COMPUTATIONAL VISION: Image Features (SIFT)

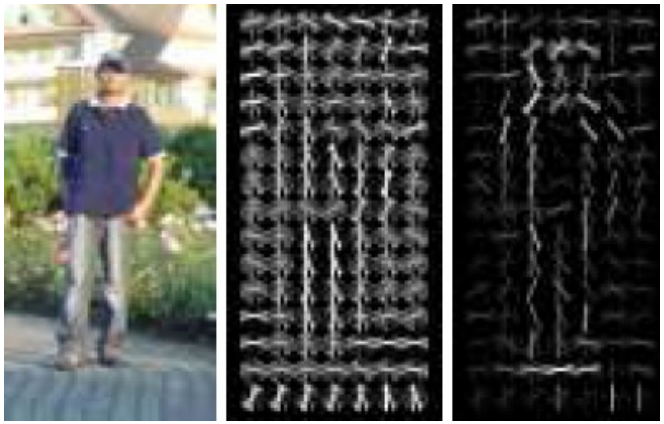
Master in Artificial Intelligence

Department of Mathematics and Computer Science

2023-2024

UNIVERSITAT DE BARCELONA

Recall: Detection based on HOG descriptors



HOG – Histogram of Gradients

Distance/similarity: d(template,image)=d(HOG(template),HOG(region))

Detection =max(d(template,image))

Disadvantages?

- Are all pixels equally important?
- Is the detection invariant to scale?
- Is the detection invariant to rotation?

How much information do we need in order to detect or recognize objects?

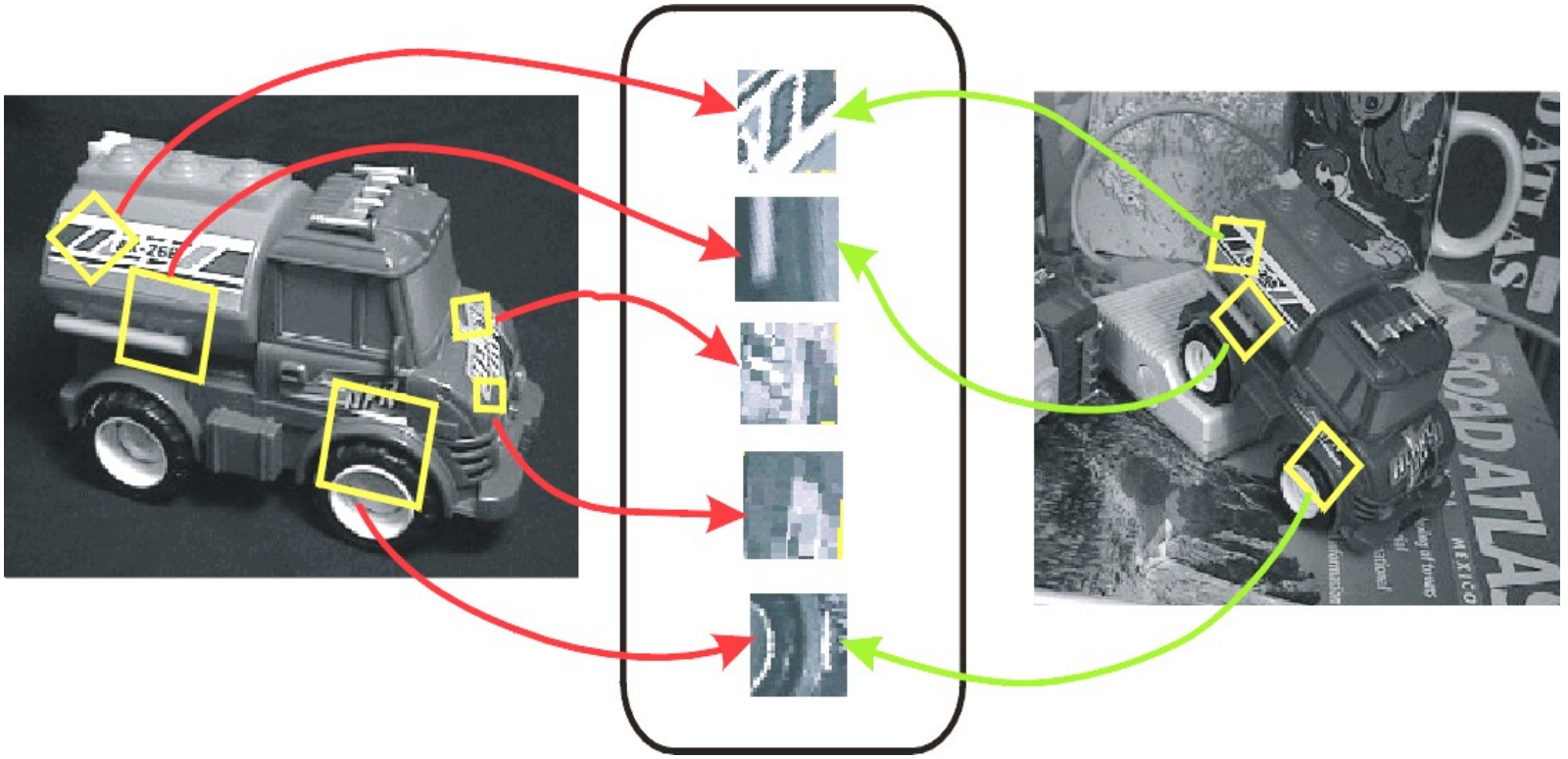
Informative vs. non-informative regions for object recognition

Local Features

To build a **panorama** image of the mountain from these two views of the mountain. What kind of local features might one use to establish a **matching** between the images?

Local Features

What kind of local features can be extracted from the left images to encounter the object on a new image?



Applications of local features

- Panoramic image creation (mosaicing)
- Camera pose estimation
- Object recognition
- Object tracking

Local features based on the appearance of the object at particular interest points.

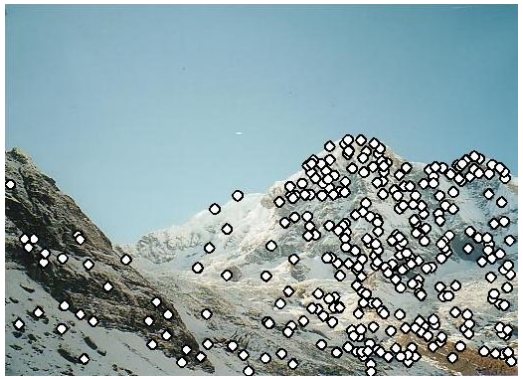
Outline

- Local invariant features

1. Detection of interest points
   - Harris corner detection and SIFT detector
   - Scale invariance
2. Description of local patches
   - SIFT : Histograms of oriented gradients
   - ORB feature descriptor
3. Matching

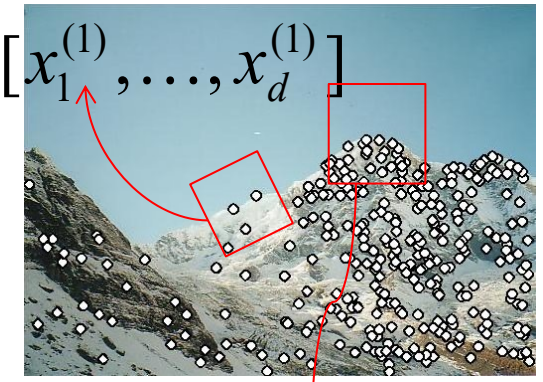
Local features: main components

1) **Detection**: Identify the interest points in the image.



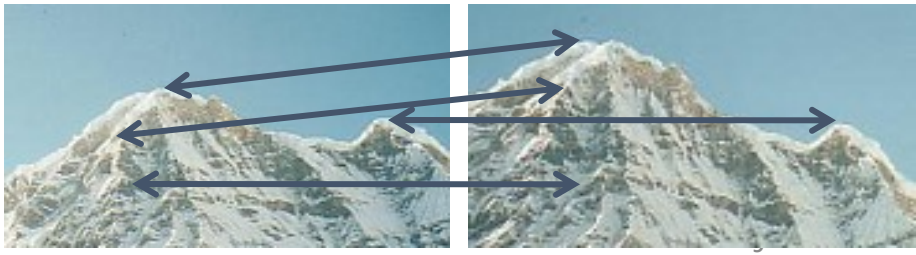
2) **Description**: Estimate vector feature descriptor of the interest patch (surrounding each interest point).

