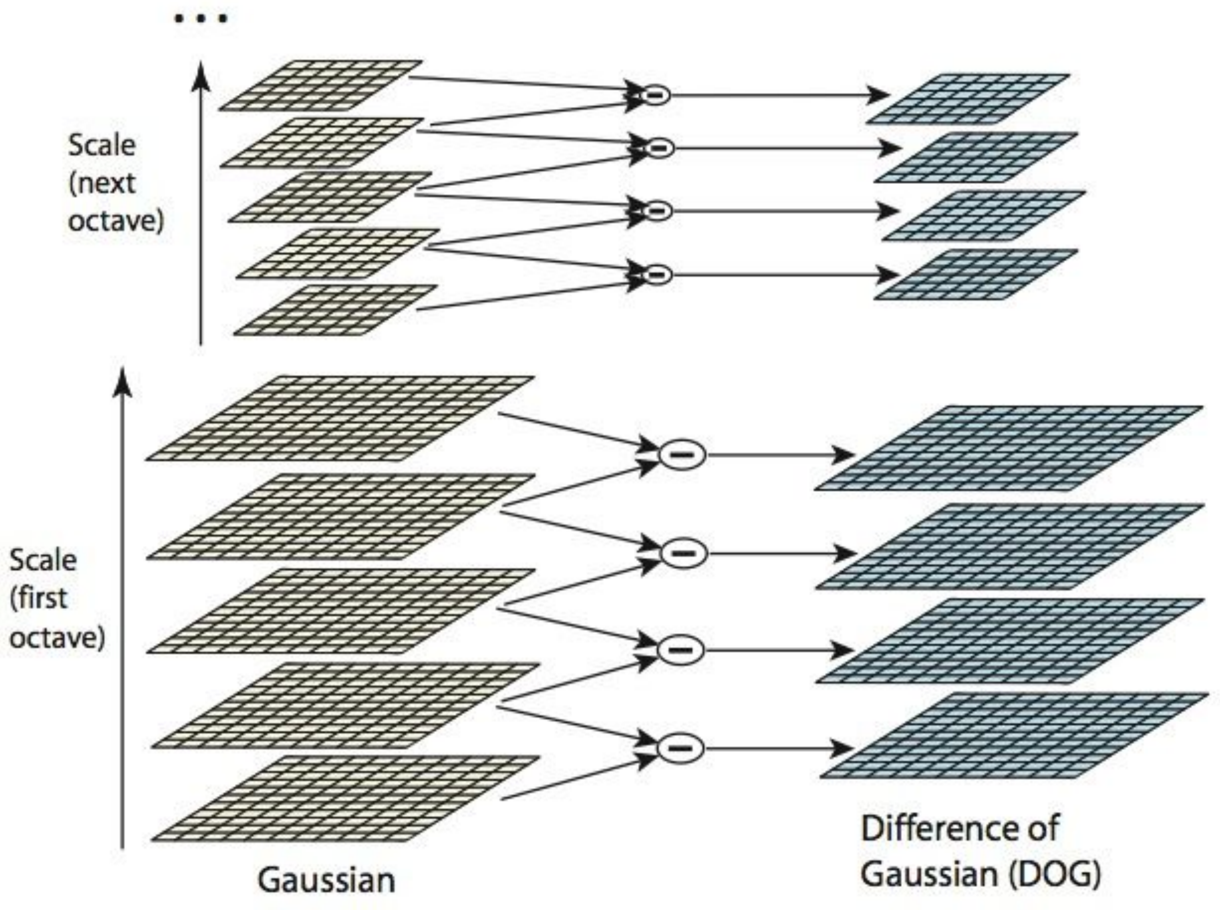
For each octave of scale space, the initial image is repeatedly convolved with Gaussians



