Keypoint detection and description

Javier González Jiménez

Reference Books:

- *Computer Vision: Algorithms and Applications*. Richard Szeliski. Springer. 2010.

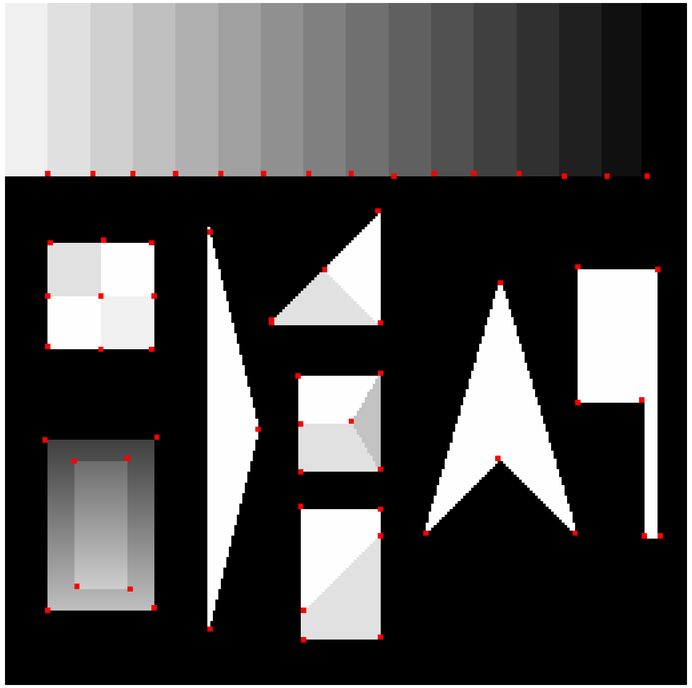
http://szeliski.org/Book

Content

1. Introduction

What are *keypoints* (also known as *interest* or *feature points*)?

Distinctive pixels in the image that can likely be projections of 3D entities.



Synthetic image

Real image

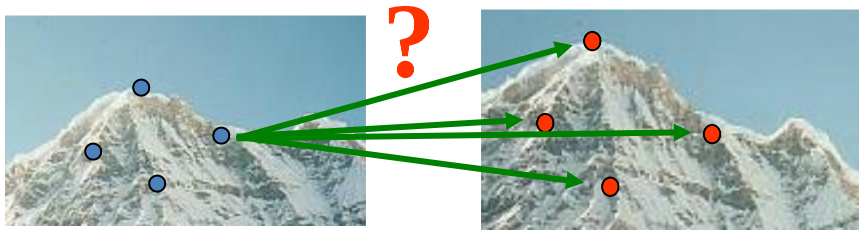
1. Introduction

Two problems involved: Detection and description

1. Detection

The operator must provide a reliable and repeatable response

2. Description: To be matched to its correspondence in other images



Invariant and discriminative description needed

Harris is just a detector (not a descriptor). The keypoint is described with a surrounding image patch and matched through correlation
SIFT is a detector and also provides its own descriptor

Keypoint detection properties:

- **Accurate localization** (subpixel accuracy, if possible)
- **Reliable detection**. No error in the detection

There are **four types of predictions** (outcomes):

- **Correct** predictions: True Positives (TP) and True Negatives (TN)
- **Wrong** predictions
  - False Positives (FP): a point is detected when actually it isnot a keypoint
  - False Negative (FN): a keypoint is not detected (goes unnoticed)

confusion matrix

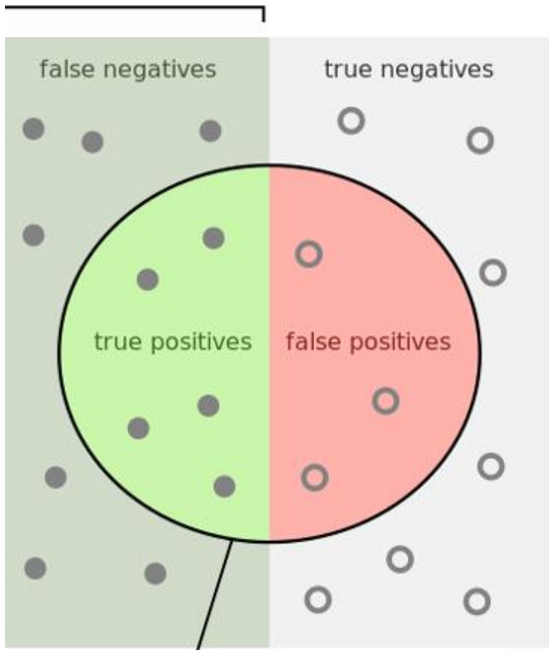
		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**

Keypoint **detection performance**:

- Detect all the keypoints in the image (few FALSE NEGATIVES: FN)
- NOT detect irrelevant points (few FALSE POSITIVES: FP)

Two normalized metrics (between 0 and 1):

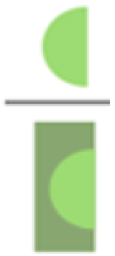
Keypoints in the image
(Ground truth)



Detected keypoints
(positive of the detector)

Image courtesy: Wikipedia

- **RECALL**: measure FN, i.e. the SENSITIVITY of the detector

Recall = 

$$= \frac{TP}{\boxed{TP + FN}}$$

actually positive

Among all matches that are actually positive (TP+FN), what percentage is a true positive (TP)

Recall=1 → no FN, very sensitive detector

- **PRECISION**: measure FP, i.e. the PRECISION of the detector

Precision = 

$$= \frac{TP}{\boxed{TP + FP}}$$

detected positive

Among all matches that were detected positive (TP+FP), what percentage is a true positive (TP)

Precision=1 → no false alert

Sometimes, instead of the precision, the SPECIFICITY is used:

$$Specificity = \frac{TN}{TN + FP}$$

Keypoint descriptor properties:

Descriptor should have *invariance* to:

✓ Illumination (changes in brightness and contrast)

✓ View point (scale, rotación, projective distortion)

Of course, both, detector and descriptor must be computationally efficient

1. Introduction

Applications

- Panoramic images (image sticking)
- 3D Reconstruction
- Object tracking
- Object recognition
- Image retrieval and indexing in database
- Robot navigation: mapping, localization, obstacle detection
- …. and many others

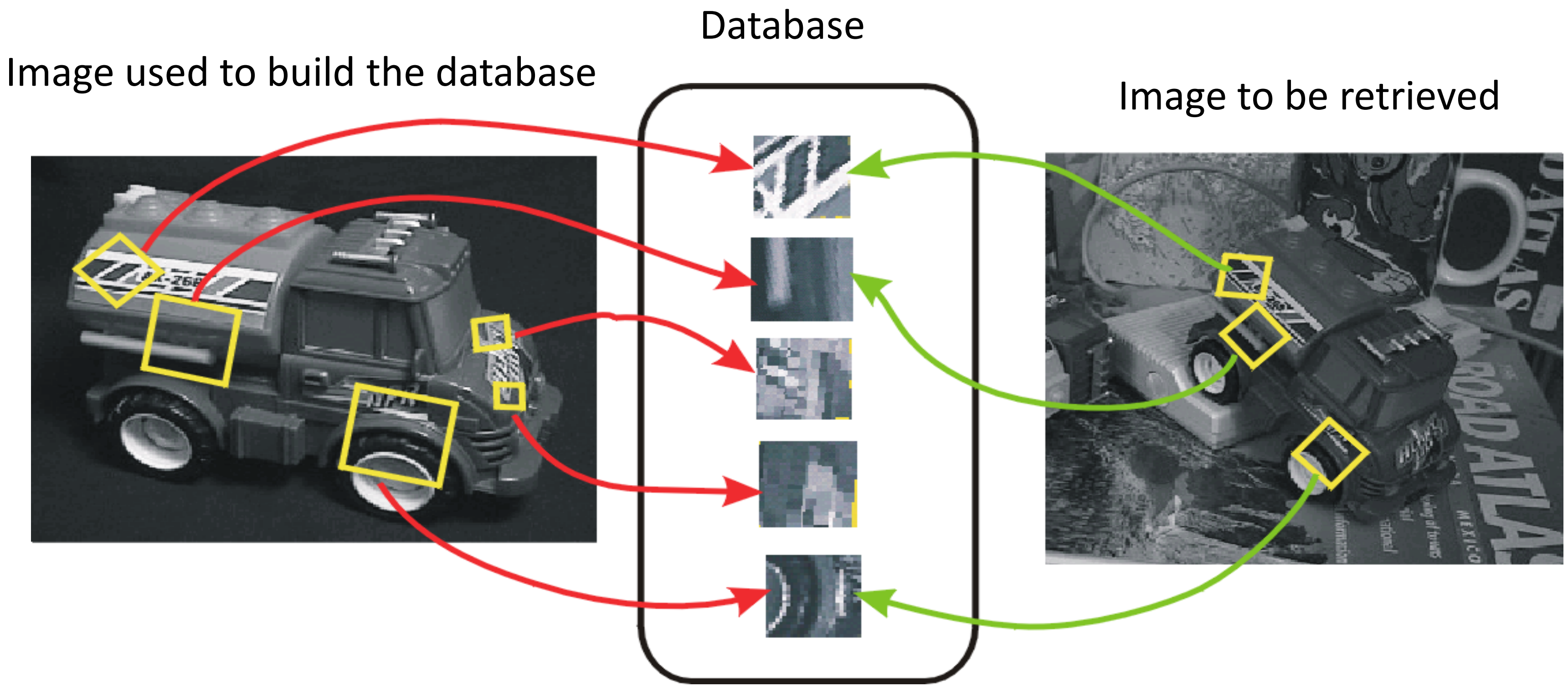
1. Introduction

Applications: Panoramic images (image stitching)

M. Brown and D. G. Lowe. Recognising Panoramas. ICCV 2003

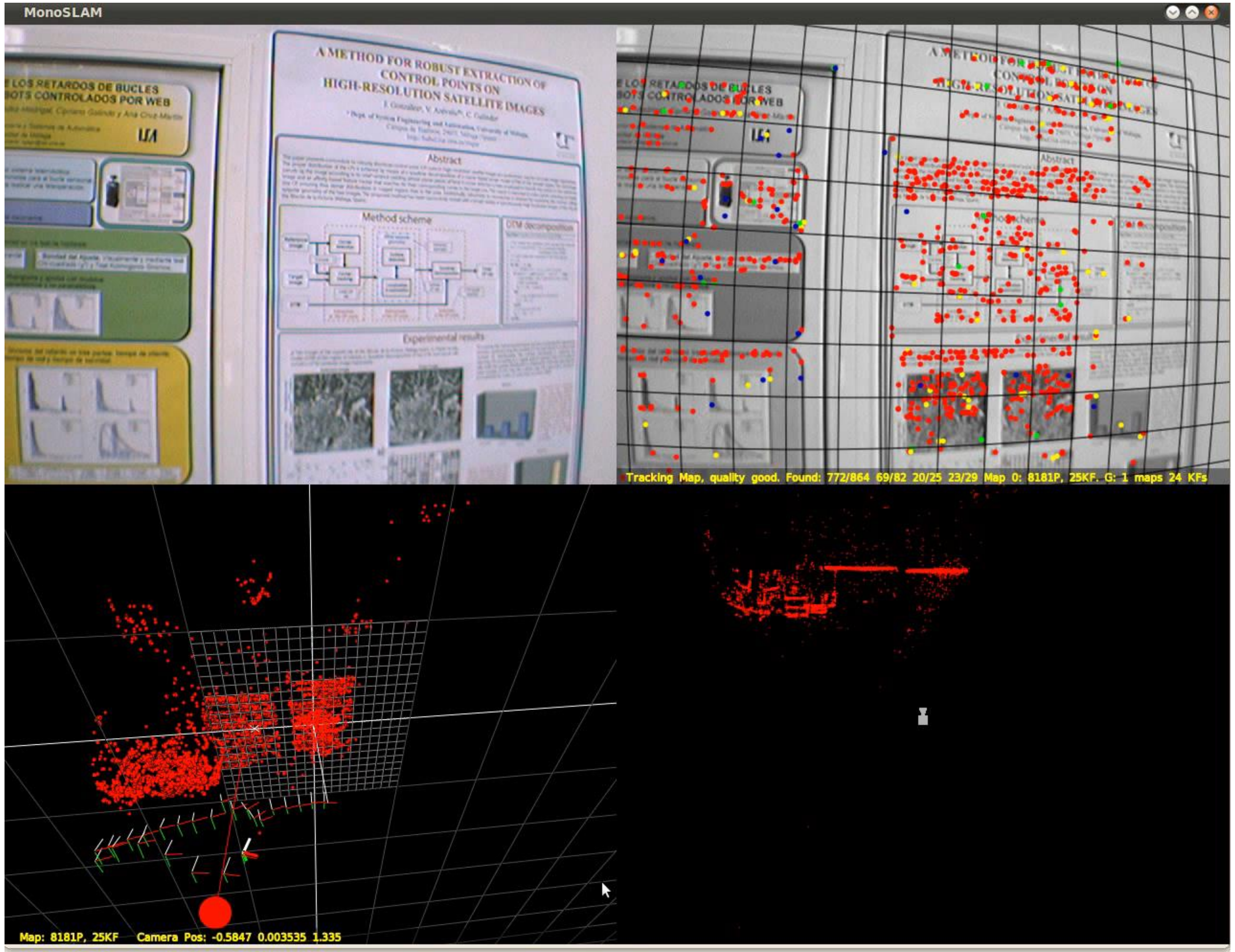
1. Introduction

Applications: Image retrieval and indexing in database



1. Introduction

Applications: Localization and 3D Reconstruction



2. Harris detector

- Detect corners
- Simple and efficient implementation
- Robustness to noise (by apply smoothing)
- Invariance to
  - Rotation: uses eigenvectors
  - Brightness (partially to contrast): uses derivatives ☺
- **Not invariance to scale** ☹

C.Harris, M.Stephens. "A Combined Corner and Edge Detector".
Proceedings of The Fourth Alvey Vision Conference, *1988*

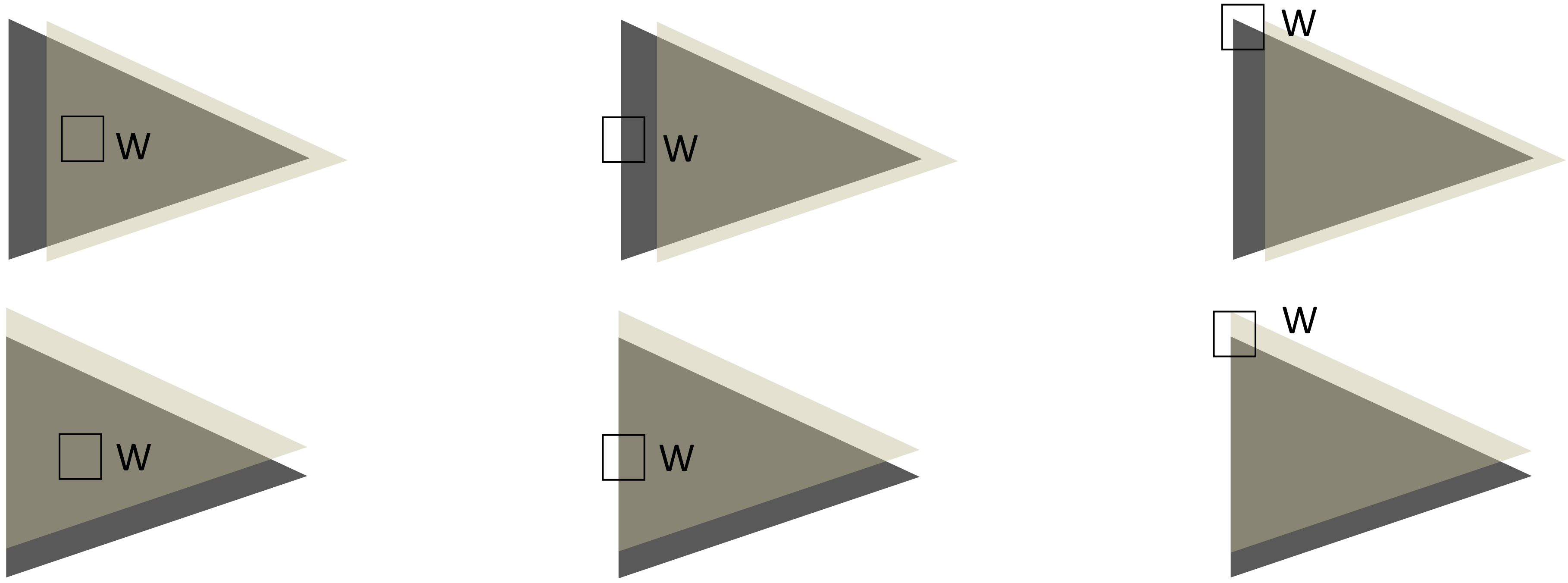
2. Harris detector

- A keypoint is a corner (“esquina”)
- A corner is a point with high variation of intensity in 2 spatial directions