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



3) **Matching**: Determine correspondence between descriptors in two views

$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$



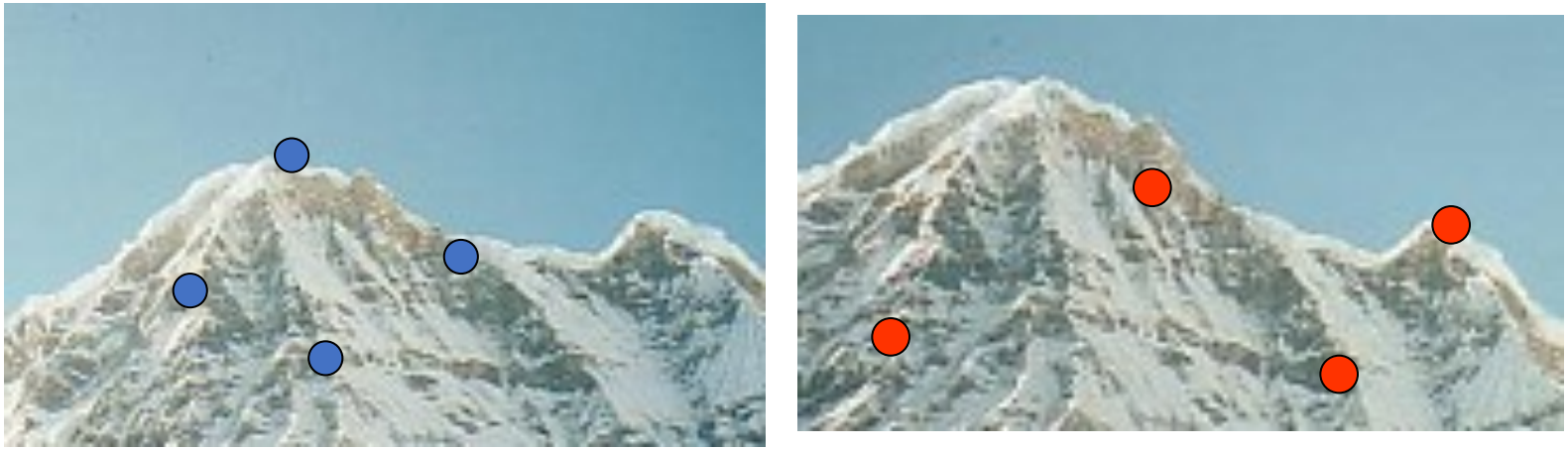
Local features: desired properties

- **Repeatability/Precision**
  - The same feature can be found in several images despite geometric and photometric transformations
- **Saliency**
  - Each feature has a distinctive description
- **Compactness and efficiency**
  - A few features vs. all image pixels

- **Locality**
  - A feature occupies a relatively small area of the image, robust to clutter and occlusion.

Goal: interest operator repeatability

- We want to detect (at least some of) the same points in both images.

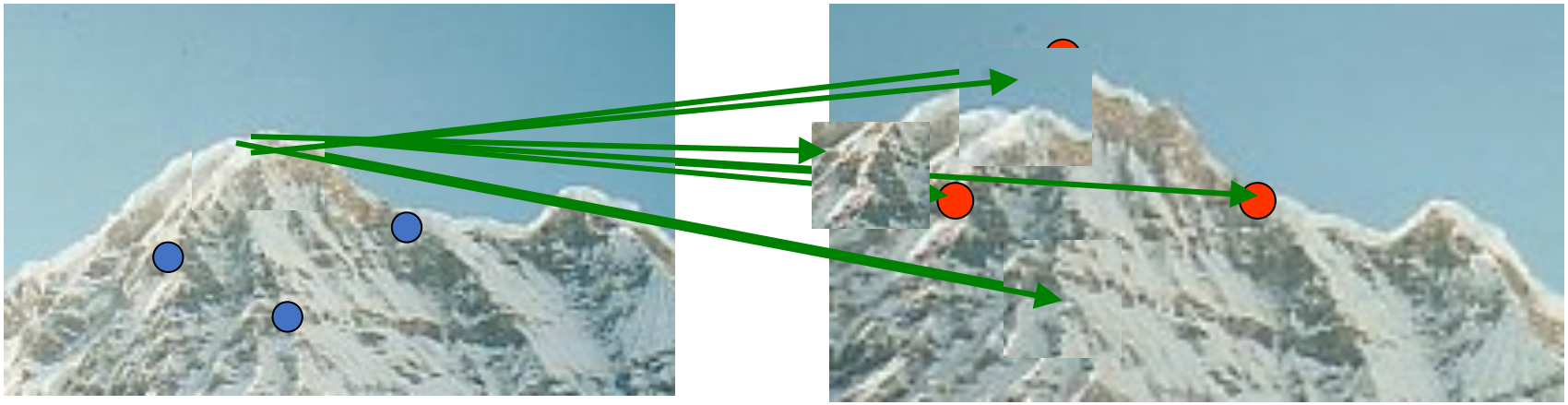


With this set of points: No chance to find true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

Goal: descriptor distinctiveness

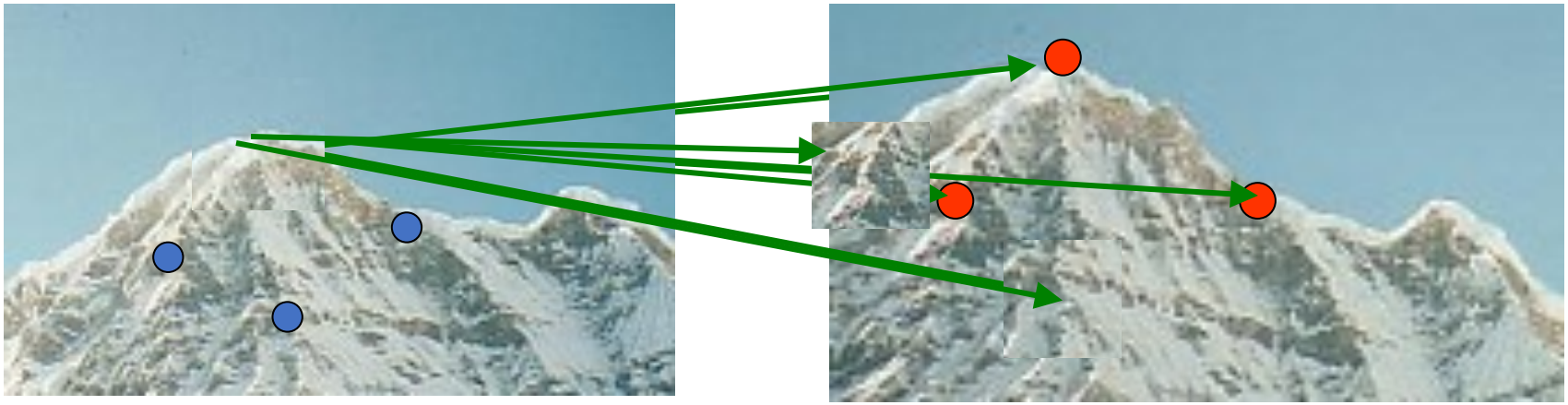
- Distinctive keypoints to find them easily in the new image (a transformed version of the first).
  - We want to be able to reliably determine which point goes with which.



- Must provide some invariance to geometric and photometric differences between the two views.

Goal: compactness, efficiency and locality

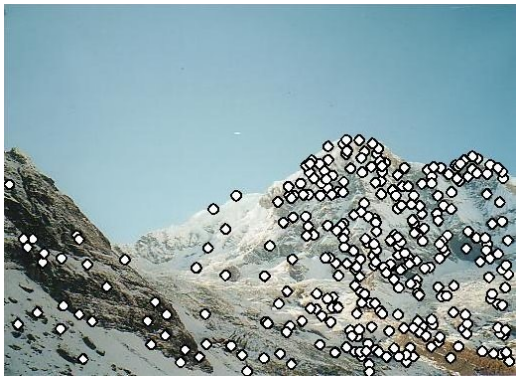
- We want to extract the minimum information that allows us to put into correspondence.



- Which should it be?