Università Ca' Foscari Venezia

DoG & Image Pyramid

Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images

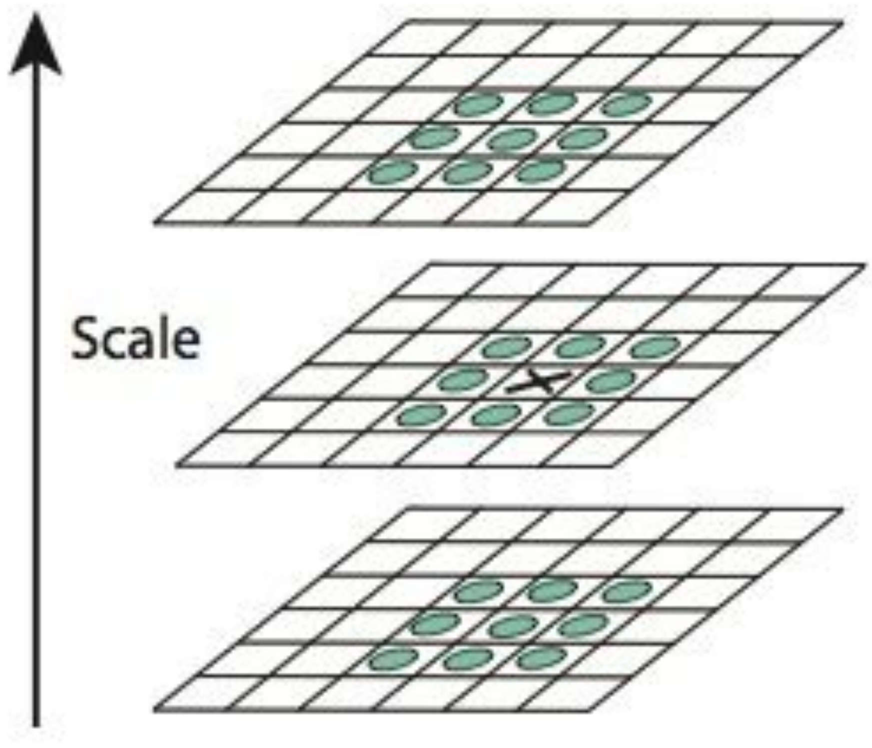


Università Ca' Foscari Venezia

Keypoint localization

Once the DoG pyramids are built, the local-maxima are extracted considering both current-scale and adjacent scales

Local-maximum is checked against 9+8+9=26 neighbours



Università
Ca' Foscari
Venezia

SIFT - Step 2 & 3

2. Localize corners: For each maximum fit a quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.

3. Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.

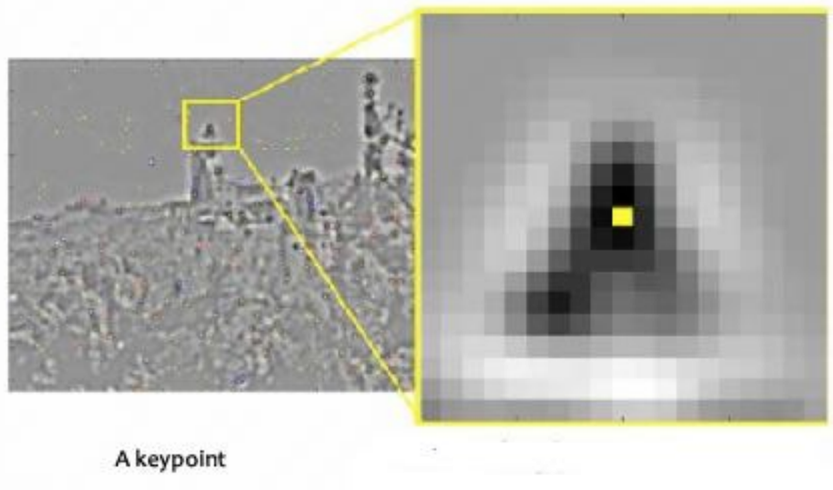
Threshold on value at DoG peak and on ratio of principal curvatures (similar to Harris approach)