Basic idea: look in a small window W around a pixel if the displacements of the image in two directions provoke changes

Displacement in Δx →

Displacement in Δy ↑



“flat” region:
No change in W

“edge”:
change in W only if motion in the Δx direction

“corner”:
change in W if motion in both the Δx and Δy directions

Detecting the local change in intensity due to a shift $(\Delta x, \Delta y)$:

Sum-of-square weighted difference at a pixel $[x_0 y_0]$ when a window image I is shifted $(\Delta x, \Delta y)$:

$$E_{x_o y_0}(\Delta x, \Delta y) = \sum_{x,y} w(x,y)[I(x+\Delta x, y+\Delta y) - I(x,y)]^2$$

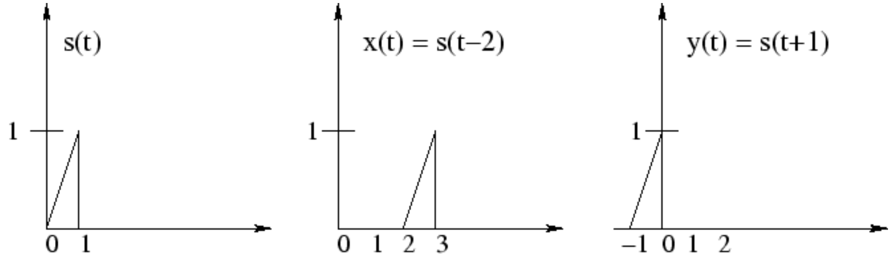
Weighting window centered at $[x_0 y_0]$

Image shifted $(\Delta x,\Delta y)$

Image

RECALL:

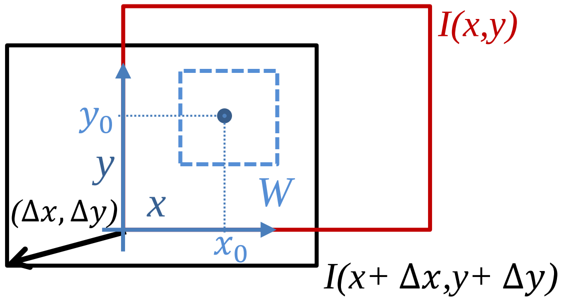
1D signal shifting



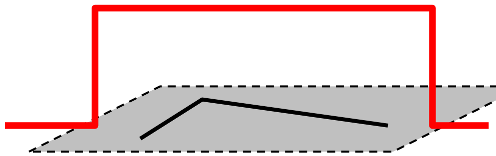
Right shifting

Left shifting

2D Left-down image shifting

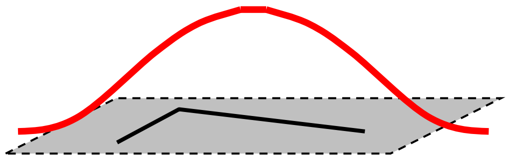


Weighting function $w(x,y)$ may be:



1 inside, 0 outside

Boolean



Gaussian weighting

Gaussian

Let's make more practical the computation of $E_{x_o y_0}(\Delta x, \Delta y)$

$$E_{x_o y_0}(\Delta x, \Delta y) = \sum_{x,y} w(x,y)[I(x+\Delta x, y+\Delta y) - I(x,y)]^2 = \sum_{(x_i,y_i)\in W} [\underline{I(x_i+\Delta x, y_i+\Delta y)} - I(x_i,y_i)]^2$$

Sum only over a boolean window W

First order Taylor approximation

Image derivative at (x_i, y_i) along the y axis

$$E(\Delta x, \Delta y) \approx \sum_{(x_i,y_i)\in W} \left[\cancel{I(x_i, y_i)} + [I_x(x_i,y_i) \quad I_y(x_i,y_i)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} - \cancel{I(x_i, y_i)} \right]^2$$

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

$(A^T B)^2 = (A^T B)^T (A^T B)$
$= B^T A A^T B = B^T M B$

$$E_{x_o y_0}(\Delta x, \Delta y) \approx [\Delta x \ \Delta y] \overbrace{\begin{bmatrix} \sum_W \left(I_x(x_i,y_i)\right)^2 & \sum_W I_x(x_i,y_i) I_y(x_i,y_i) \\ \sum_W I_x(x_i,y_i) I_y(x_i,y_i) & \sum_W \left(I_y(x_i,y_i)\right)^2 \end{bmatrix}}^{M} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = [\Delta x \ \Delta y] M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

M is computed from the **first derivatives at each image point** $(x_o y_0)$

$E_{x_o y_0}(\Delta x, \Delta y)$ is a **quadratic polynomial** whose coefficients are the entries of M

Understanding M

Summation of all the square derivatives **along the x-axis** of the points inside W

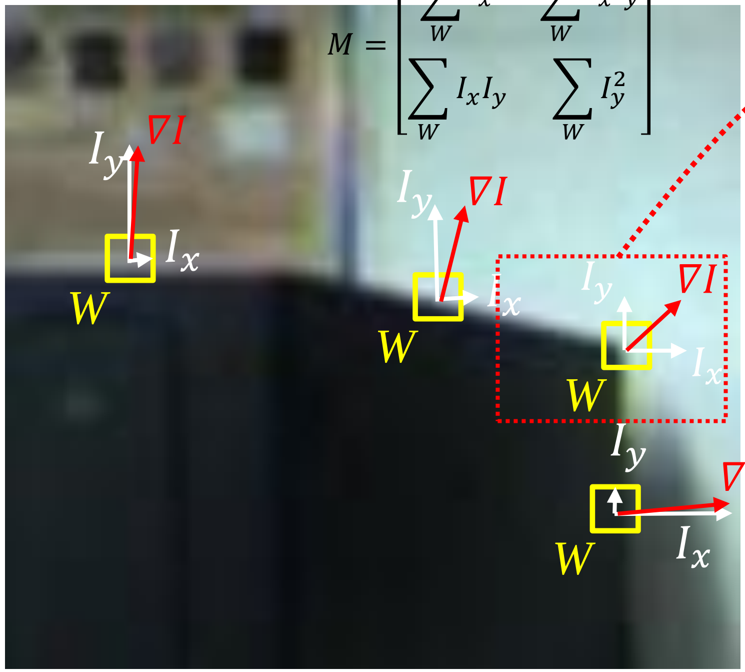
$$M = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}$$

M is computed from the **first derivatives at each image point** $(x_o y_0)$

These two M matrices are at an edge point and have different non-zero structure

$$M = \begin{bmatrix} \sim 0 & \sim 0 \\ \sim 0 & \sum_W I_y^2 \end{bmatrix}$$

$$M = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}$$

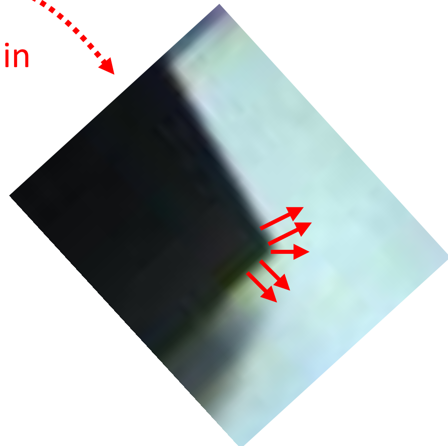


Derivative vector at different pixels (always perpendicular to the border)

$$M = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}$$

$$M = \begin{bmatrix} \sum_W I_x^2 & \sim 0 \\ \sim 0 & \sim 0 \end{bmatrix}$$

Same corner rotated in the direction of ∇I to have a diagonal M



$$M = \begin{bmatrix} \sum_W I_x^2 & \sim 0 \\ \sim 0 & \sum_W I_y^2 \end{bmatrix}$$

M changes when the corners and edges appear rotated in the image

How to get an orientation-invariant matrix M?

$$M = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}$$

Diagonalization of M
(Eigenvector base v_1,v_2)

$$M = VDV^T$$

D is equivalent to M, but derivatives are along the eigenvectors v_1, v_2

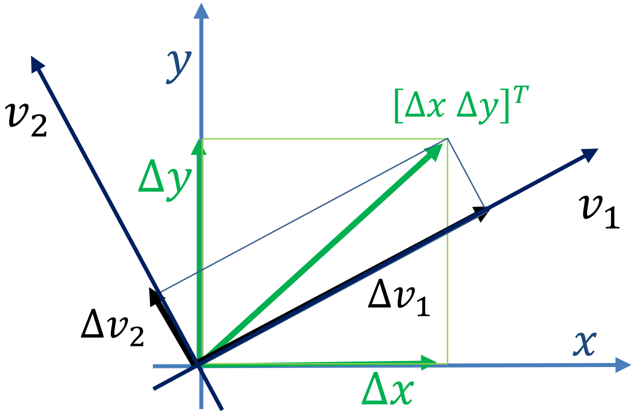
$$D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \quad V = \begin{array}{cc} v_1 & v_2 \\ \left[\begin{matrix} v_{1x} \\ v_{1y} \end{matrix}\right. & \left.\begin{matrix} v_{2x} \\ v_{2y} \end{matrix}\right] \end{array}$$

Summation of all the square derivatives **along $\boldsymbol{v_1}$** of the points inside W

This is a rotation matrix

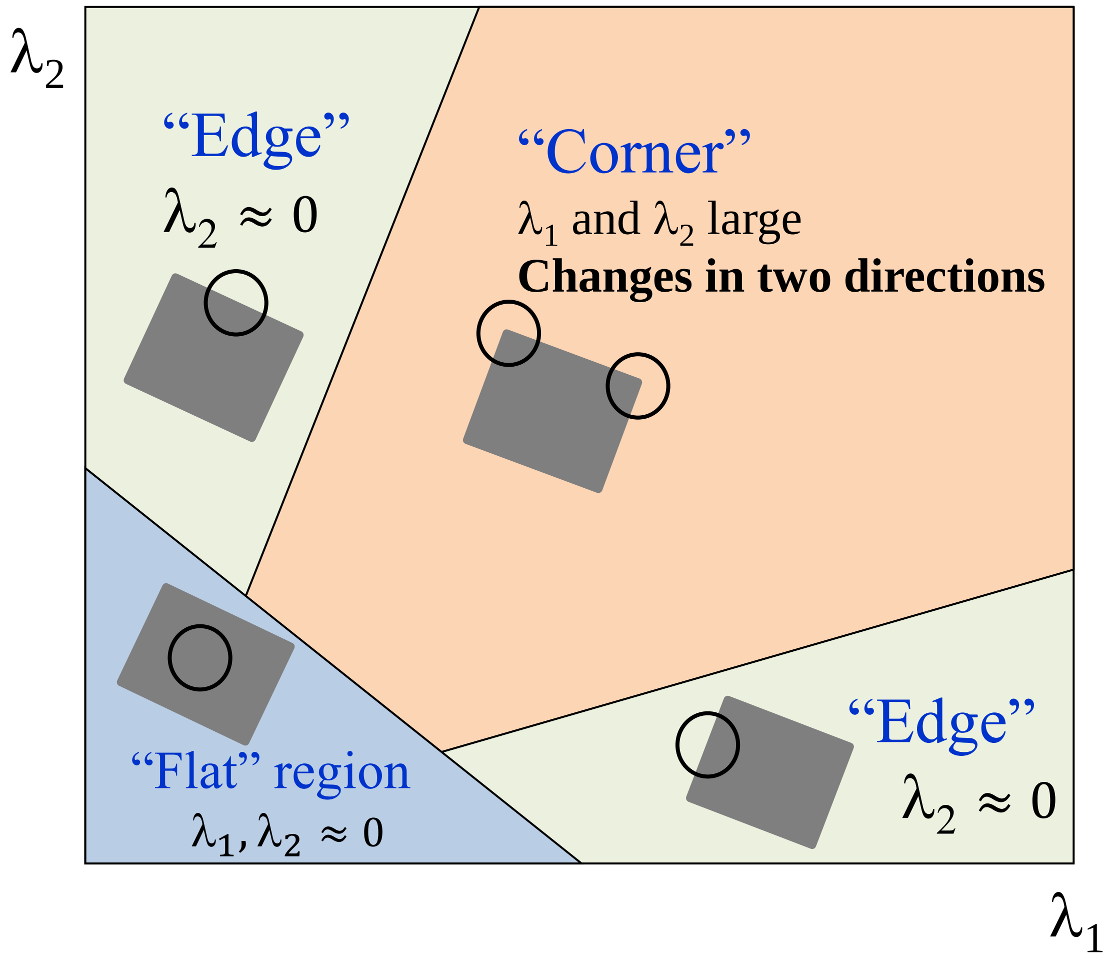
$$E(\Delta x, \Delta y) \approx [\Delta x\ \Delta y] M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \boxed{[\Delta x\ \Delta y] V} D \boxed{V^T \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}} = [\Delta v_1 \Delta v_2] D \begin{bmatrix} \Delta v_1 \\ \Delta v_2 \end{bmatrix} \approx E(\Delta v_1, \Delta v_2)$$

$[\Delta x\ \Delta y]$ rotated by V



2. Harris detector

The image points are now classified according to the eigenvalues



But computing the eigenvalues of a 2x2 matrix (M) at each pixel is costly!

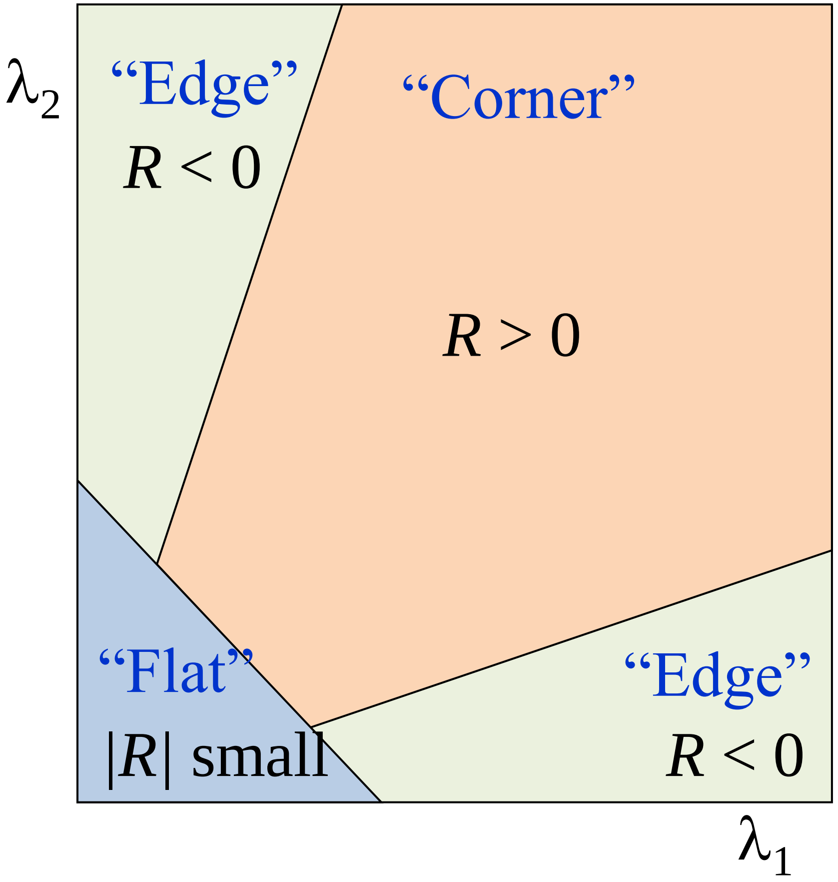
2. Harris detector

Let's define a scalar variable *R* that indexes the same domain:

$$R = \underbrace{\lambda_1 \lambda_2}_{\text{Determinant}} - k\big(\underbrace{\lambda_1 + \lambda_2}_{\text{Trace}}\big)^2$$

(k = 0.04-0.06 is an empiric constant)

- R is large and positive at corners
- R is negative at edges
- $|R|$ is small at flat regions



Trace and determinant of M and $D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$ are the same:

$$\det M = \lambda_1 \lambda_2$$

$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

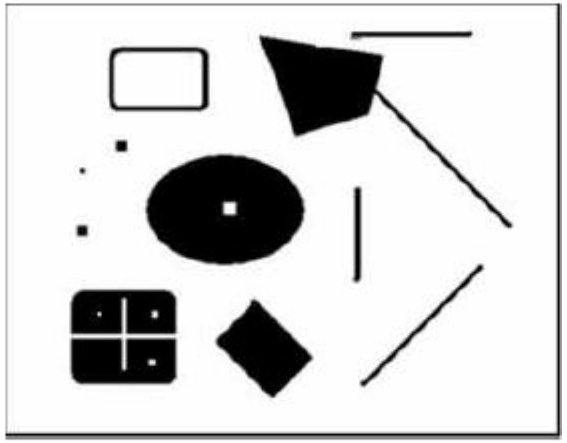
$$\boxed{R = \det M - k\left(\operatorname{trace} M\right)^2}$$

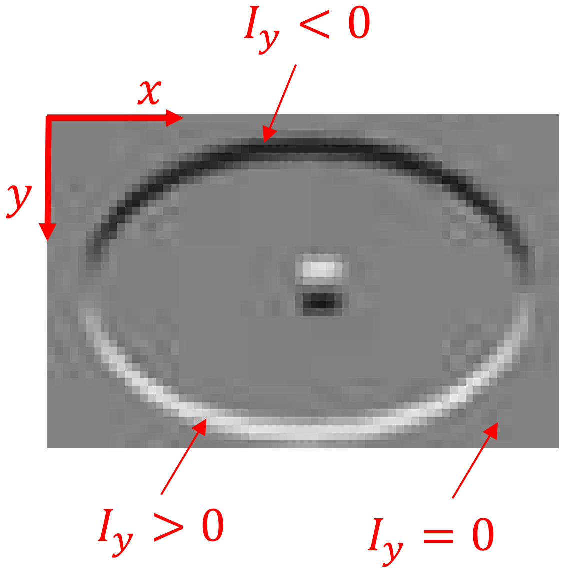
No need to compute the eigenvalues!!