Local features: main components

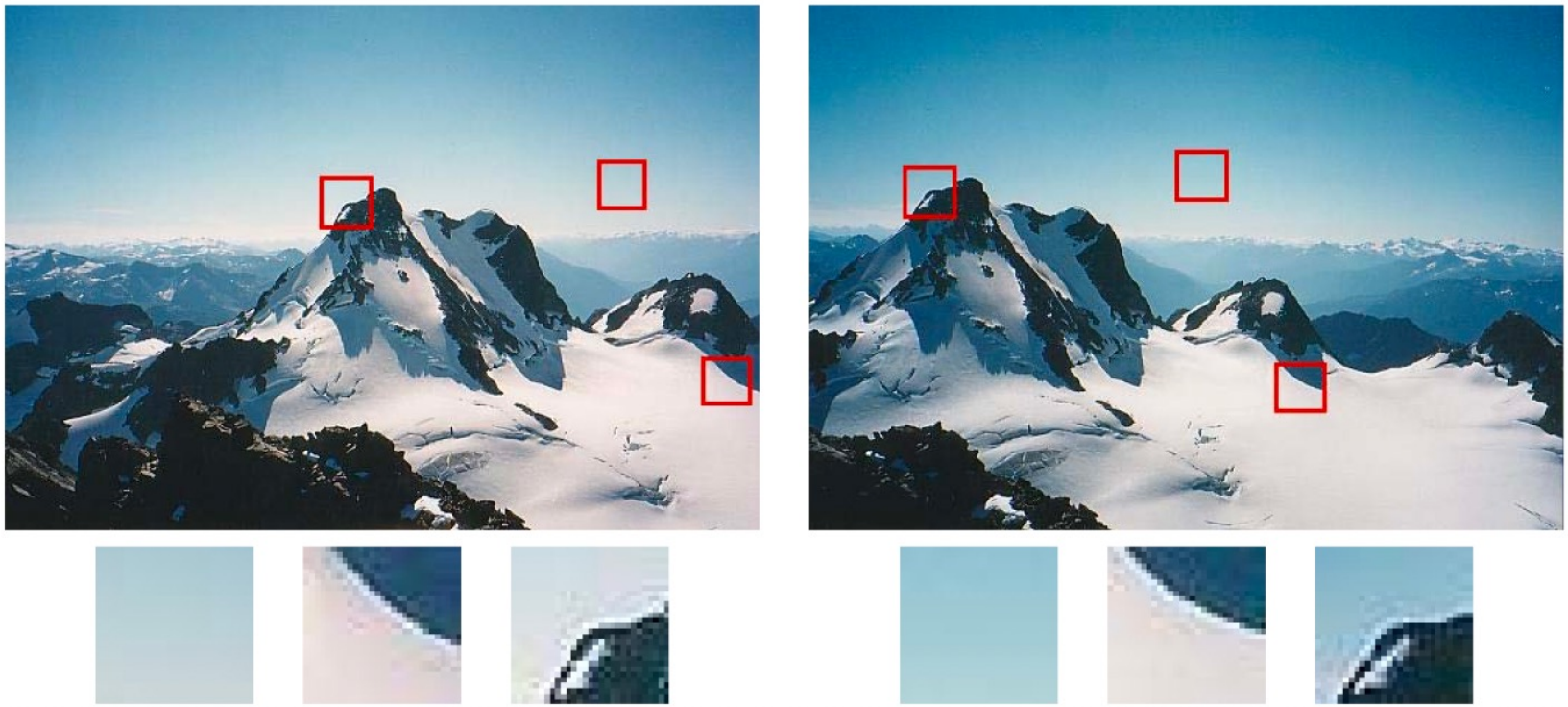
1) **Detection**: Identify the interest points



2) Description: Extract vector feature descriptor surrounding each interest point.

3) Matching: Determine correspondence between descriptors in two views

Feature Detection



Which are good patches for candidates to describe the scene?

Distinctive interest points

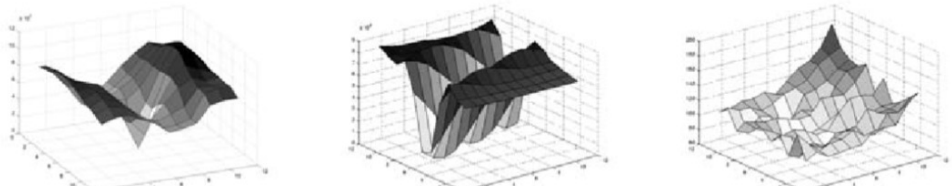
- Let us compare the patch R with the patches around it
- Autocorrelation function:

$$\sum_{(x,y)\in R} \left(I(x+d_x, y+d_y) - I(x,y)\right)^2$$

Example of autocorrelation applied to three locations:

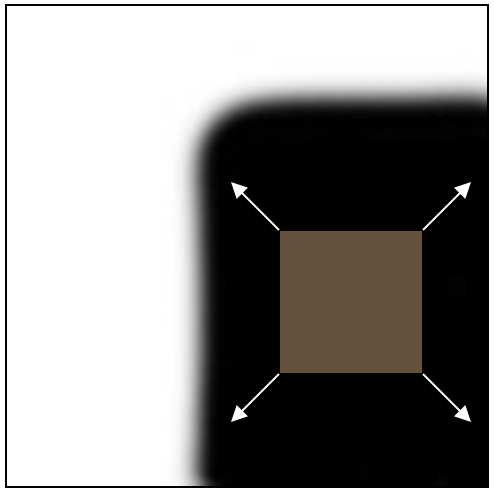
To which point does each autocorrelation surface correspond?

(a) Flower (unique minimum, a stable corner)
(b) Roof edge (aperture problem)
(c) Cloud (texture-less region)

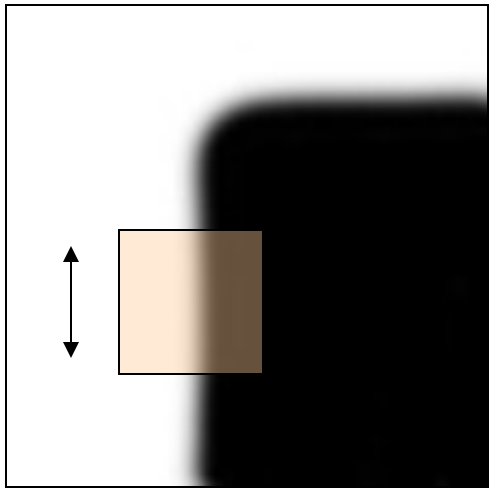


Distinctive interest points

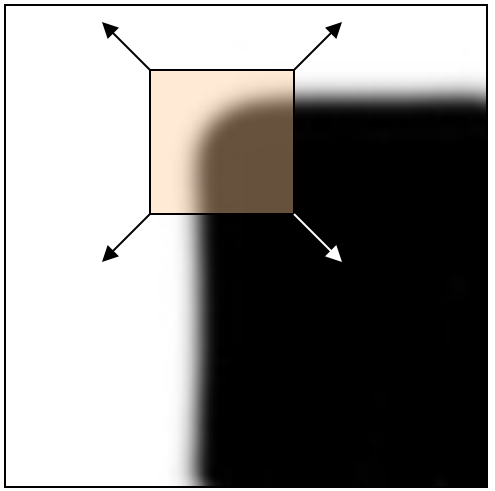
- We should easily recognize the point by looking through a small window
- Shifting the window in any direction should give a large change in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction (aperture problem)



"corner":
significant change in all directions

(patch matches multiple positions)

(patch matches one position)

Definition: Corners are distinctive image interest points

Slide credit: Alyosha Efros, Darya Frolova, Denis Simakov

Distinctive interest points

- How can we easily know, for a given patch R, if it is stable or not?
- We need to compute the shape of the autocorrelation function for each possible patch.
  - This is very time consuming!
- Isn't there any other way to tackle the problem?
  - Maths are here to help!
  - We are going to **linearize** the previous function using Taylor series.

Distinctive interest points

The autocorrelaction function is approximated as $\boldsymbol{d}^T \boldsymbol{M} \boldsymbol{d}$ where $\boldsymbol{d} = (d_x, d_y)$, and $\boldsymbol{M}$ is a 2x2 matrix of image derivatives (averaged in pixel neighborhood).

$$M = \sum_{\text{Neighborhood of (x,y)}} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

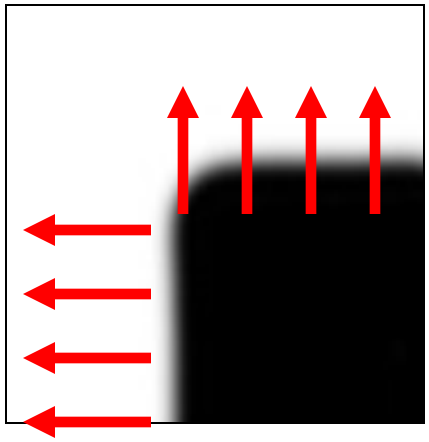
M can be easily computed for each patch of the image.