SIFT - Step 4

4. Enforce invariance to orientation

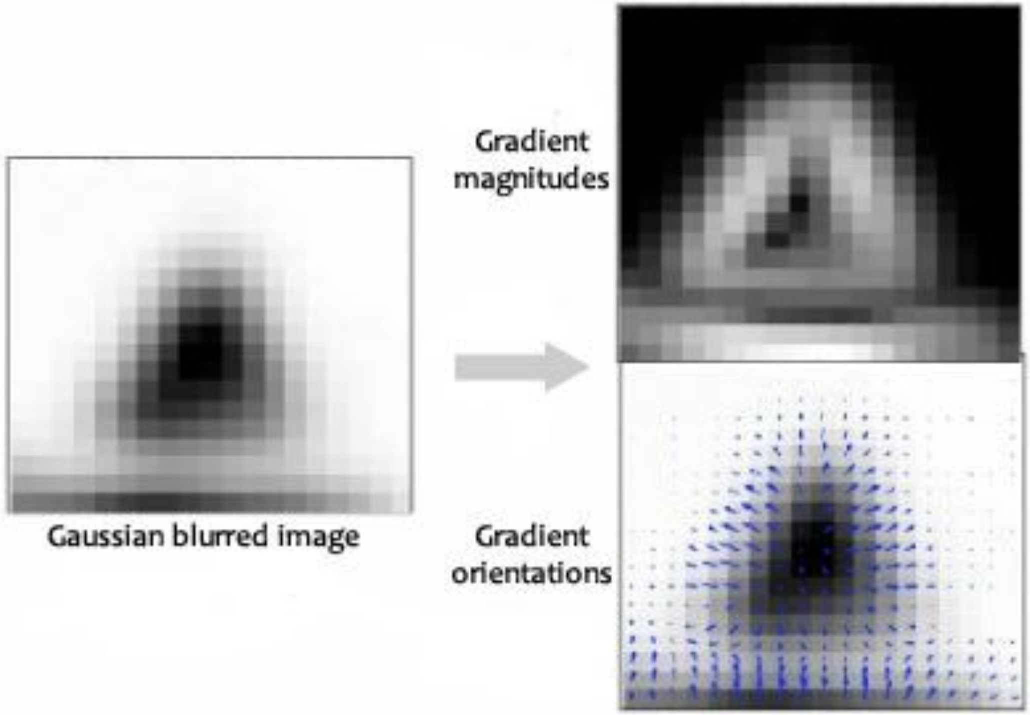
By assigning a consistent orientation to each key point based on local image properties we can achieve invariance to image rotation.

Suppose that we want to assign an orientation to this detected keypoint:



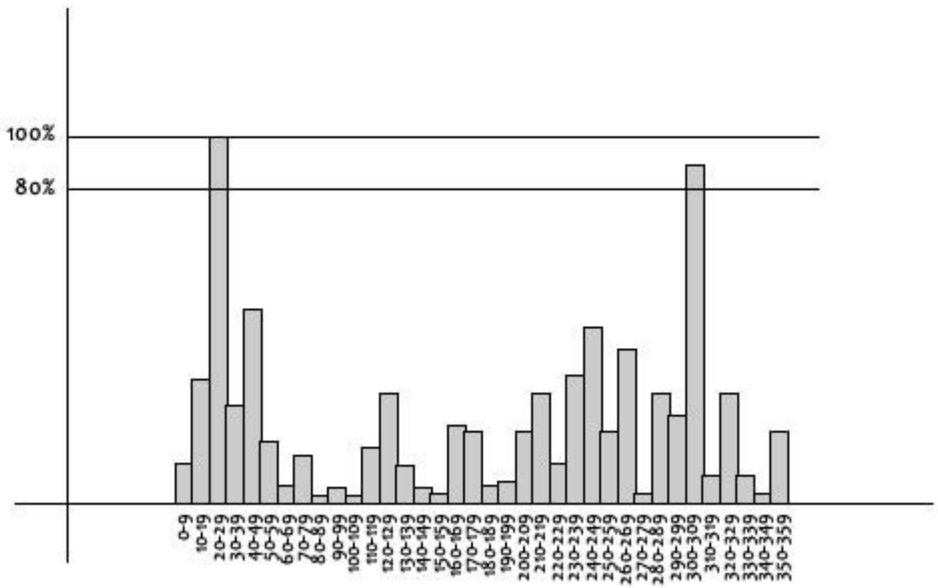
SIFT - Step 4

Gradient magnitude and orientation is calculated for each pixel in the keypoint region (region size depends on the detected scale)



SIFT - Step 4

And an orientation histogram is formed with these orientations and magnitudes



The maximum value of the histogram gives the orientation of the detected keypoint

Università
Ca' Foscari
Venezia

SIFT - Step 4

After the step 4, each detected keypoint is characterized by:

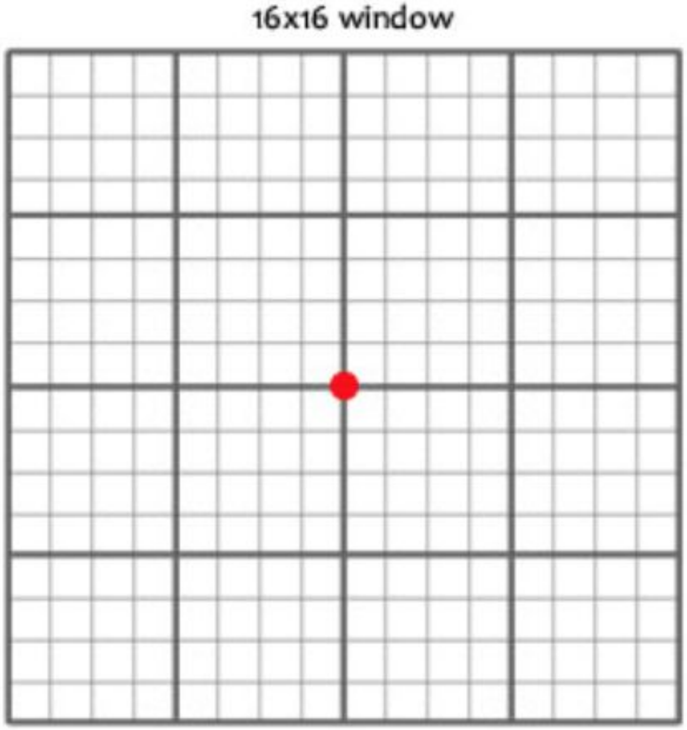
- A coordinate in the image space (x,y)
- A scale
- An orientation

Steps 5 and 6 aims to create a signature (or descriptor) that can be used to uniquely identify the features with respect to the others

Università
Ca' Foscari
Venezia

SIFT - Step 5

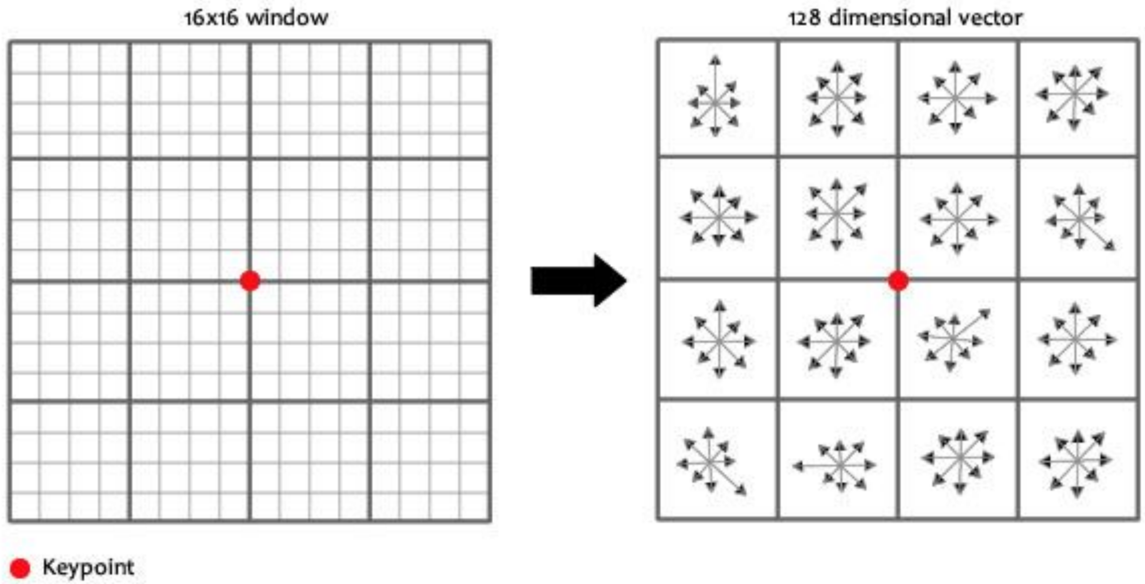
To create the descriptor we look at the gradient vectors in a 16x16 window around each keypoint (window is rotated with respect to the keypoint orientation)



Università
Ca' Foscari
Venezia

SIFT - Step 5

- The 16x16 window is divided into 16 4x4 windows.
- For each 4x4 window, an 8-bins (45° steps) histogram of gradient orientation is formed
- Histograms are concatenated all together to produce a 16x8=128-values feature vector



SIFT - Step 6

To reduce the effect of illumination change, feature vectors (descriptor) are normalized to have unitary length.