2. Harris detector

Summary:

Original image

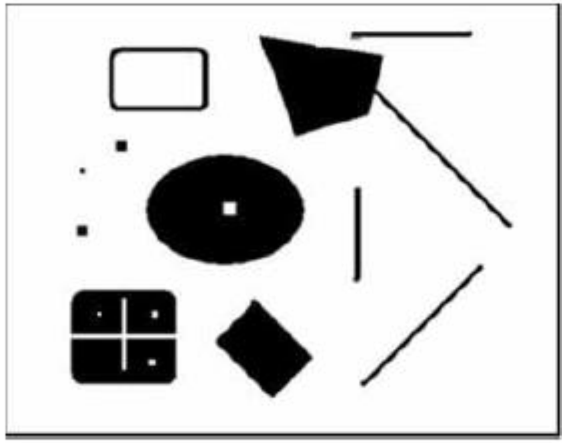


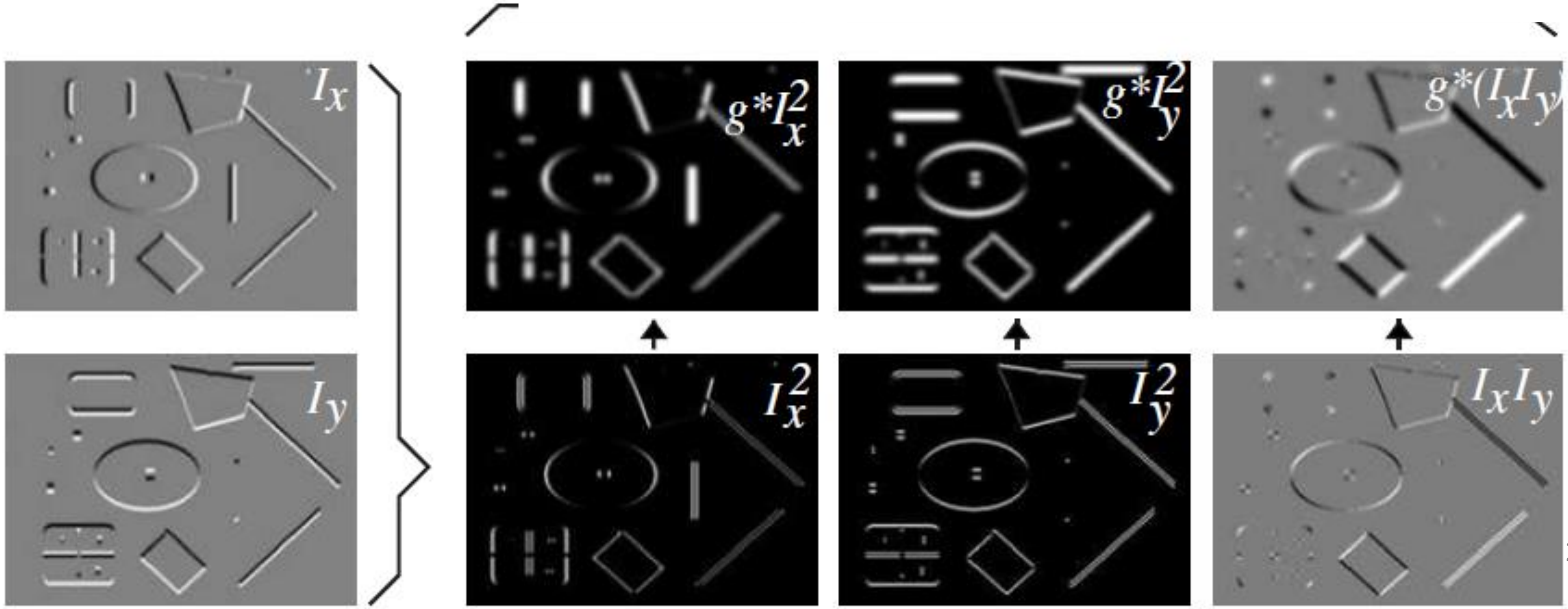


2. Harris detector

Summary:

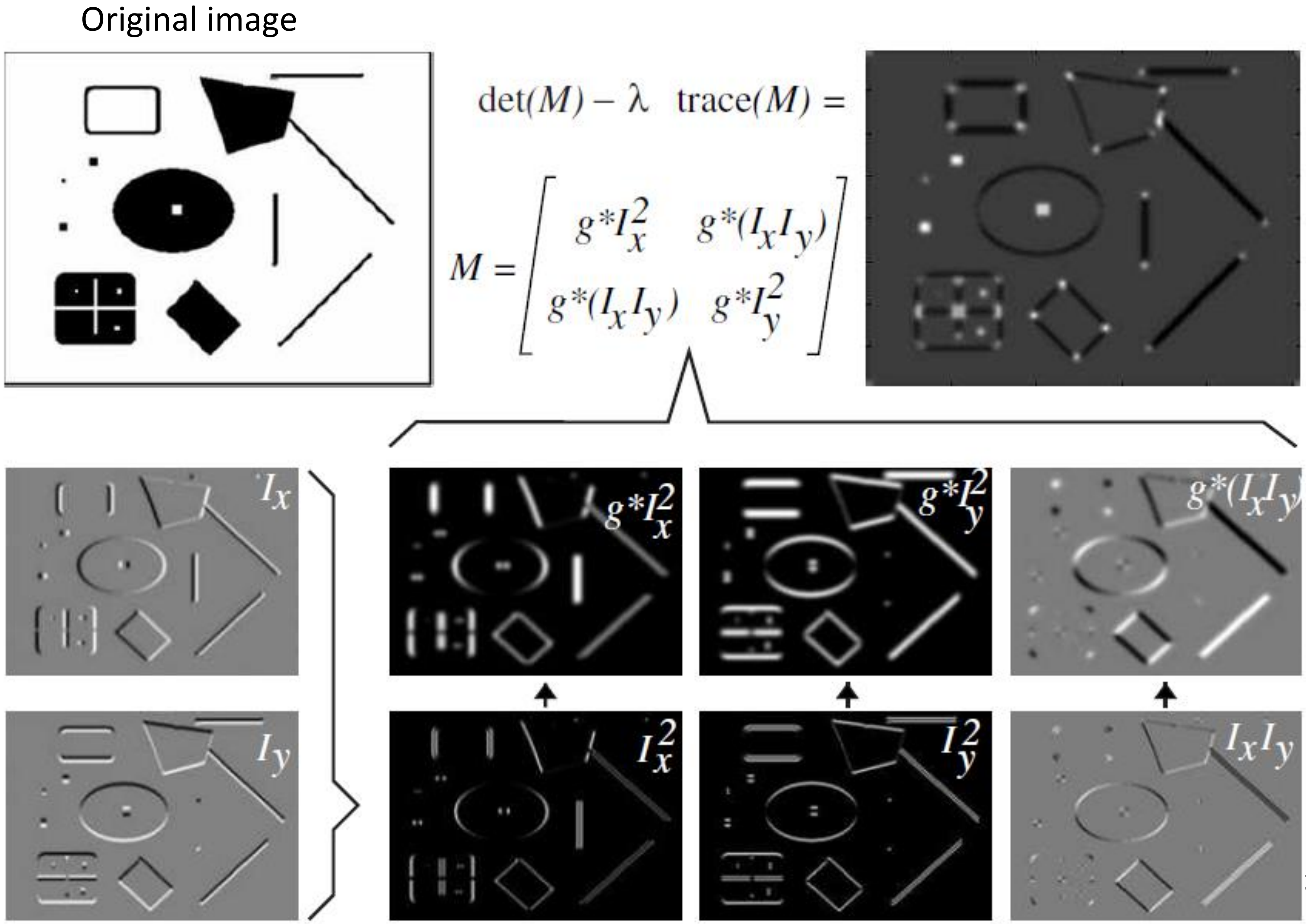
Original image





2. Harris detector

Summary:



2. Harris detector

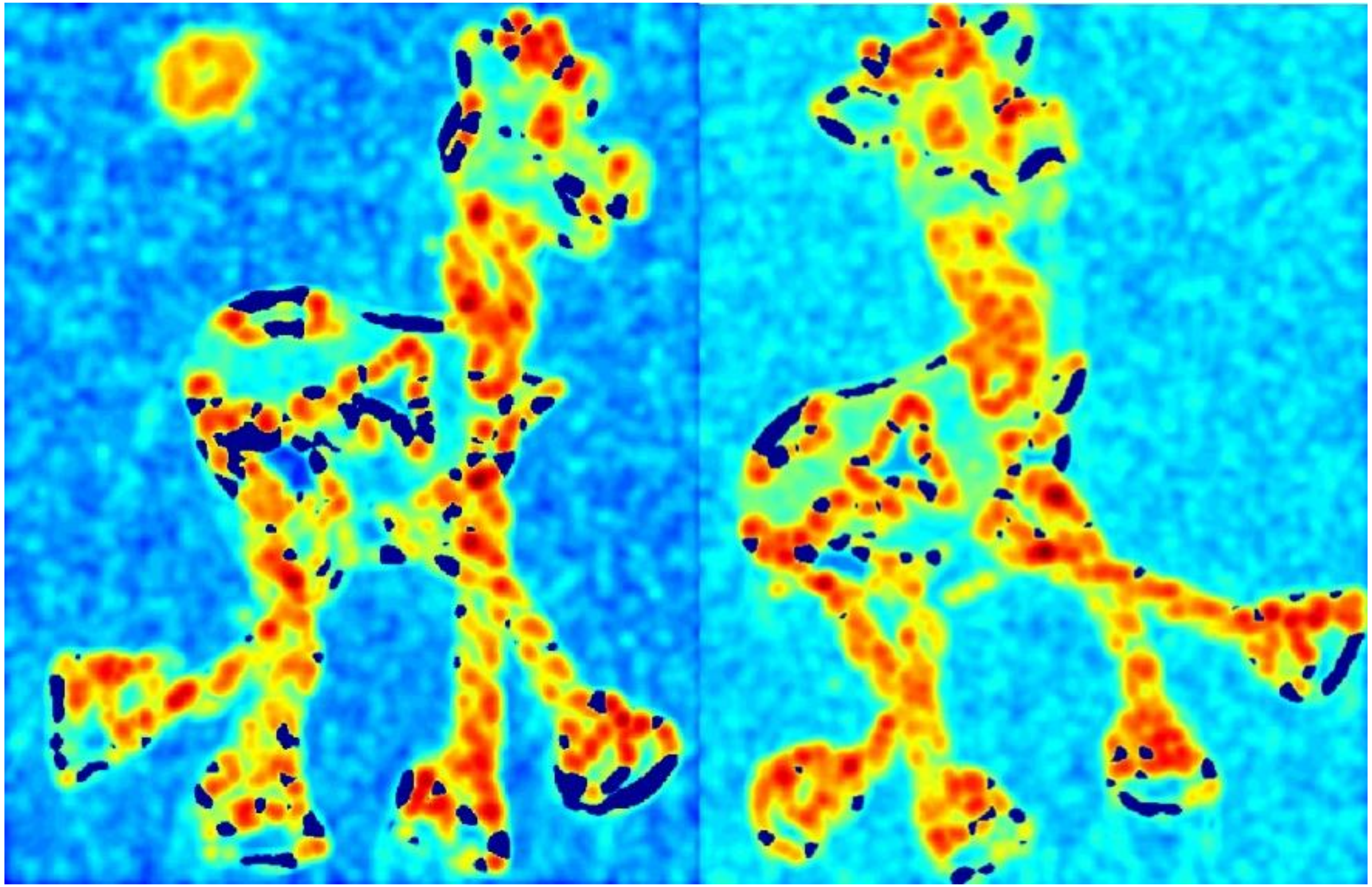
Algorithm:

- Compute the image derivatives I_x, I_y (i.e. Sobel)
- Gaussian smoothing of the 3 images: $(I_x)^2$,$(I_y)^2$, I_xI_y
- Compute the image *R* from the formula (trace and determinant)
- Find regions of pixels where *R* is high (*R* > threshold)
- Select local maxima of each region → the maximum *R* in each