Notation:

$$I_x \Leftrightarrow \frac{\partial I}{\partial x} \quad I_y \Leftrightarrow \frac{\partial I}{\partial y} \quad I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

What does this matrix reveal?

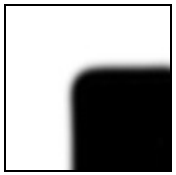
First, consider an axis-aligned corner:



Can we infer the shape of the autocorrelation function using the matrix *M*?

What does this matrix reveal?

First, consider an axis-aligned corner:



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

- This means that dominant gradient directions align with x or y axis. For each pixel of the image, we have or $I_x = 0$ or $I_y = 0$ or both = 0 --> The matrix *M* is diagonal.
- If either λ is close to 0, then this is **not** corner-like.
- Look for locations where **both** λ's are large.

What does this matrix reveal?

What if we have a corner that is not aligned with the image axes?

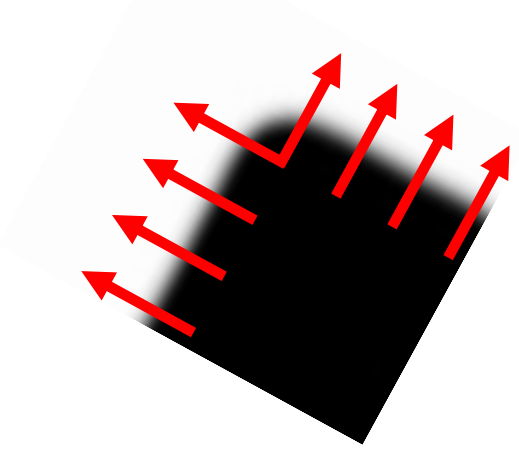
A little bit of Math:

Given a symmetric matrix ***M***, it can be decomposed:

$$M = E\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}E^T \quad Me_i = \lambda_i e_i, E = [e_1, e_2]$$

λ_1 and λ_2 are called **eigenvalues**.

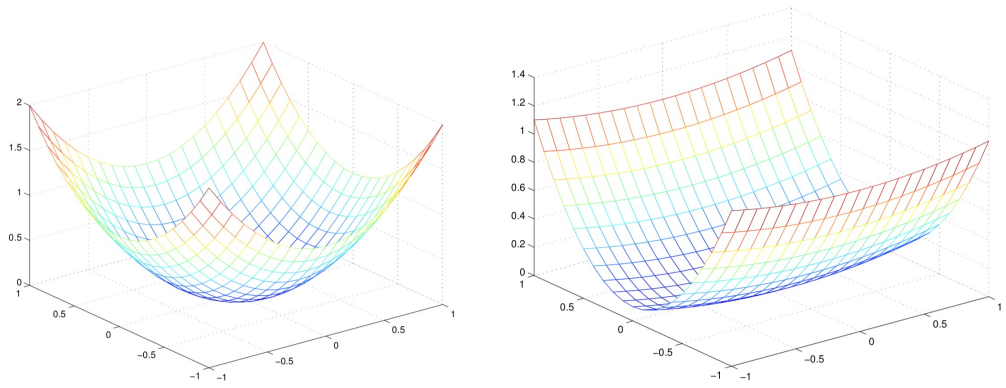
Vectors e_1 and e_2 are called **eigenvectors**.



Corners-distinctive interest points

Since *M* is symmetric, we have

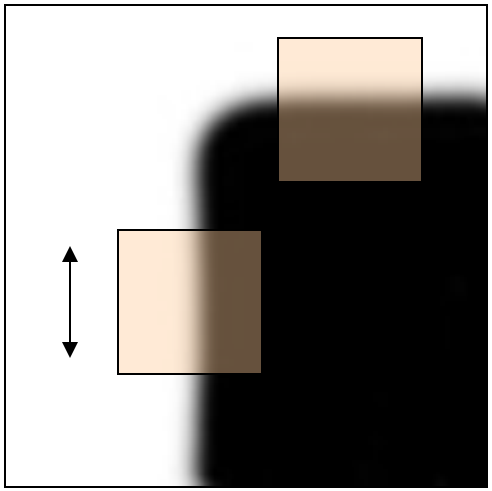
$$M = E\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} E^T$$

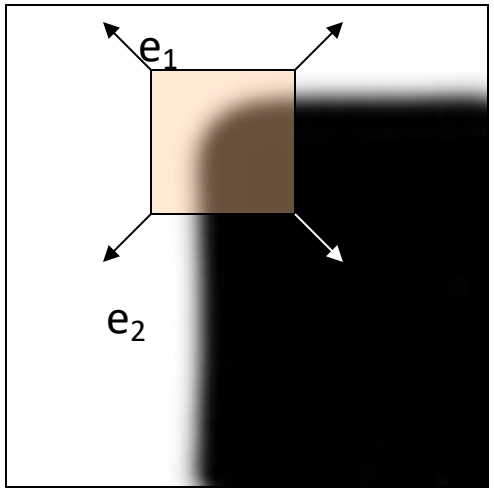


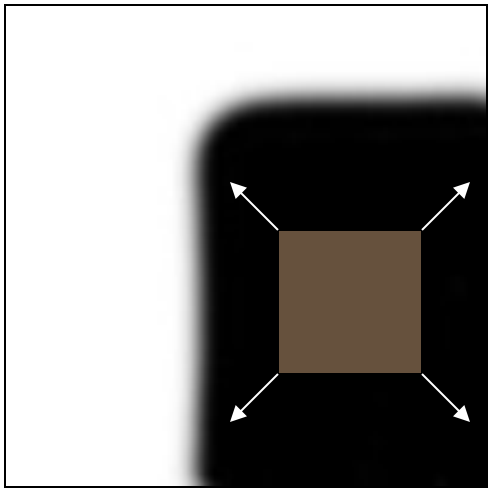
$$Me_i = \lambda_i e_i, E = [e_1, e_2]$$

- The *eigenvalues* of ***M*** *(*λ_1 and λ_2) reveal the **amount of intensity change** in the two principal orthogonal gradient directions in the window.
- The *eigenvectors* of M (e_1 and e_2) give the **direction of maximum and minimum change** in the image.
- E → rotational matrix

Corners as distinctive interest points







"edge":

$|\lambda_1| >> |\lambda_2|$
or $|\lambda_2| >> |\lambda_1|$

"corner":

$|\lambda_1|$ and $|\lambda_2|$ are large,
$\lambda_1 \sim \lambda_2$;

"flat" region

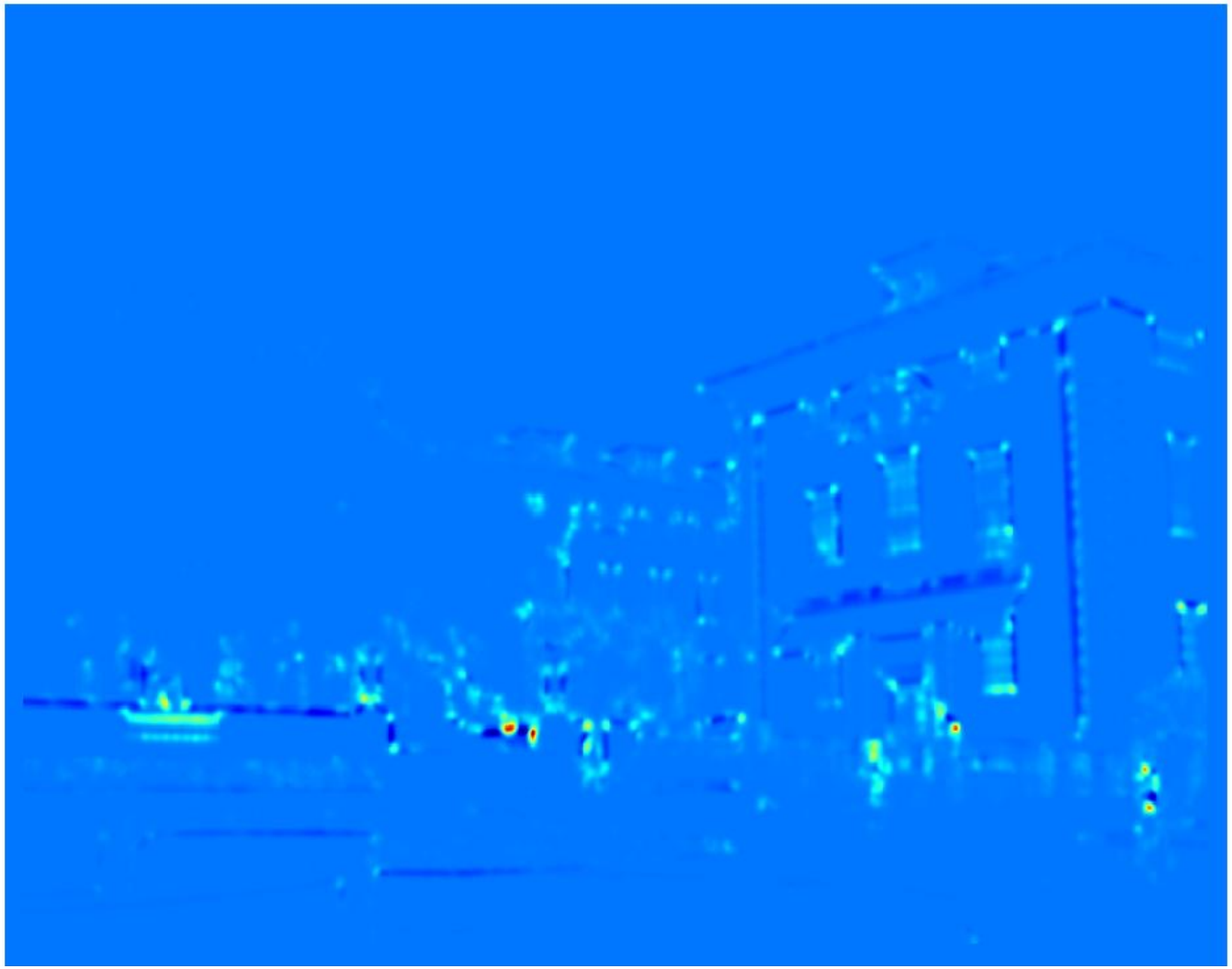
$|\lambda_1|$ and $|\lambda_2|$ are small;