2. Harris detector

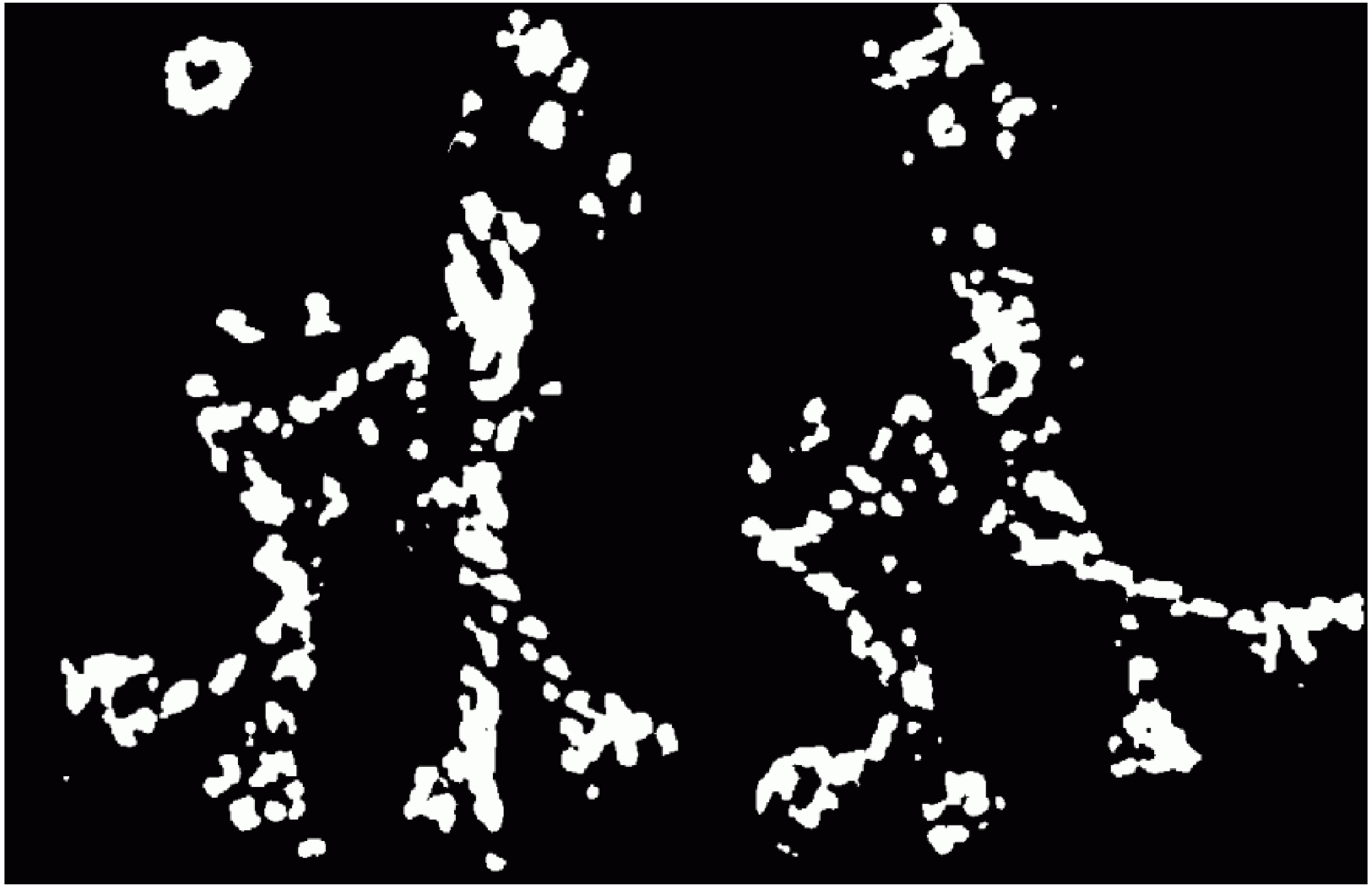
2. Harris detector

R image



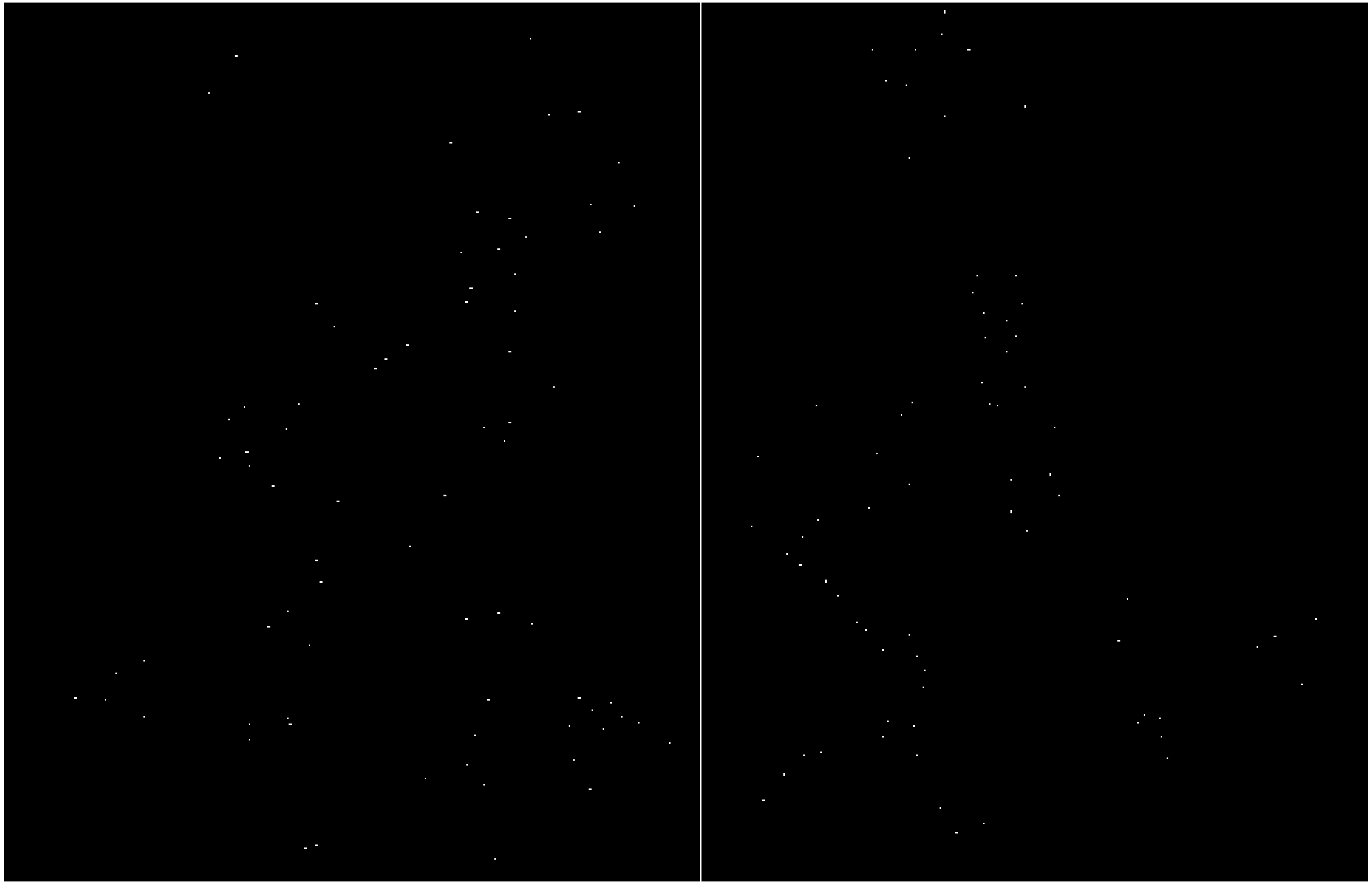
2. Harris detector

R>threshold



2. Harris detector

Local maxima of R



2. Harris detector

3. KLT operator (Kanade-Lucas-Tomasi)

Objective: Detect distintive points, suitable to be tracked in a image sequence (video)

- Similar principle to Harris: "A good keypoint is that with a high intensity derivative in two directions" → **Min (λ_1, λ_2) > threshold**
- Also based on the first derivative matrix:

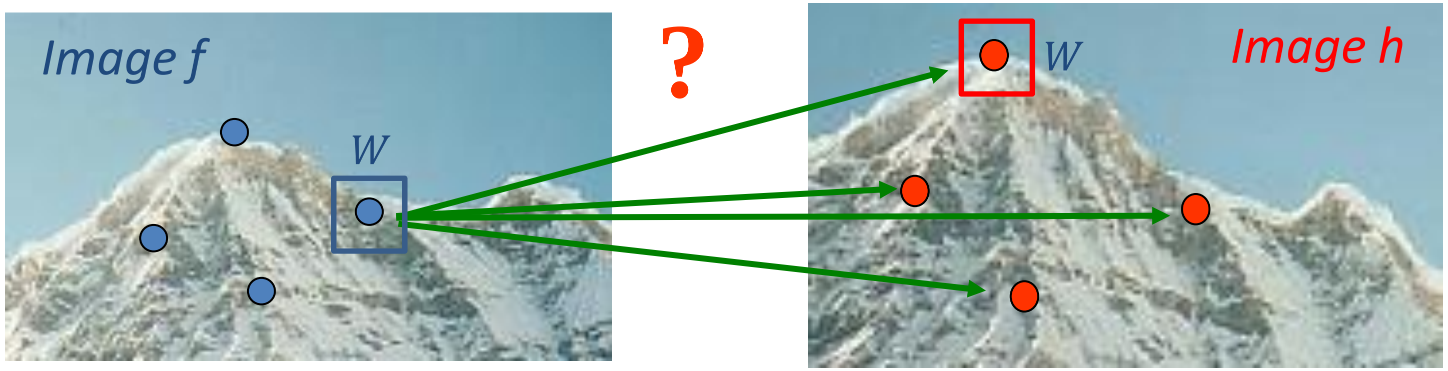
$$M = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix} \longrightarrow D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

- But now, the eigenvalues are computed (no approximation with *R*) → better behavior under affine image deformation

Lines remain parallel

4. Keypoint matching through correlation

Which is the correspondence of each point in other image?



W: *patch around the keypoint*

Sum of Squared Differences (SSD): $SSD(f,h) = \sum_{(i,j)\in W} [f(i,j) - h(i,j)]^2$

SSD is approximated by the SAD (more efficient computationally):

Sum of Absolute Differences (SAD): $\sum_{(i,j)\in W} |f(i,j) - h(i,j)|$

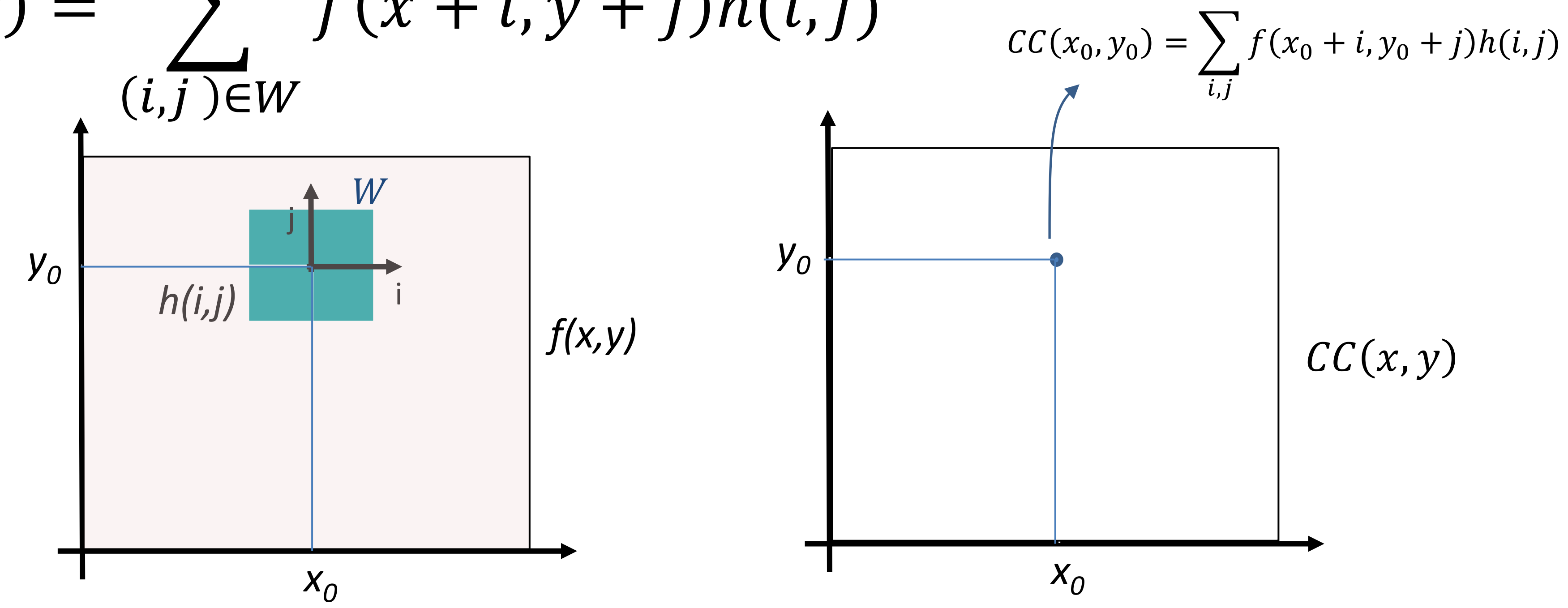
<u>Problem</u>: SSD and SAD are not invariant to brightness or contrast changes

Still, SAD is employed in keypoint tracking along an image sequence, where image brightness and contrast do not change very much from frame to frame.

4. Keypoint matching through correlation

Cross-Correlation (CC):

$$CC(x,y) = \sum_{(i,j)\in W} f(x+i, y+j)h(i,j)$$

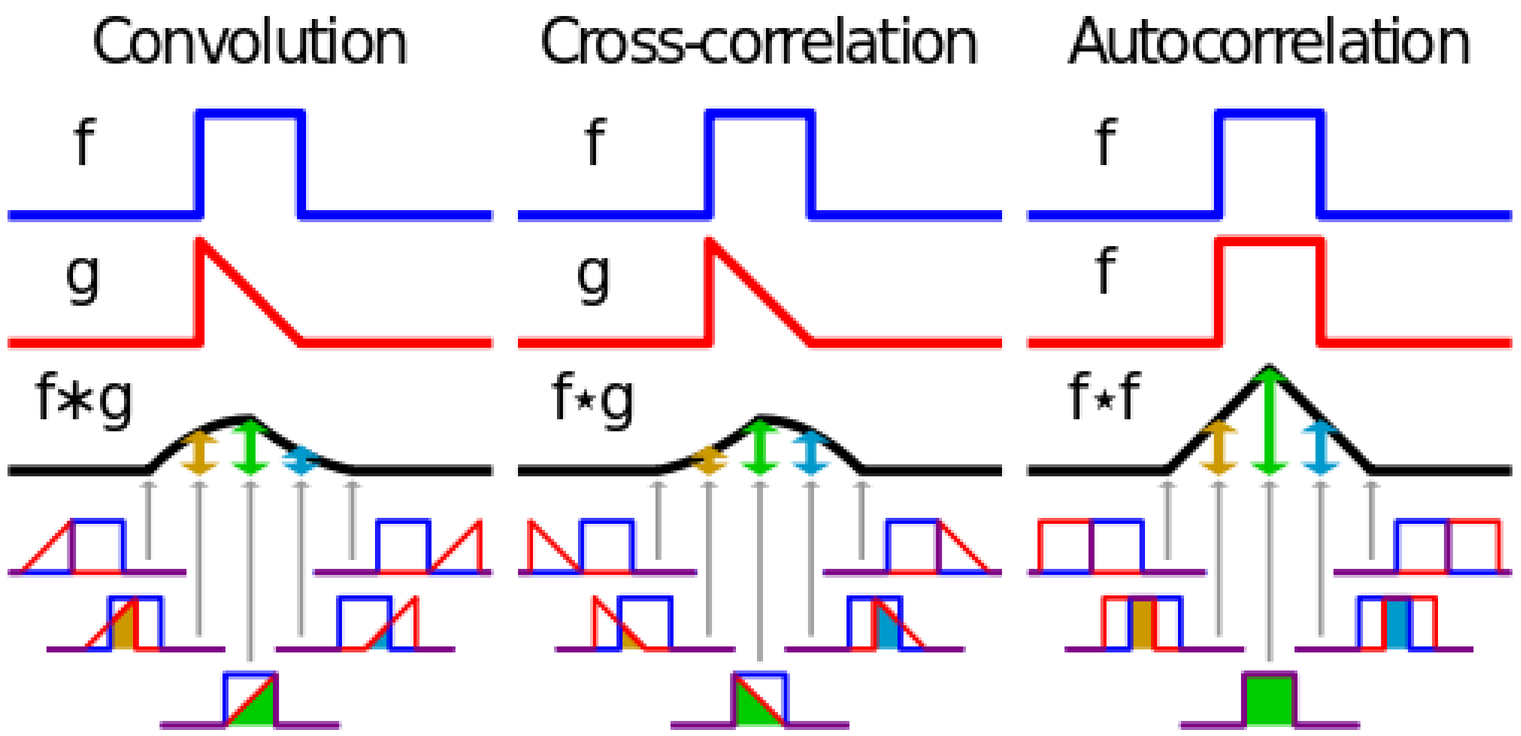


CC is similar to the convolution but without flipping the kernel

Convolution: $(f \otimes h)(x,y) = \sum_{(i,j)\in W} f(x-i, y-j)h(i,j)$

equivalent to the cross-correlation of $f(-i,-j)$ and $h(i,j)$

Correlation vs. Convolution:



The convolution f*g is equivalent to the cross-correlation of f(t) and g(−t)

Normalized Cross-Correlation (NCC):

Cross-correlation is not invariant to changes in brightness and contrast of f and h
→ Normalization required

$$NCC(x,y) = \sum_{i,j} \hat{f}(x+i, y+j)\hat{h}(i,j)$$

Normalization: $\hat{f}$ and $\hat{h}$ have zero mean and contrast one

$$\hat{f}(x+i, y+j) = \frac{f(x+i, y+j) - \bar{f}(x,y)}{\|f - \bar{f}\|_{W(x,y)}}$$

→ Brightness and contrast of f in a window at (x,y)

$$\hat{h}(i,j) = \frac{h(i,j) - \bar{h}}{\|h - \bar{h}\|_W}$$

→ Brightness and contrast of h (constant)

Mean and contrast of f in the window W centered at (x,y)

Mean: $\bar{f}(x,y) = \frac{1}{N}\sum_{(i,j)\in W} f(x+i, y+j)$

Contrast: $\|f - \bar{f}\|_{W(x,y)} = \sqrt{\sum_{(i,j)\in W}[f(x+i, y+j) - \bar{f}(x,y)]^2}$

N: number of pixel in W

Contrast of brightness of h in the window W (constant)

$$\bar{h} = \frac{1}{N}\sum_{(i,j)\in W} h(i,j)$$

$$\|h - \bar{h}\|_W = \frac{1}{N}\sqrt{\sum_{i,j)\in W}[h(i,j) - \bar{h}(i,j)]^2}$$

Why does NCC measure similarity between two image patches?