Harris corners: 'cornerness' score: $f = \dfrac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$

What points would you choose?

Example of Harris application

- Compute corner response at every pixel.



Example of Harris application

Kristen Grauman

Harris corner detector: Algorithm

1. Compute Gaussian derivatives at each pixel
2. For each pixel of the image, **compute matrix *M*** in a window

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

$$Me_i = \lambda_i e_i, E = [e_1, e_2]$$

where λ_i are the eigenvalues and e_i are the eigenvectors of ***M***

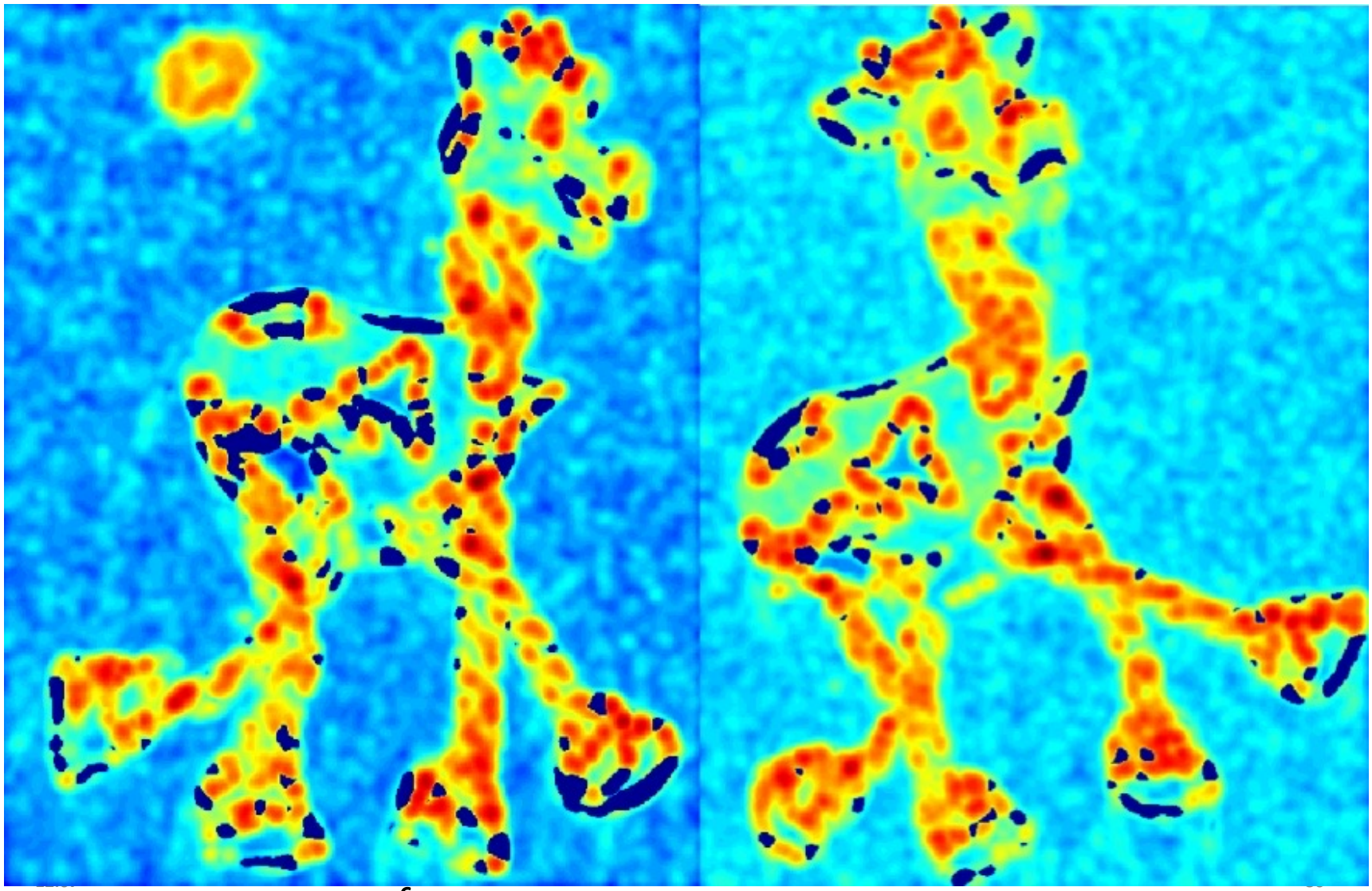
3. Compute the cornerness score: $f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$

4. Find **points with large corner response** (f > threshold).
5. Take the points of **local maxima for each window**, i.e. perform non-maximum suppression in the neighborhood.

*Chris Harris and Mike Stephens (1988). "A Combined Corner and Edge Detector". Alvey Vision Conference. **15**.*

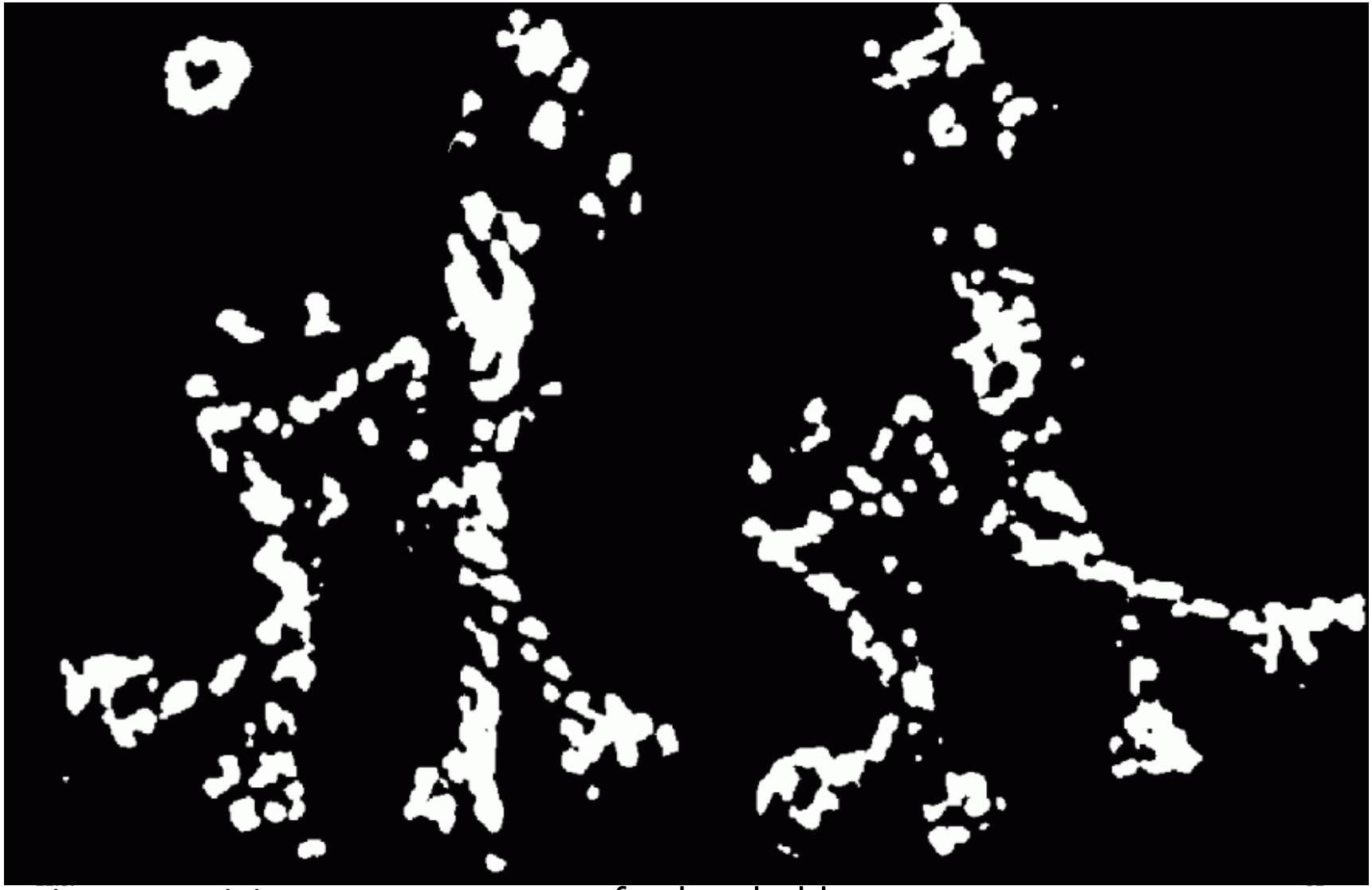
Harris Detector: Steps

Harris Detector: Cornesness evaluation



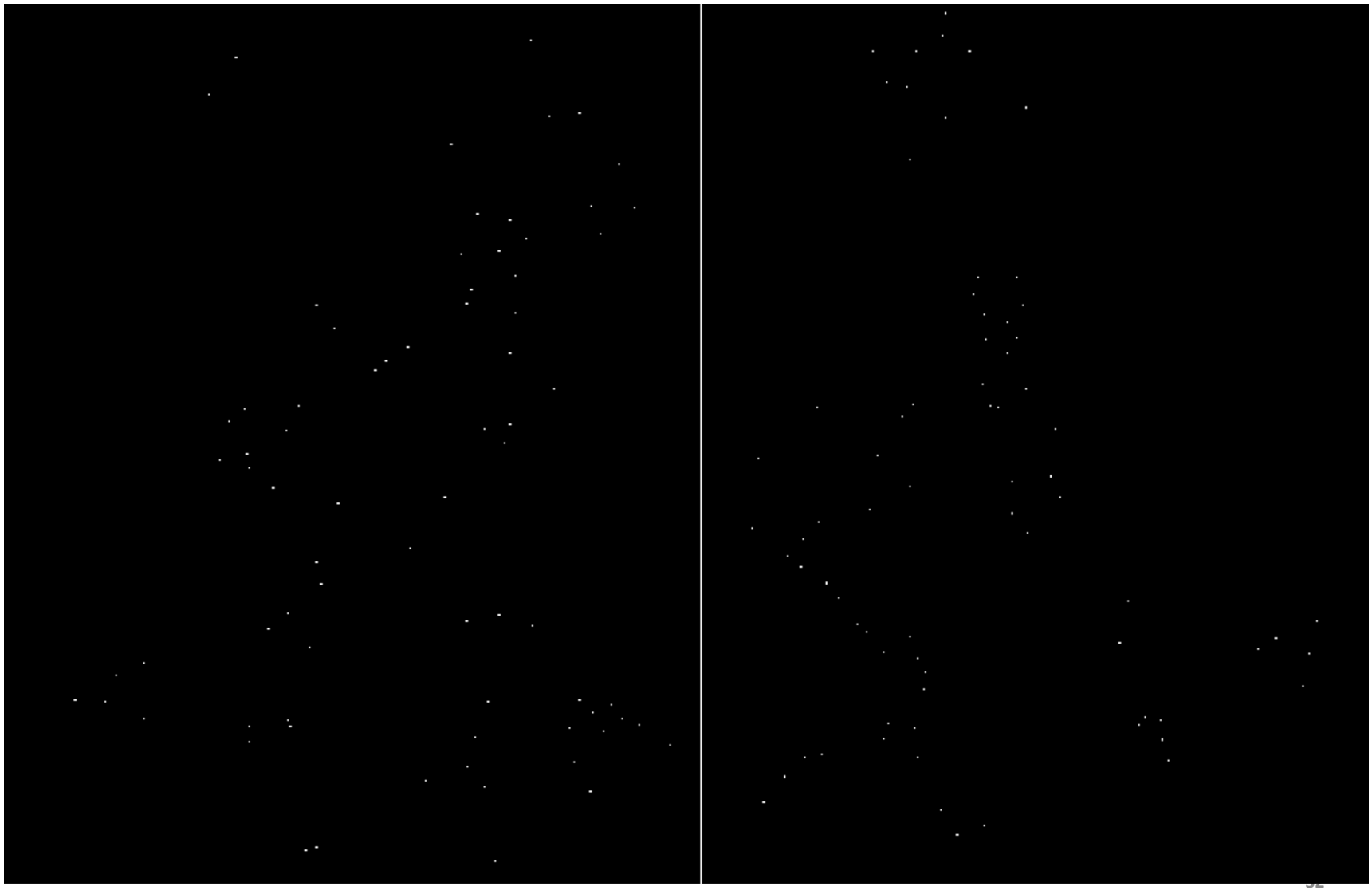
Compute corner response f

Harris Detector: Steps



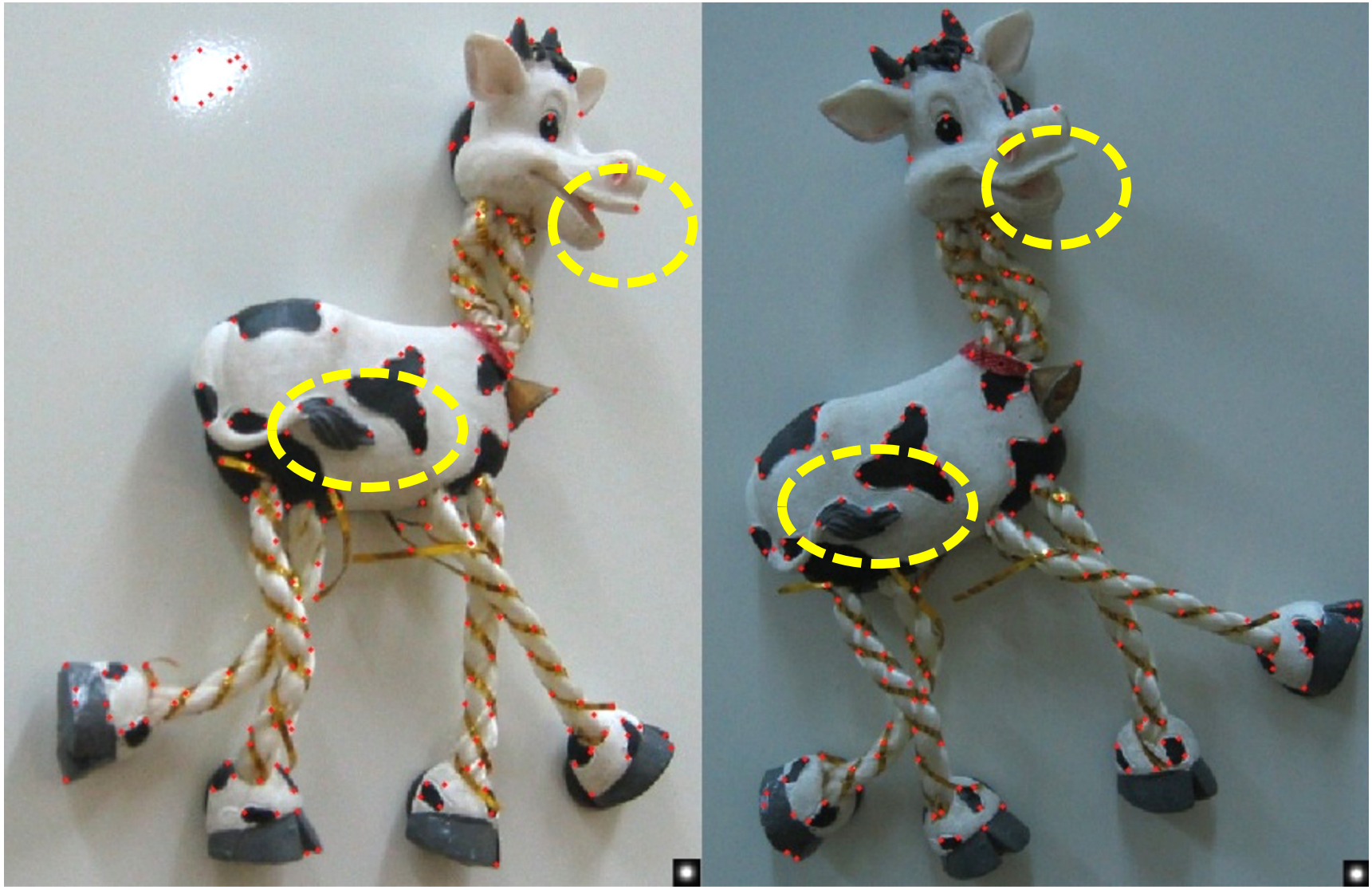
Find points with large corner response: $f >$ threshold