Let's consider an image patch as a vector

Image patch $f - \bar{f}$ (to have **brightness** normalization)

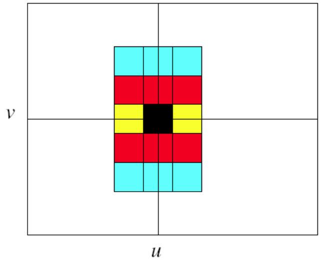
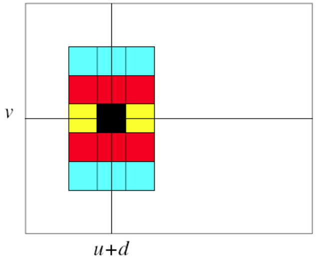


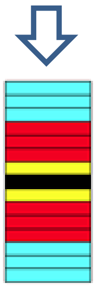
Image patch $h - \bar{h}$

Average brightness

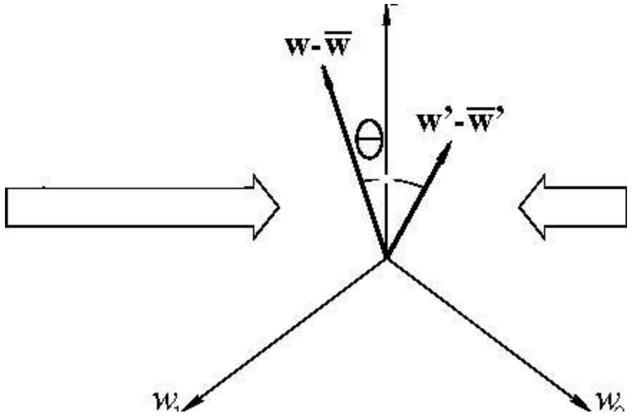


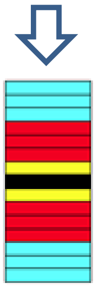
Space dimension = number of pixels in the vector (image patch)

Image patch put as a vector w



$w - \bar{w}$





$w' - \bar{w}'$

Average Brightness

$$\frac{w' - \bar{w}'}{\boxed{|w' - \bar{w}'|}}$$

Unity vector

To have **contrast** normalization

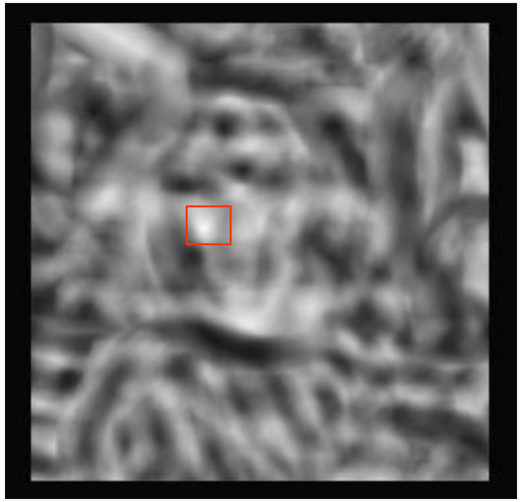
$$C(d) = \frac{1}{|w - \bar{w}|} \frac{1}{|w' - \bar{w}'|} (w - \bar{w}) \cdot (w' - \bar{w}') = \cos\theta$$

The similarity between two unity vectors is given by their dot producto, i.e. angle (or *cos*) between them

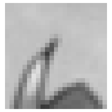
4. Keypoint matching through correlation

*

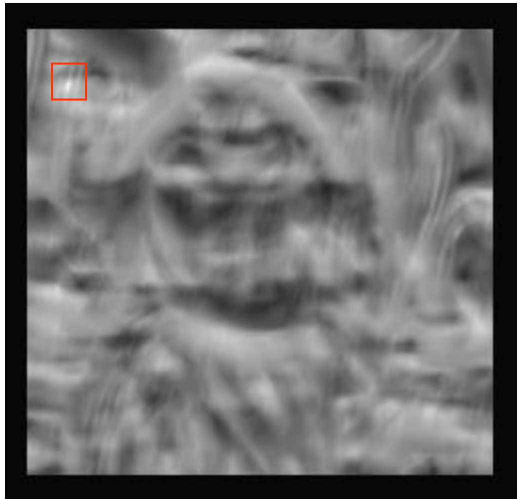
=



*



=



Correlation output

DEMO IN MATLAB

```
%Read and show the image
flowers = imread('flowers.tif'); figure, imshow(flowers)
% select a template from the image with the mouse
[sub_flowers,rect_flowers] = imcrop(flowers);
% Show the selected template
figure, imshow(sub_flowers)
% Do NCC with the blue channel and display the result
c = normxcorr2(sub_flowers(:,:,1),flowers(:,:,1));
figure, surf(c), shading flat
```

Notice: Output not invariant to the rotation of the patch

Content

1. Introduction
2. Harris detector
   - Idea
   - Formulation
   - Implementation
3. KLT operator
4. Keypoint matching through correlation
5. SIFT operator
   - Scale Space
   - Detector
   - Descriptor

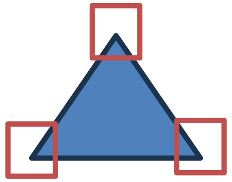
5. The SIFT (**S**cale **I**nvariant **F**eature **T**ransform) operator

- **Objective**: Find blobs (not corners!) that are invariant to scale
  (Scale means image (object) size)

Big image size

Small image size

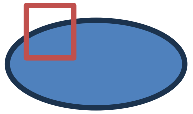
Harris



3 corners at this scale

The corners disapear at this scale. The riangle becomes a point

SIFT



No corner is detected at this scale

Same blob at a much smaller size. A distintive point is detected at this scale

- **Provides** both: Detector y descriptor of the detected keypoints

Proposed (and patented) by **David Lowe**: **"Distinctive image features from scale-invariant keypoints,"** *International Journal of Computer Vision,* 60, 2 (2004).

Some slides borrowed from: Francisco A. Moreno Dueñas

5. The SIFT operator

- Both, detector and descriptor have invariance to:
  - Scale
  - Rotation
  - Ilumination [constrast+brightness]
  - Affine transformation [parcially]

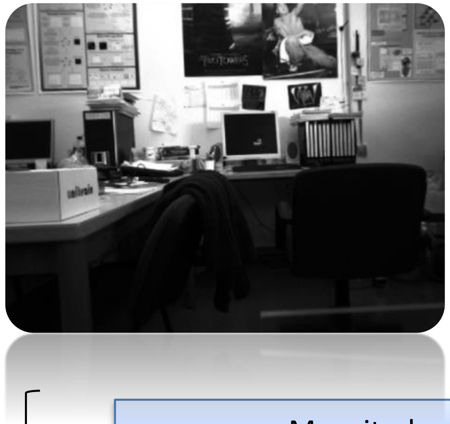
Important difference from Harris

Important difference from NCC

- Principle
  - Search for extrema in the scale space [**Detector]**
  - Normalized histogram of orientation [**Descriptor**]
- Descriptor is a vector up to 128 elements
- SIFT is employed in important whole-image descriptors as Bag-of-Words (BoW) or VLAD, for Image Retrival and Place Recognition

5. The SIFT operator

Overview of the method



Build scale space

Invariance to Scale

Difference of images

Search for extrema

Filter out
- Low contrast extrema
- Extrema at image borders

Descriptor computation

Descriptor computation in 4 steps

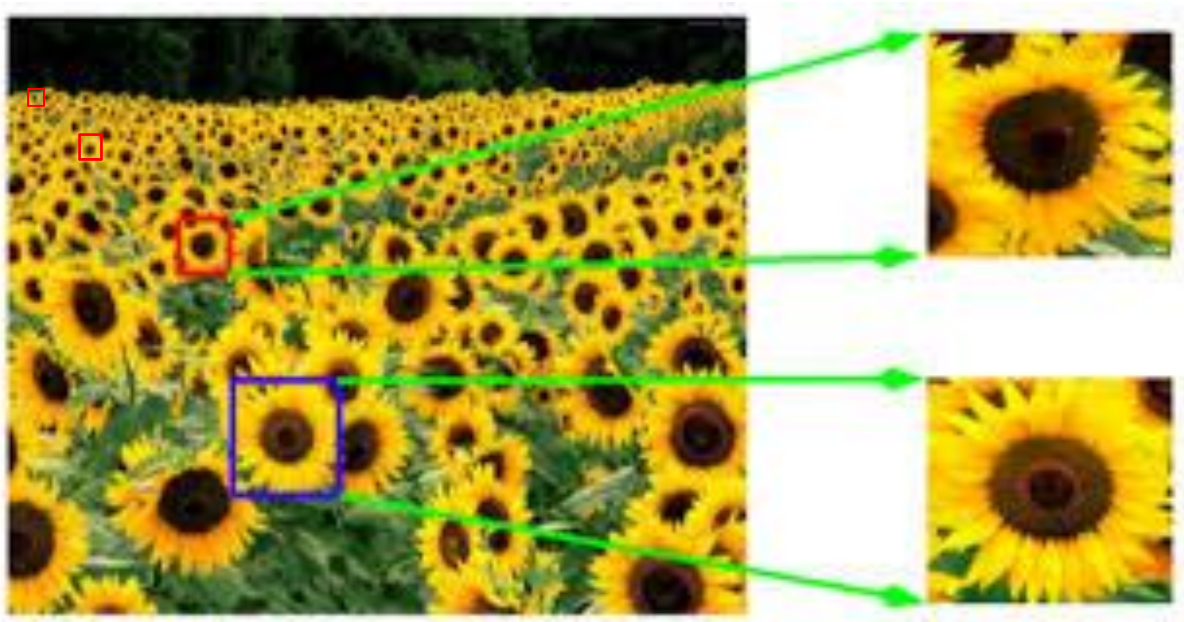
- Magnitude of the gradiente
- Dominant orientation of the gradient
  Invariance to Orientation
- Histogram of orientation
- Contrast and brightness normalization
  Invariance to Illumination