Harris Detector: Steps



Non-maxima suppression: Take only the points of local maxima of *f* in each window

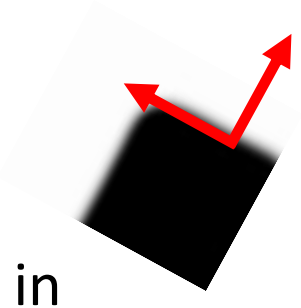
Harris Detector: Steps



Properties of the Harris corner detector

- Rotation invariant?

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



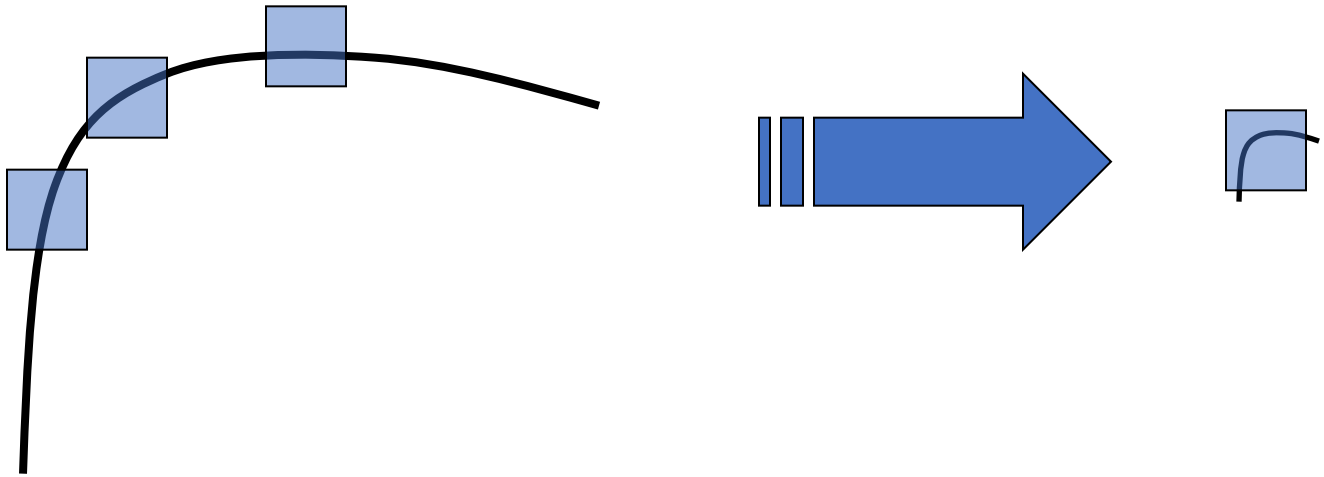
Yes, X rotates the matrix so that the maximal change is in the direction of e_1 (if $|\lambda_1| > |\lambda_2|$) or e_2.

Representation of SIFT detector

Properties of the Harris corner detector

- Scale invariant?

No



All points will be classified as edges

Corner !

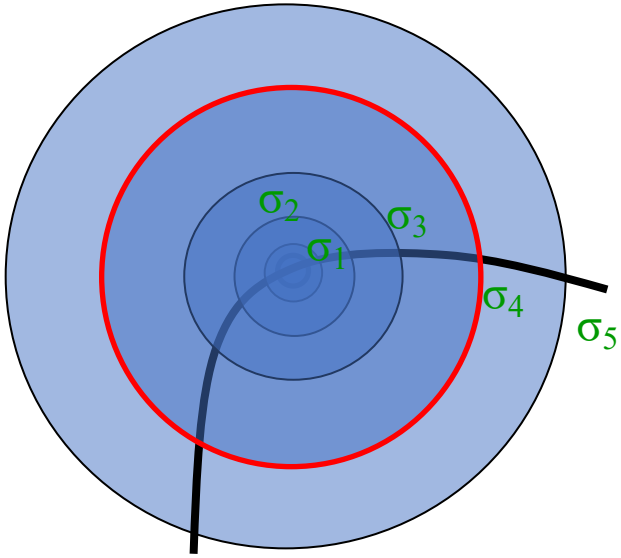
Scale invariant interest points

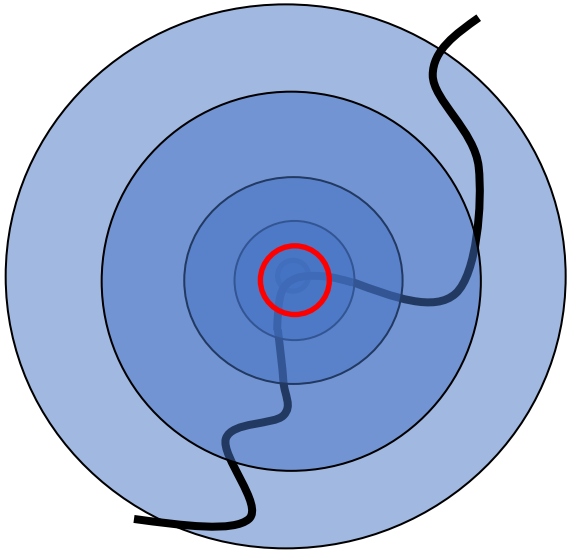
How can we independently select interest points in each image, such that the detections are repeatable across different scales?

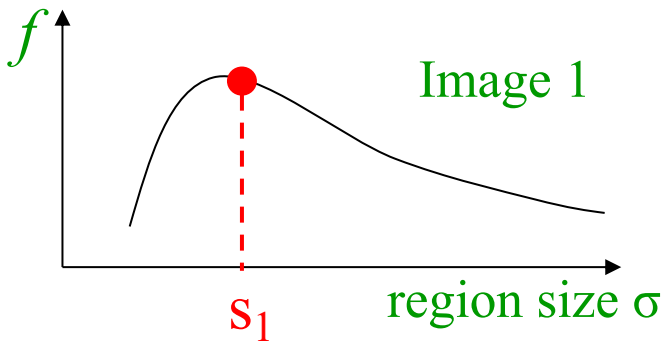
Automatic scale selection

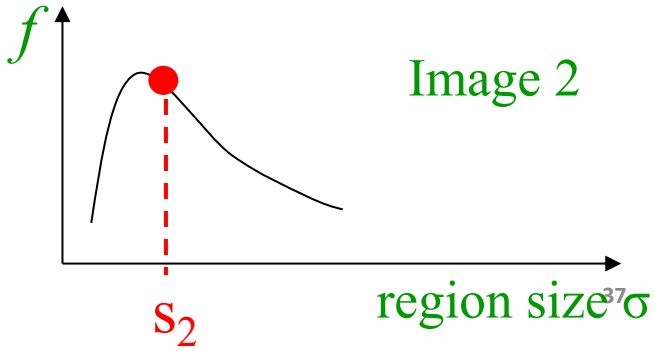
Intuition:

- Compute the derivatives (I_x,I_y) with different $\boldsymbol{\sigma}$, and find the scale that gives local maxima of the *cornerness* function *f* in both position and scale.