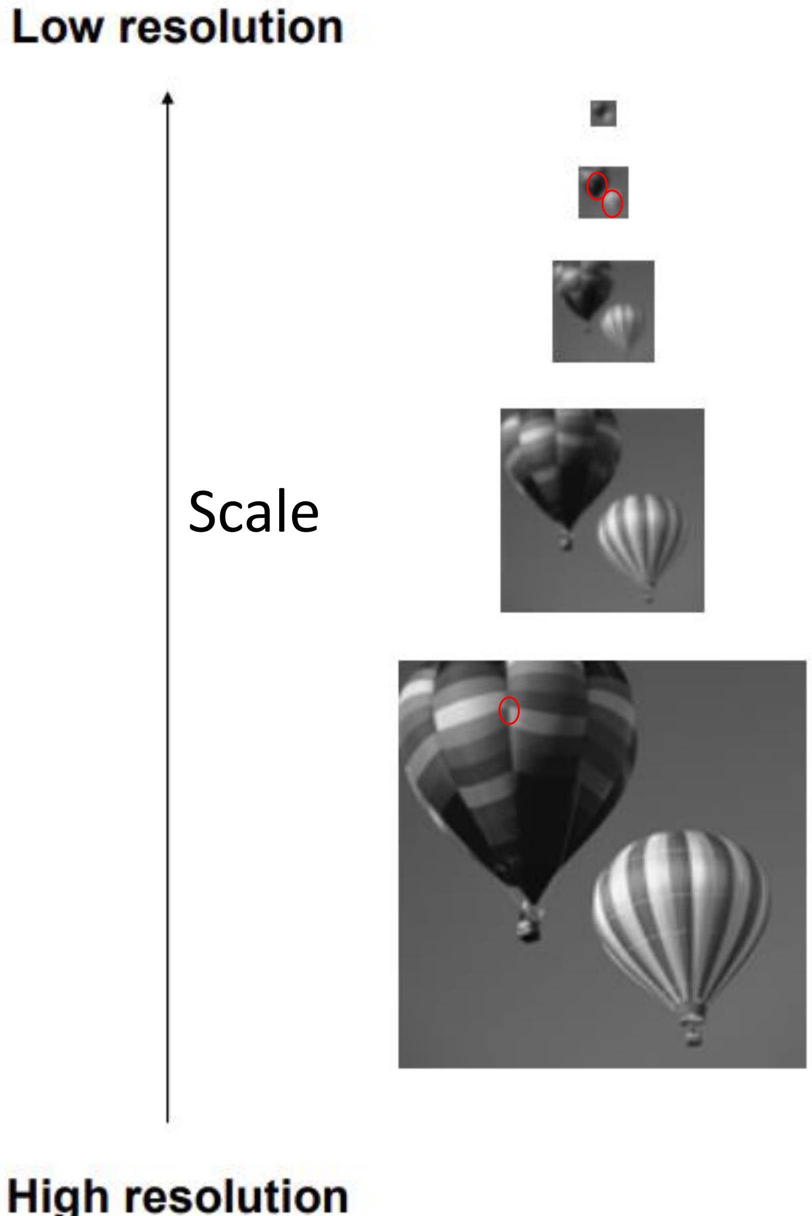
5. The SIFT operator

Changing the scale (size) shows up different features

Scale Space

In images, features emerge at different scales



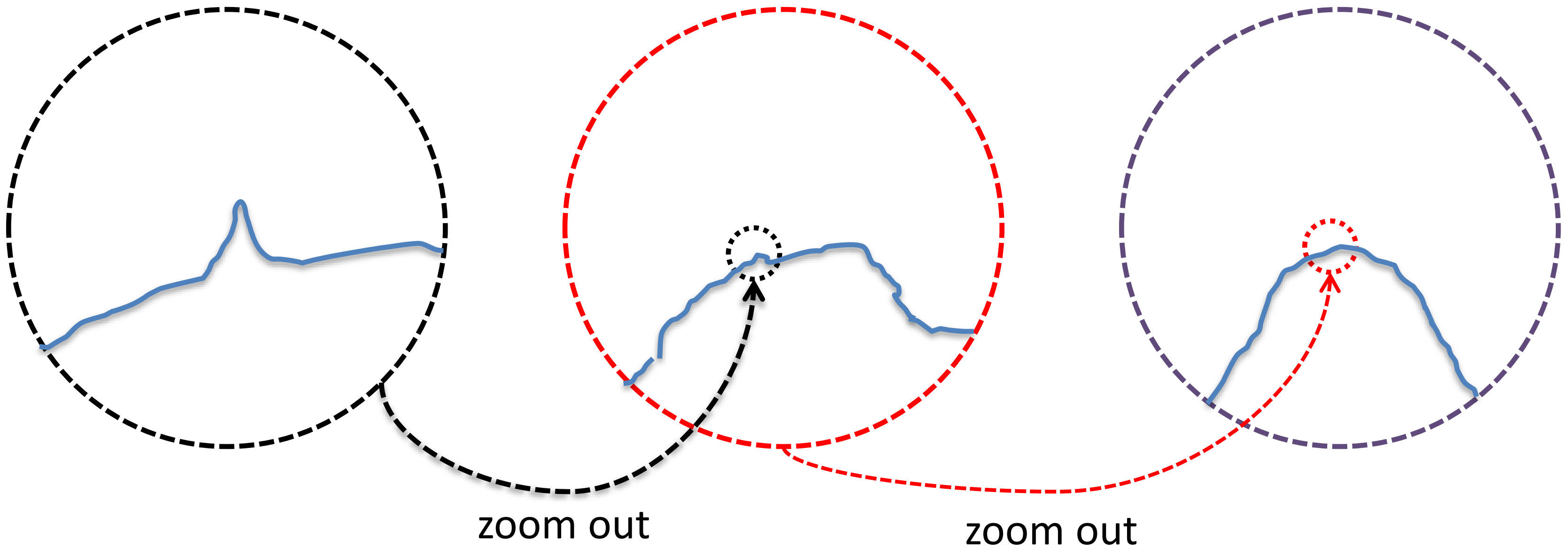


5. The SIFT operator

Scale Space

Features show up at a given scale

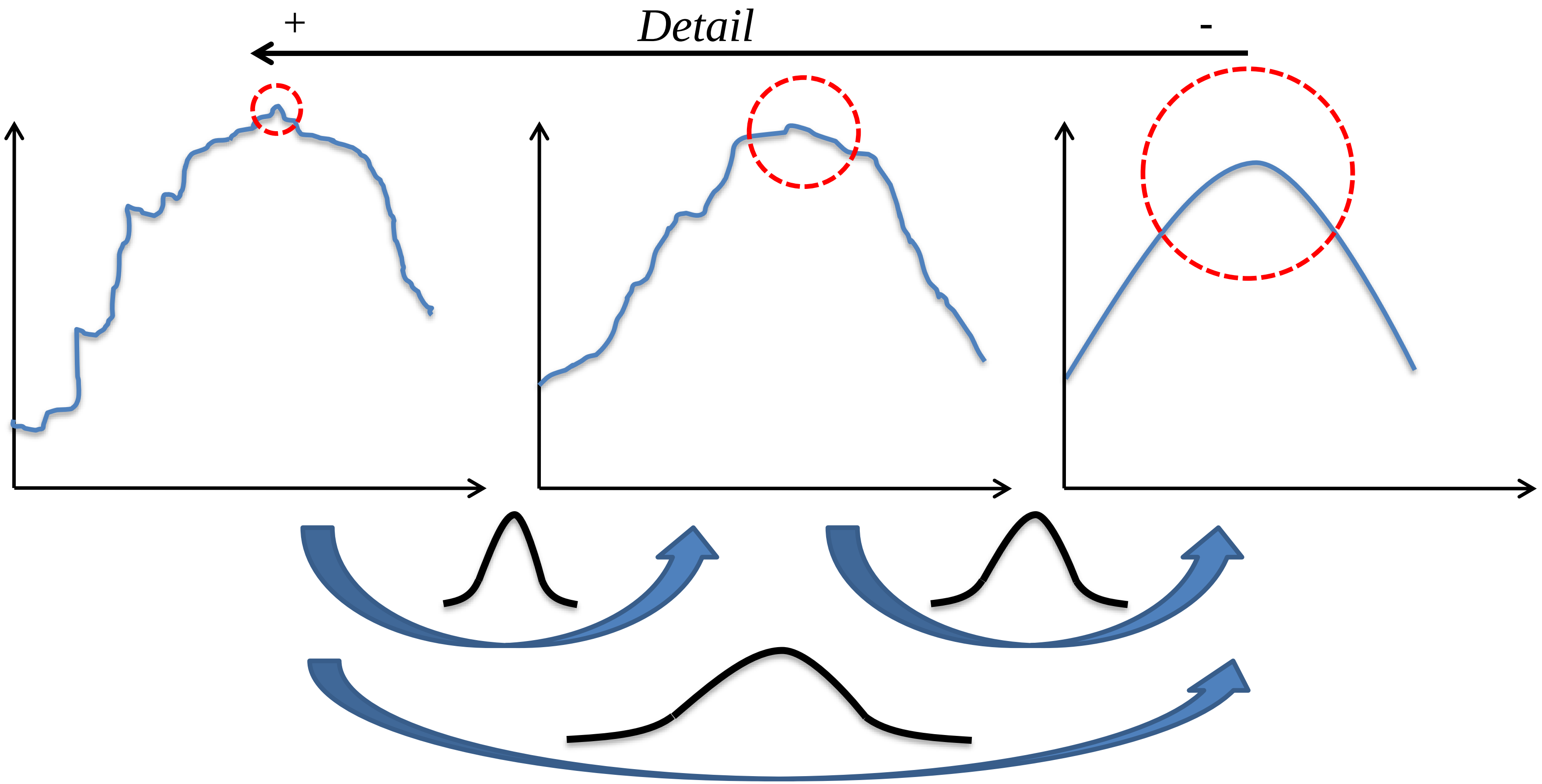
Example in one-dimension



5. The SIFT operator

Scale Space

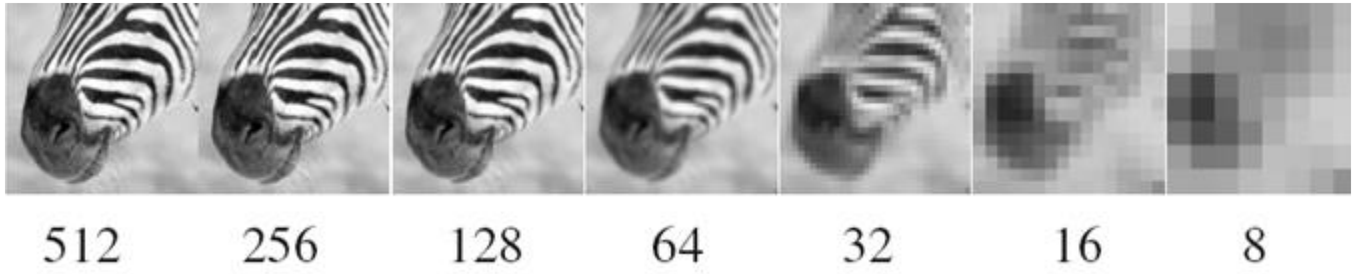
We can change the scale by smoothing the signal with a Gaussian



5. The SIFT operator

Scale Space

Changing the scale of an image by smoothing with a Gaussian



Scale Space

Changing the scale of an image by smoothing with a Gaussian

Gaussian operator: $G(x,y,t) = \frac{1}{2\pi t} e^{-\frac{(x^2+y^2)}{2t}}$ $\quad t = \sigma^2 \Rightarrow \sigma = \sqrt{t}$

Smoothed image: $L(x,y,t) = I(x,y) * G(x,y,t)$

\+ *Detail* -

The starting image is always the original image (*t*=0)

$L(x,y,0) = I(x,y)$

starting image *t*=0

$L(x,y,1)$

σ=1

$L(x,y,4)$

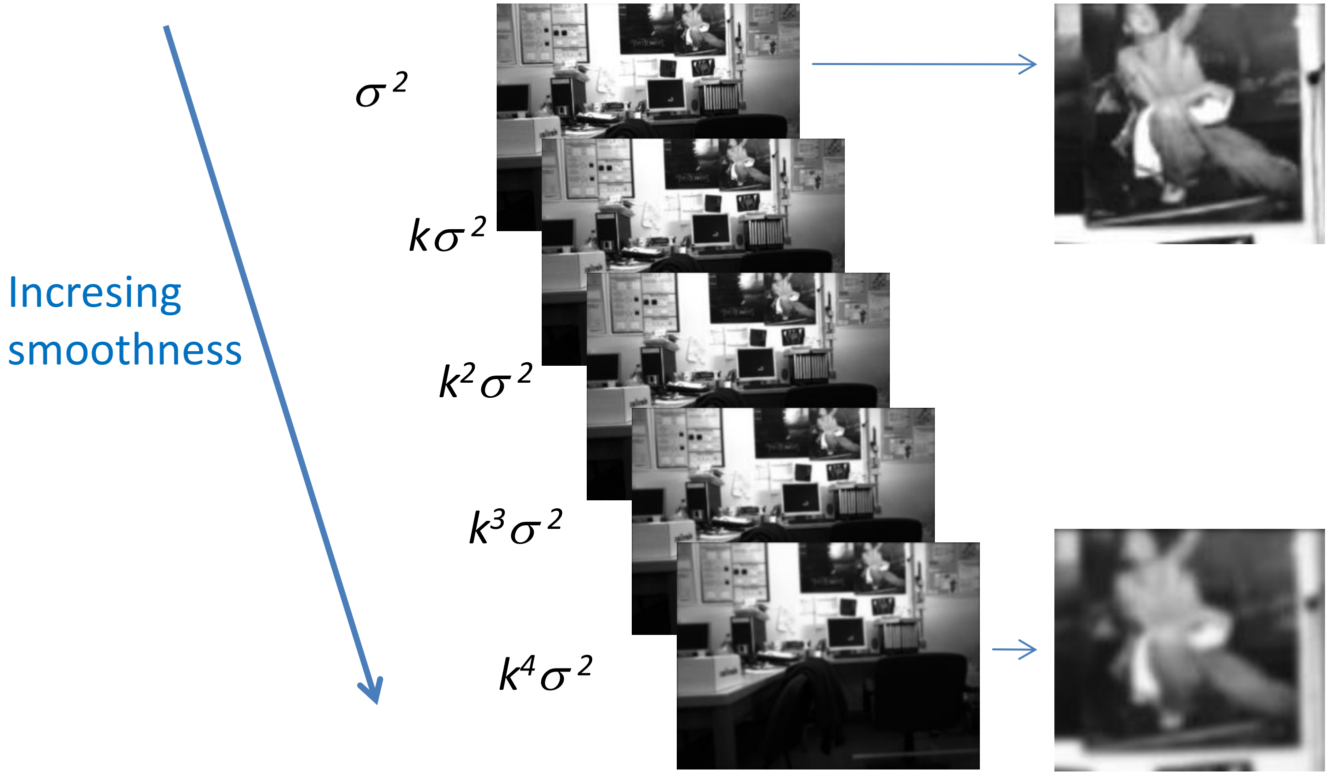
σ=2

So, the scale space stores samples of the function $L(x,y,\sigma^2)$

5. The SIFT operator

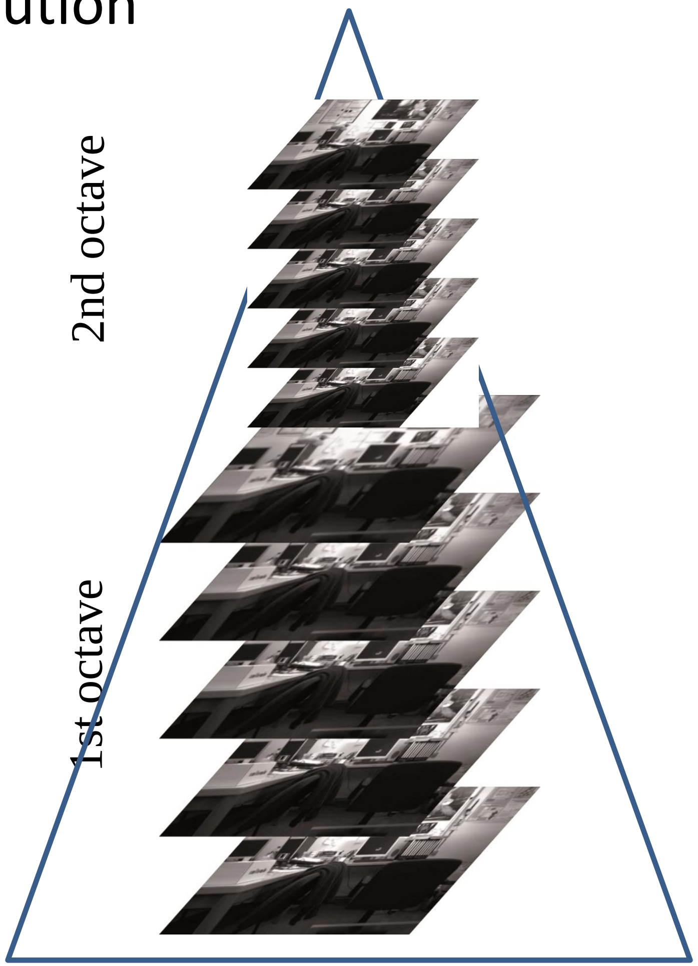
Objetive: “continuous” scale space

Progressive convolution of the input image with a Gaussian controlled with a constant factor K



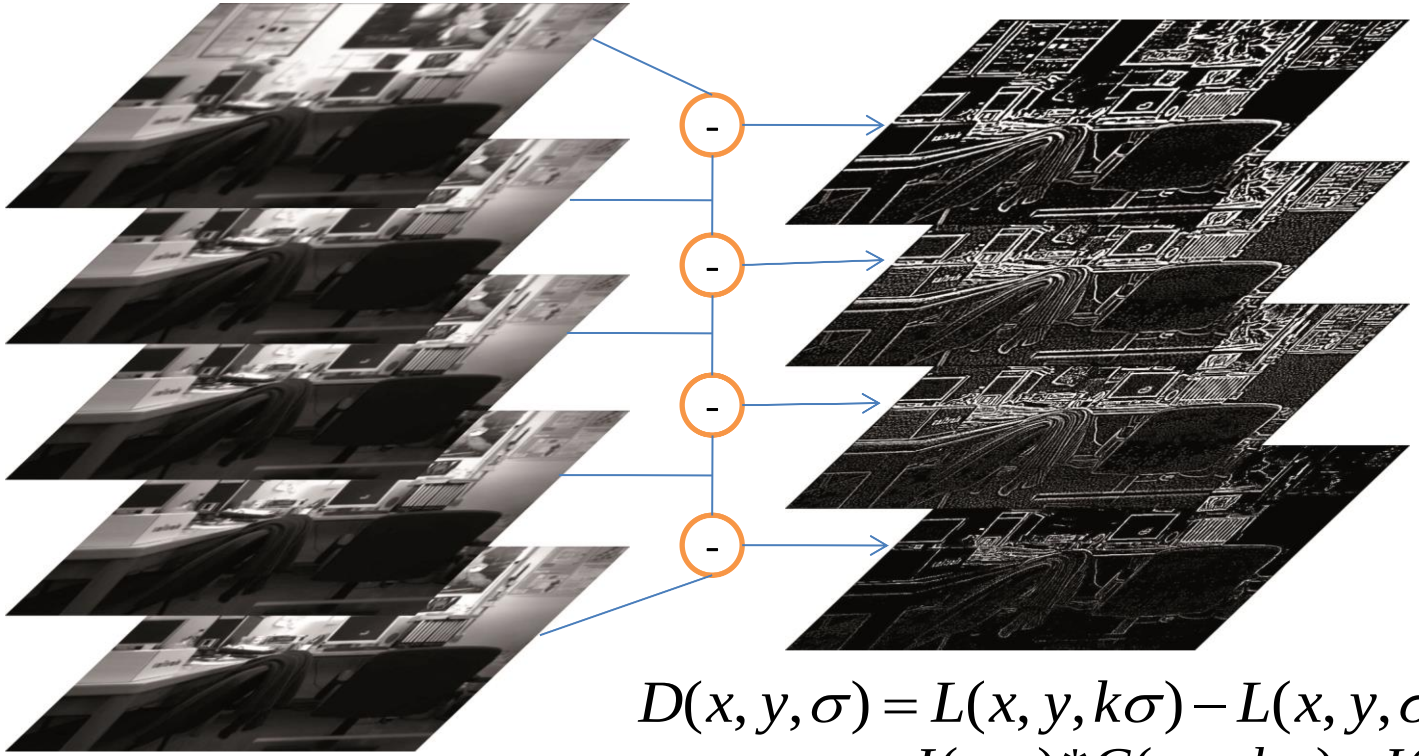
5. The SIFT operator

- The scale space has the structure of a ***pyramid***: a collection of digital images sampled at progressively coarser spatial resolution and hence of progressively smaller size.
- The pyramid is built upon a number of ***Octaves.***
- Each ***Octave (o)*** consists of s+2 images of the same size (resolution) with increasing smoothness (typically, s= 3).
- In the **following *Octave*** the image has half the resolution (size) since it does not make sense to keep the resolution when small details have been removed.



5. The SIFT operator

From smoothed images to **Difference of Gaussians** (DoG)



The DoG images at the ends are also in the neighbors octaves. See next slide.

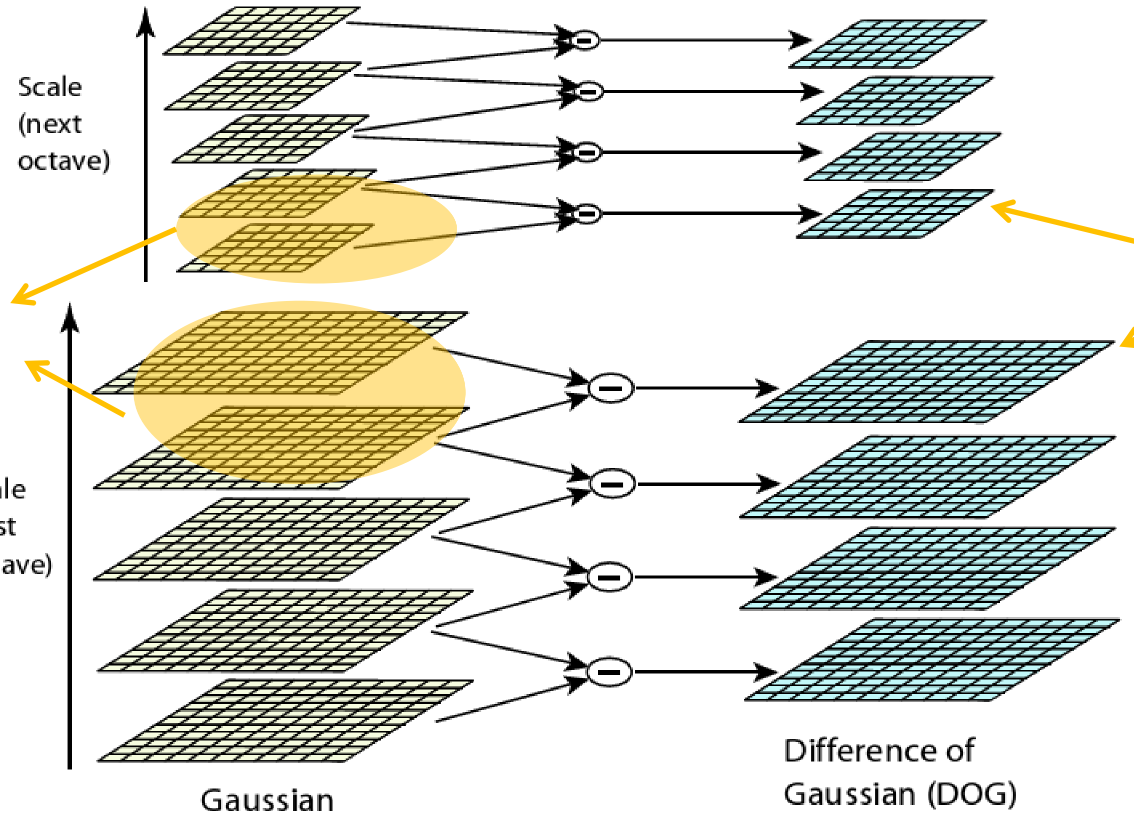
$$
\begin{aligned}
D(x,y,\sigma) &= L(x,y,k\sigma) - L(x,y,\sigma) \\
&= I(x,y) * G(x,y,k\sigma) - I(x,y)G(x,y,\sigma) \\
&= I(x,y) * \left[G(x,y,k\sigma) - G(x,y,\sigma)\right] \\
&= I(x,y) * DoG(x,y,\sigma)
\end{aligned}
$$

Diference of smoothed images = Image convolved with a DoG

5. The SIFT operator

Construction of the scale space through “octaves”

These two images are the same but with half resolution



Same DOG images but half resolution

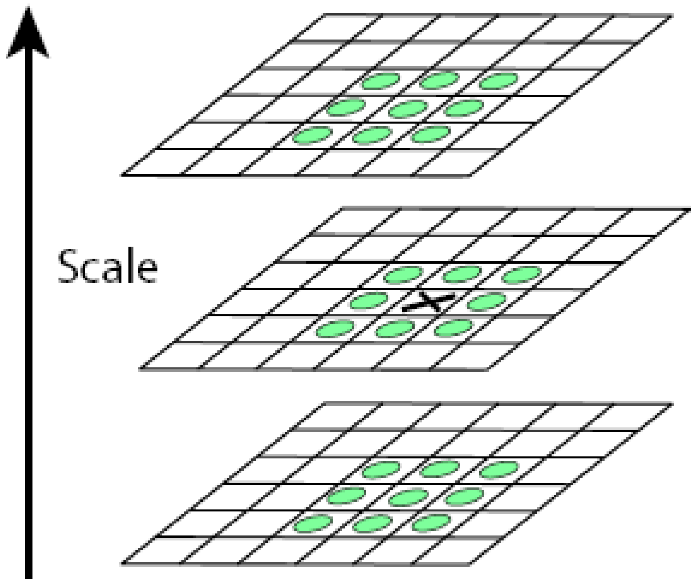
DoG are used here to detect BLOBS (not edges)!!

5. The SIFT operator

Search for extreme points

Each pixel value is compared to its 26 neighbors along the full scale:

- **8** in ithe same scale,
- **9** in the upper scale and
- **9** in the lower scale

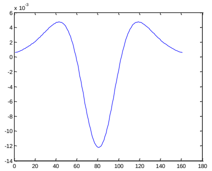


An extreme point in the scale gives us a distintive point in (x,y) and in the pyramid

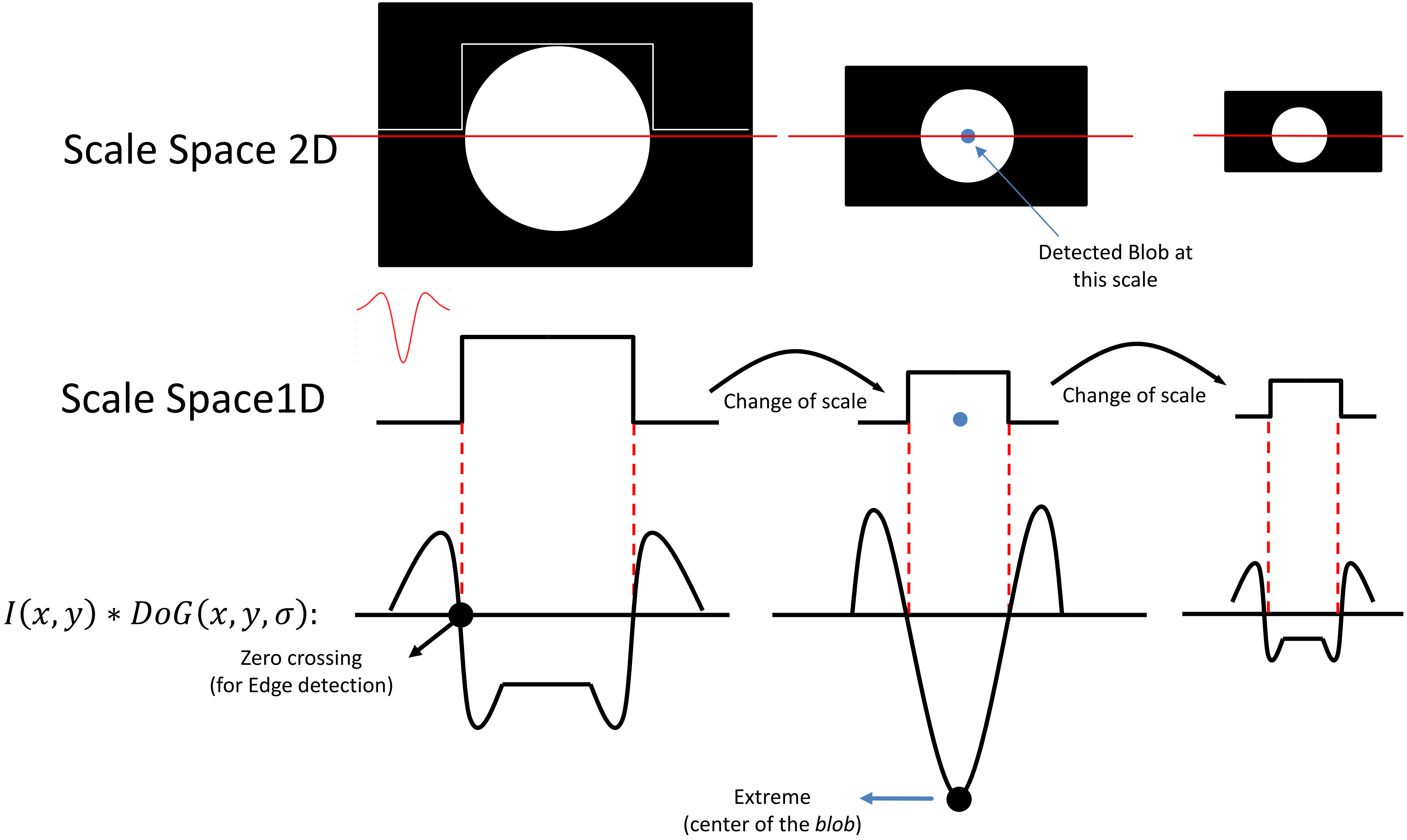
A extreme point in the scale gives us a distintive point in (x,y) and in the pyramid (k)

WHY?

Recall:

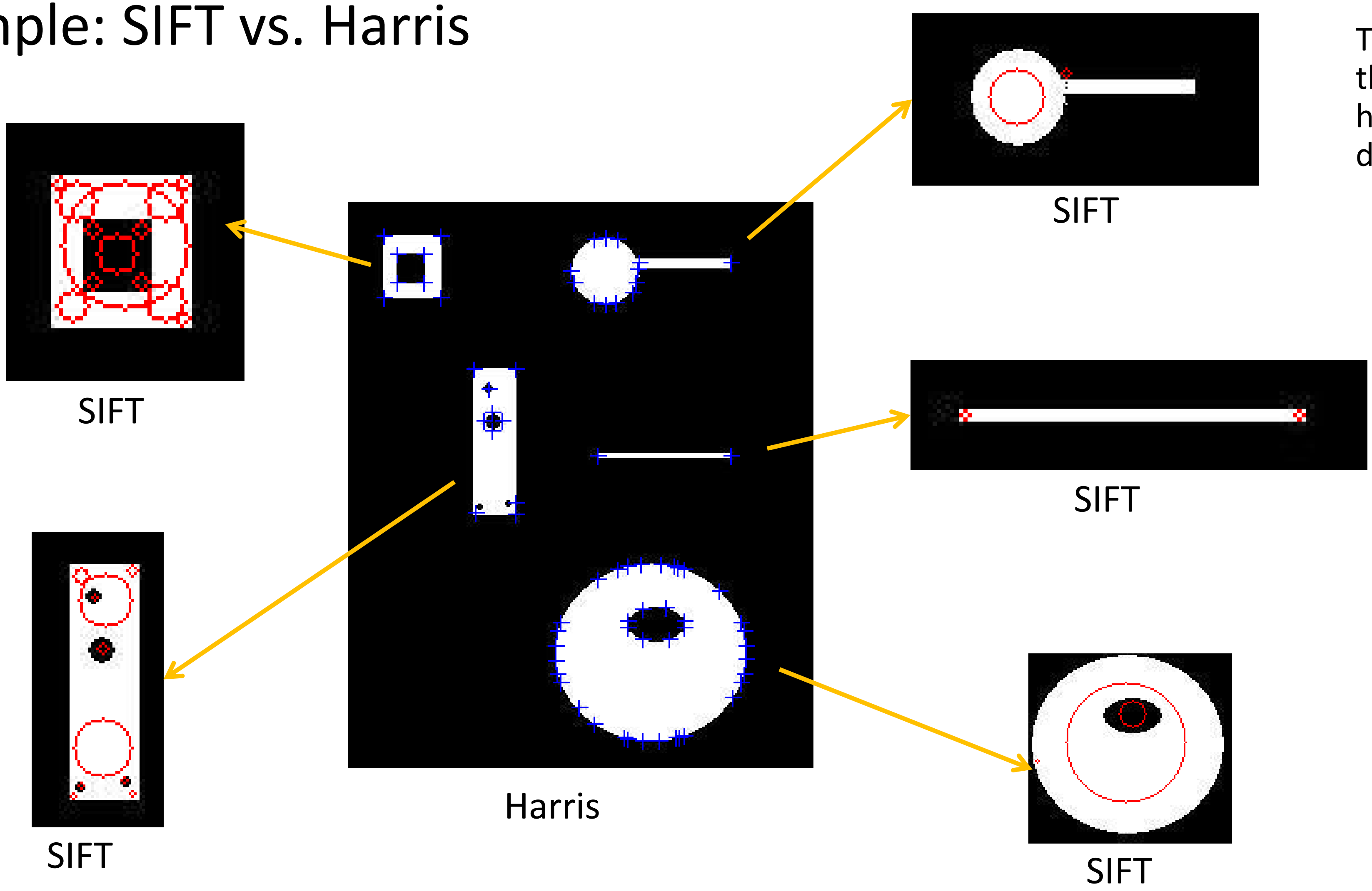


DoG operator (1D)



5. The SIFT operator

Example: SIFT vs. Harris



The size of the circle indicates the scale where the extrema has been detected: bigger diameter → less resolution

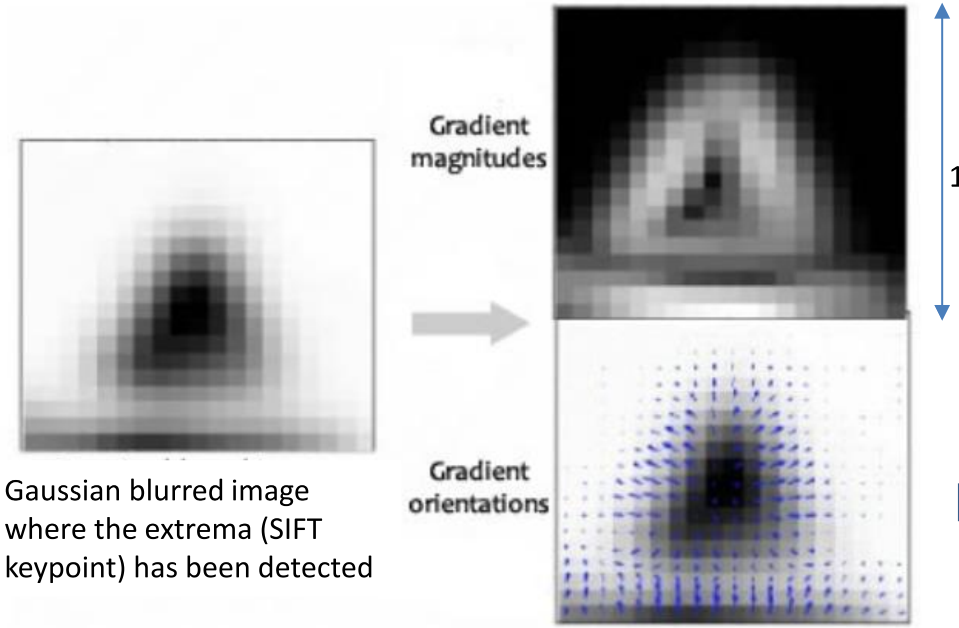
SIFT Descriptor: Histogram of orientations around the extreme point

Obtaining the descriptor orientation:

The magnitude (*m*) and orientation (θ) of the gradient is calculated for all pixels around the detected keypoint (in the smoothed image where the extreme was detected)

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

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

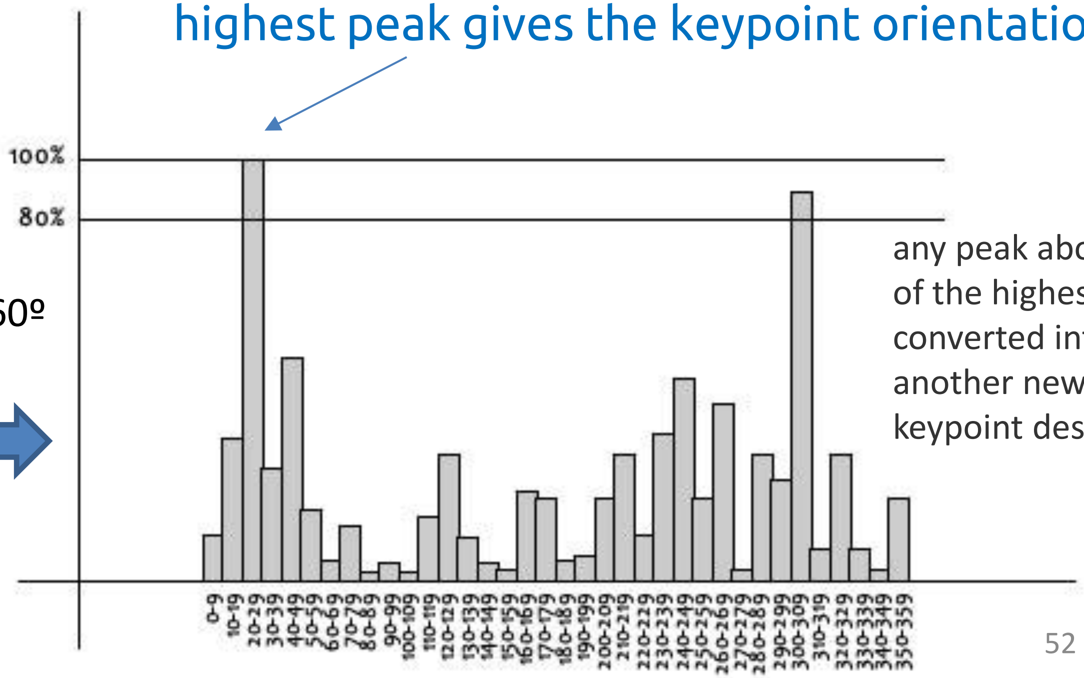


Gaussian blurred image where the extrema (SIFT keypoint) has been detected

36 bins for 360º (every 10º)

A histogram is created for these orientations, weighted by the magnitude of the gradient

highest peak gives the keypoint orientation



any peak above 80% of the highest is converted into another new keypoint descriptor

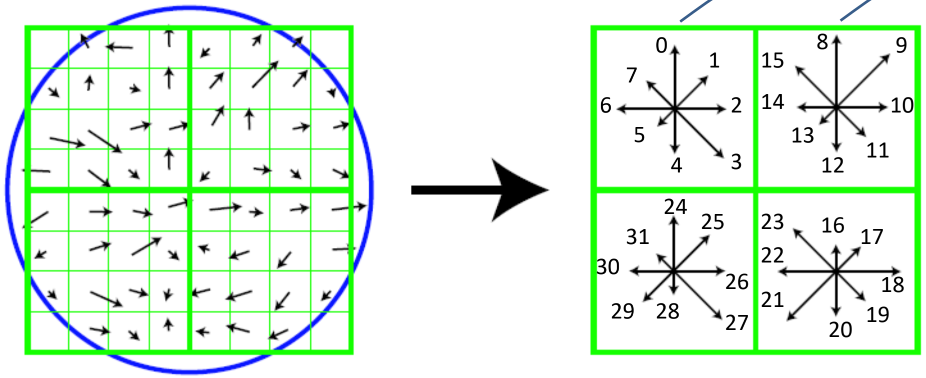
SIFT Descriptor: Histogram of orientations around the keypoint

Obtaining the descriptor vector:

- The neighborhood of the keypoint is divided in **m** x **m cells** (a cell is 4x4 pixels)
- Histogram of 8 orientations for each cell → Size of descriptor vector: **mxmx8**

Example: m = 2 -> Descriptor vector: 2x2x8= 32 entries

m = 2



$v = (0,1,\ldots,7,8,\ldots,15,16,\ldots,23,24,\ldots,31)^T$

In the Lowe's original paper (D. Lowe): **m = 4 -> Descriptor**: 4x4x8 = **128D**

m = 4

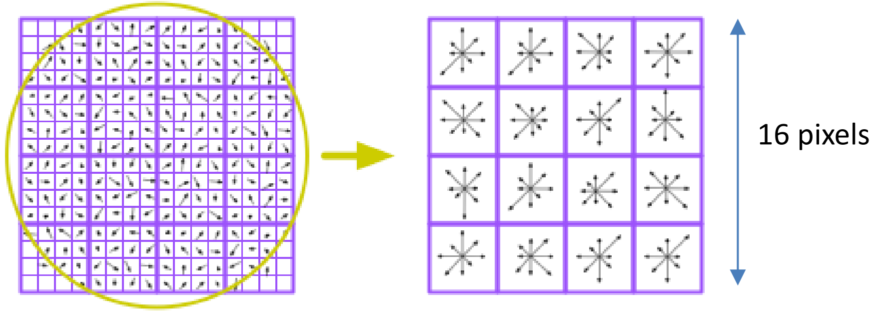
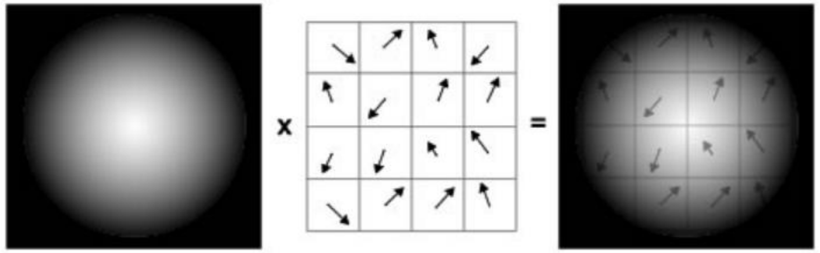


Image gradients　　Keypoint descriptor

Obtaining the **descriptor vector**:

The histogram of orientations is weighted by

- Magnitude of the gradient: more importance to strong gradients
- Gaussian centered at the extreme point: more importance to close pixels



Descriptor: 4x4x8 = **128D**

SIFT Invariances

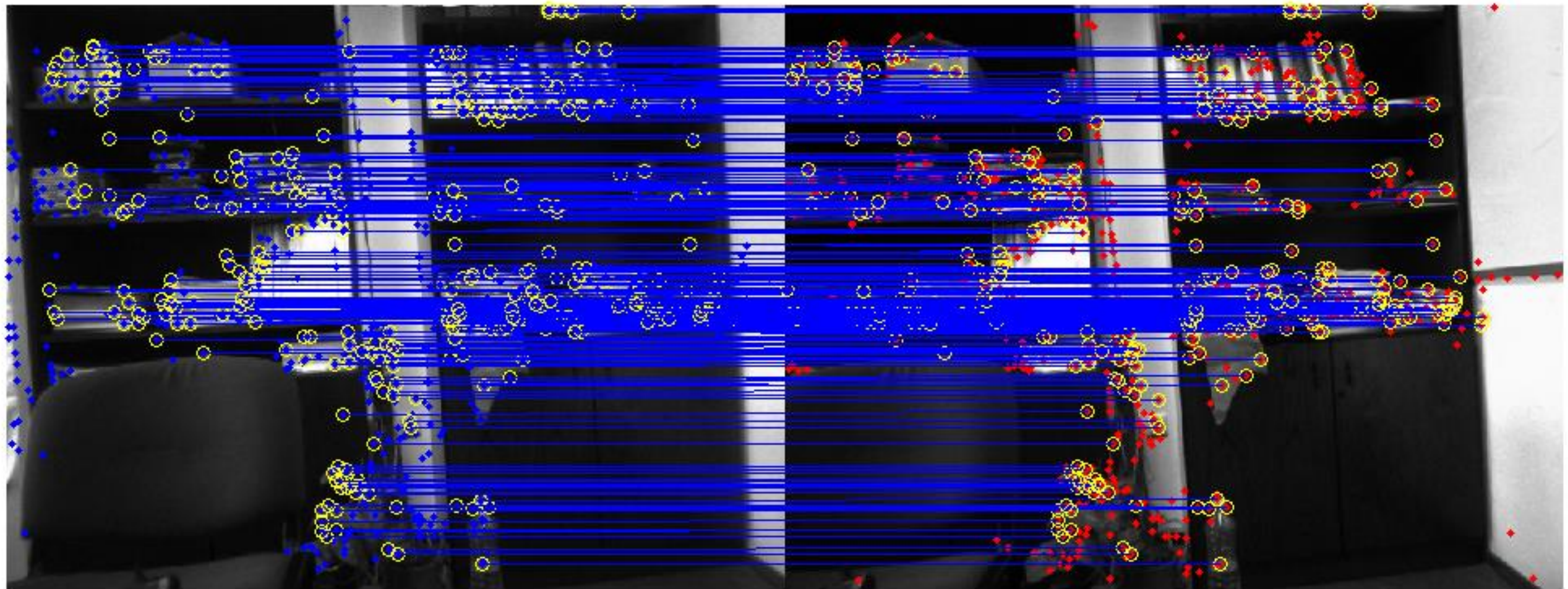
- **Scale**: Window size based on the scale at which the extreme was found
- **Orientation:** Histogram rotated along the keypoint orientation: the keypoint orientation is subtracted from each orientation of the vector

5. The SIFT operator

Example: Stereo matches from Euclidean distance

A keypoint of the let image is matched to the one in right that has the closest descriptor

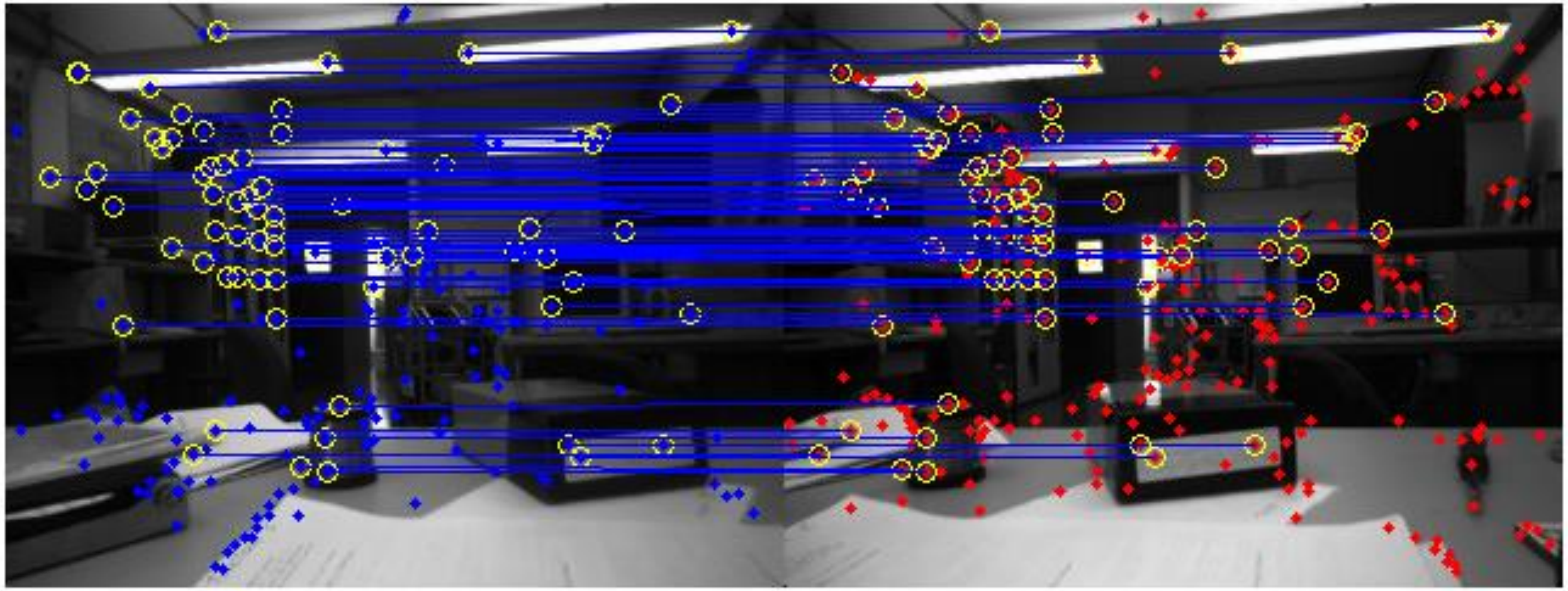
Points: 965/994 Matches: 359 37.20%



5. The SIFT operator

Example: Stereo matches

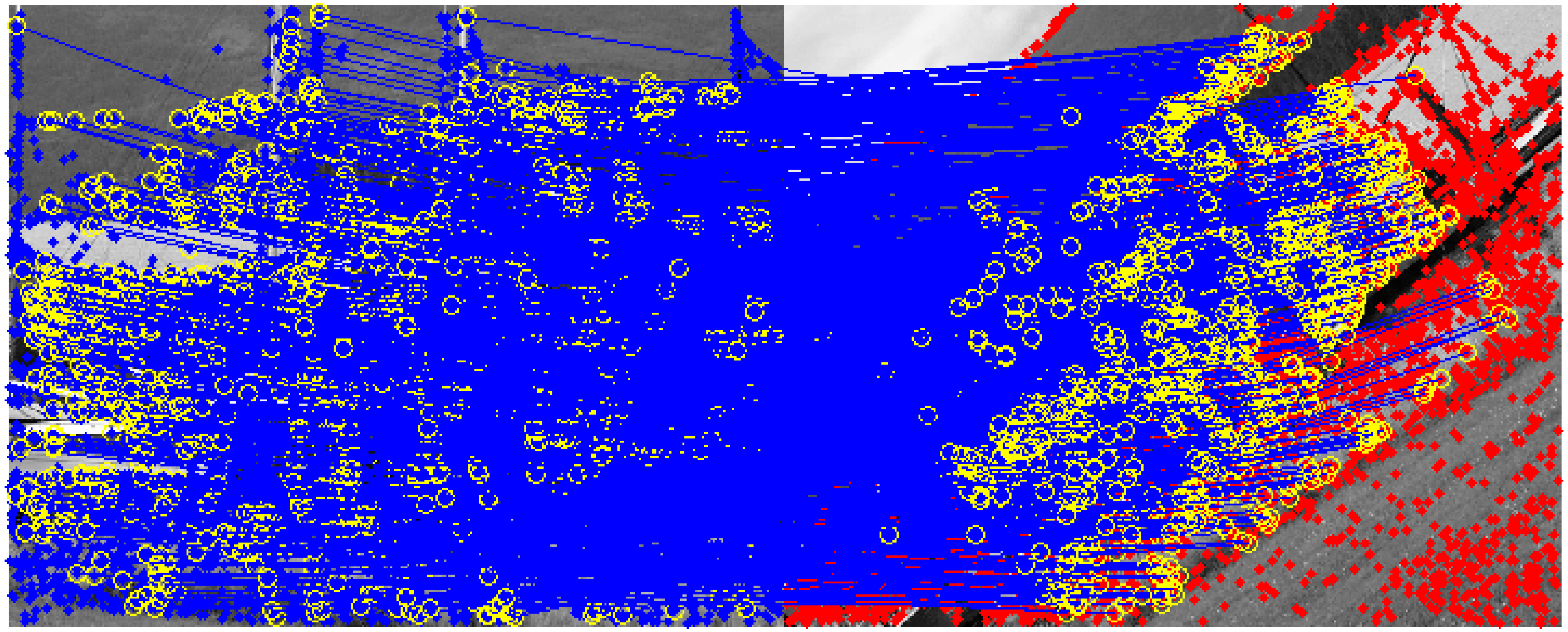
Points: 245/326 Matches: 89 36.33%



5. The SIFT operator

Example: Stereo matches (with rotation)

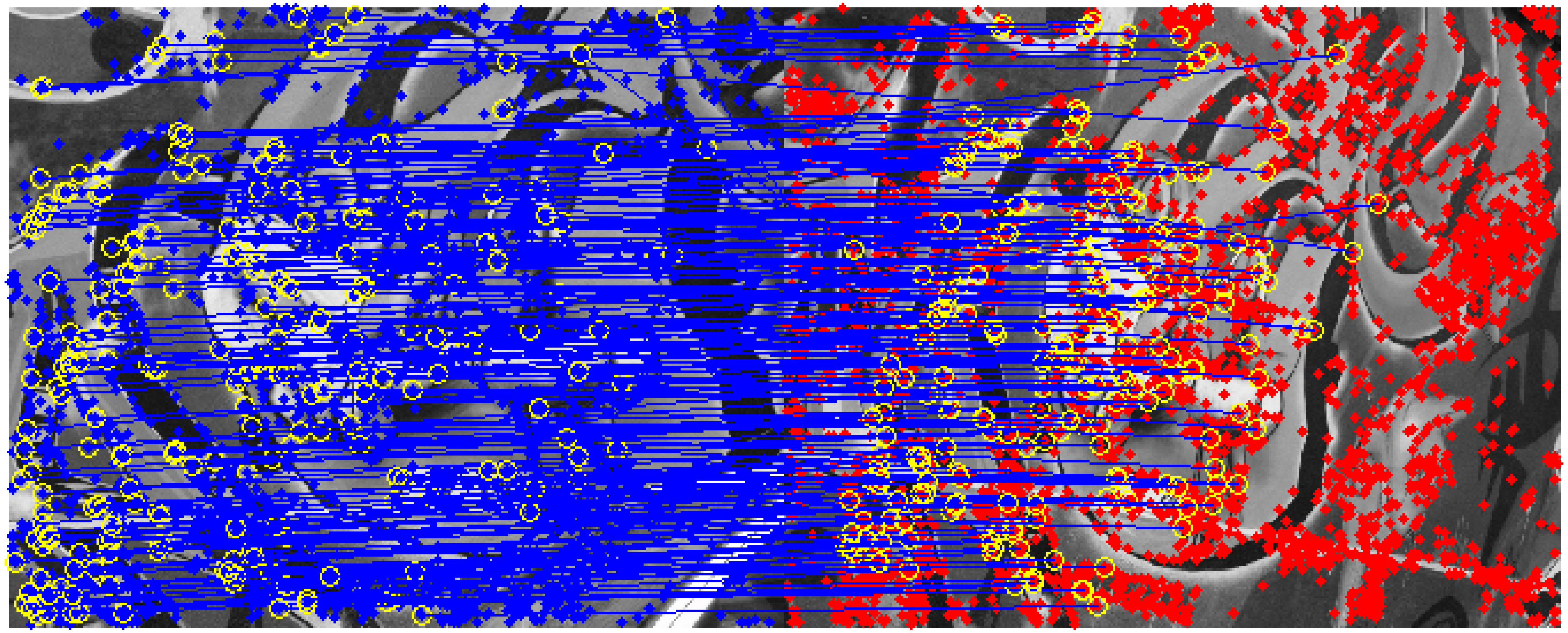
Points: 9687/7113 Matches: 1621 22.79%



5. The SIFT operator

Example: Stereo matches (with rotation)

Points: 3105/3990 Matches: 242 7,79%



Summary

- Harris operator is …
  - a corner detector which is combined with NCC for matching in other images
  - Based on first-order image derivatives
  - invariant to rotation (because derivatives along the eigenvectors)
  - NOT invariant to scale
  - Invariant to brightness (pixel intensities are not directly considered but derivatives)
  - Robust to noise (because of gaussian smoothing)
- KLT operator
  - Same idea as Harris but the two eigenvalues are used
- SIFT operator
  - Detect Blobs at different scales based on the DoG operator
  - Provides a descriptor with the information of the gradient around the detected keypoint at its scale