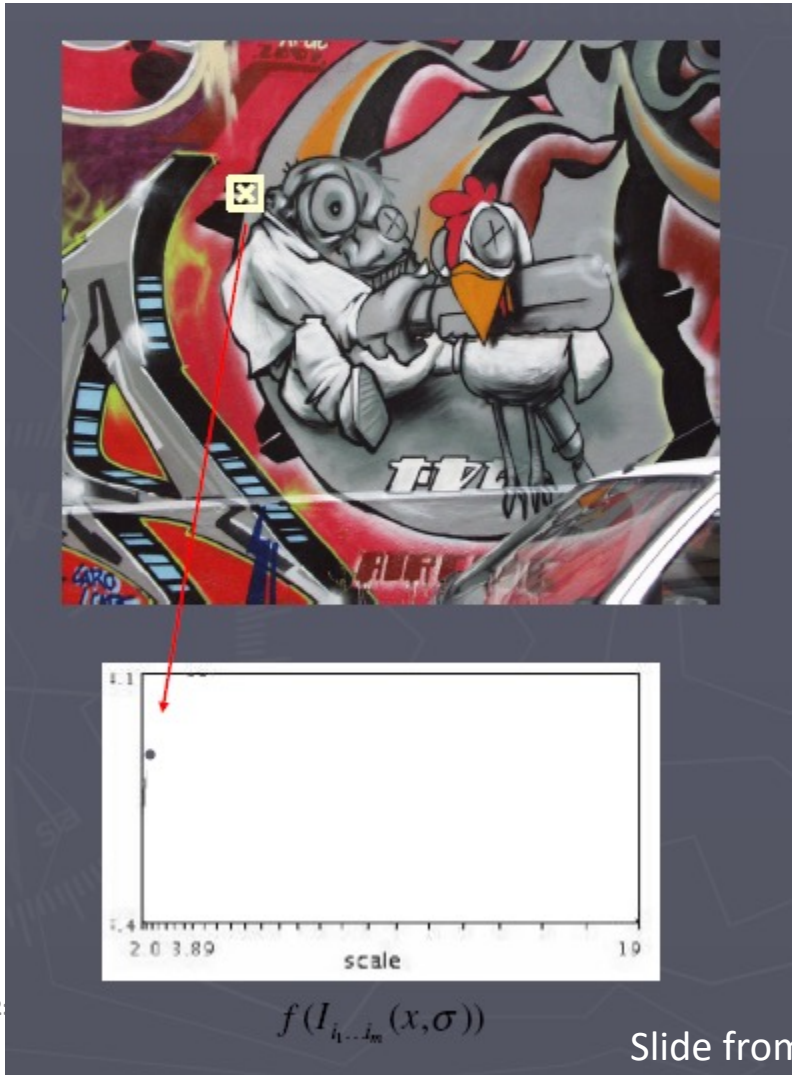






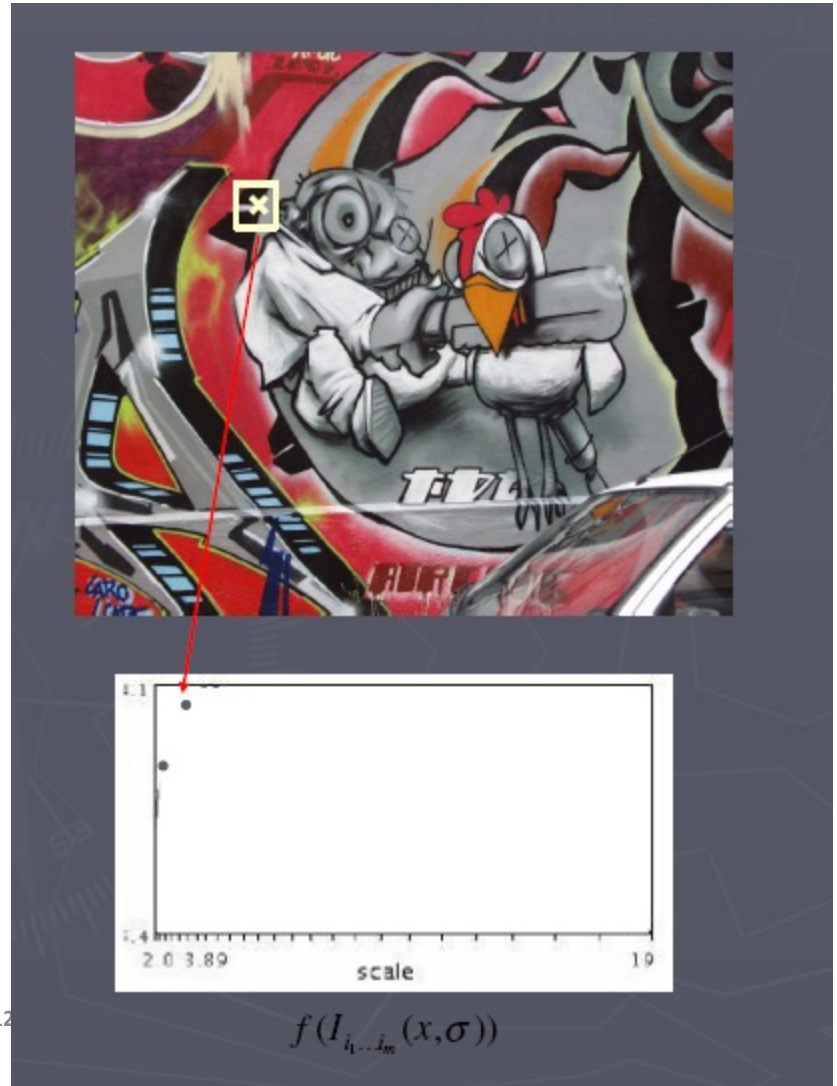


Automatic scale detection

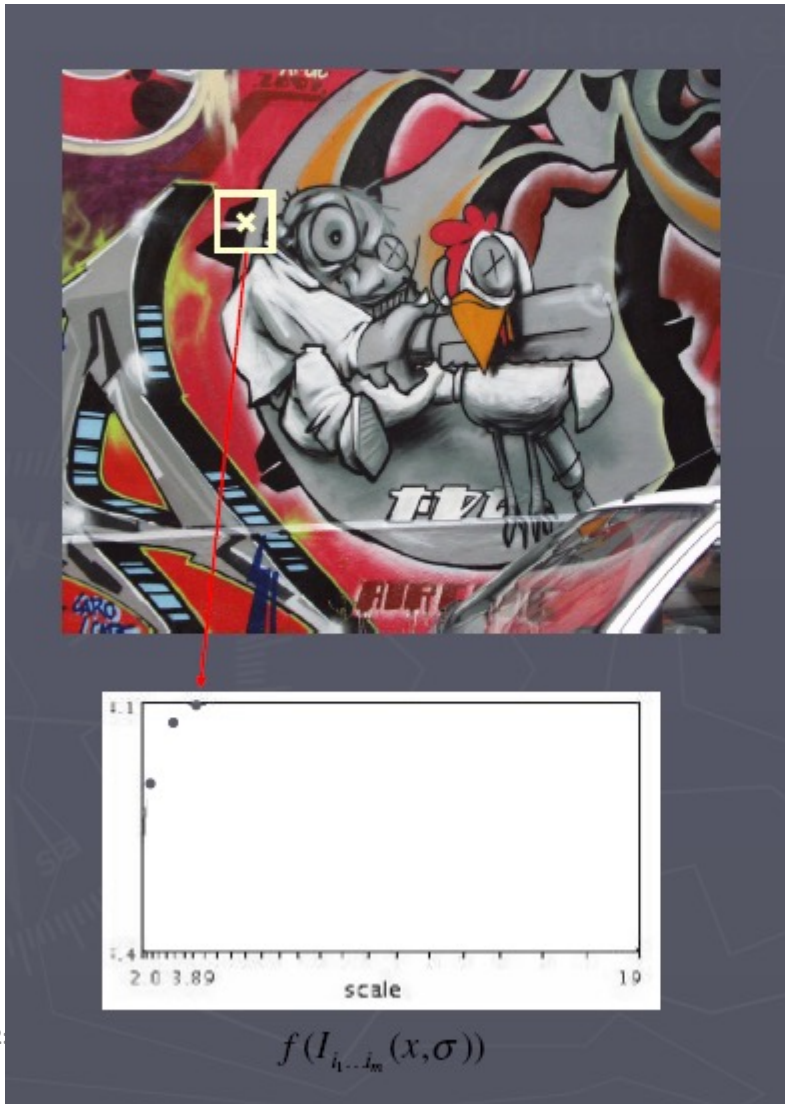


Slide from

Automatic scale detection



Automatic scale detection

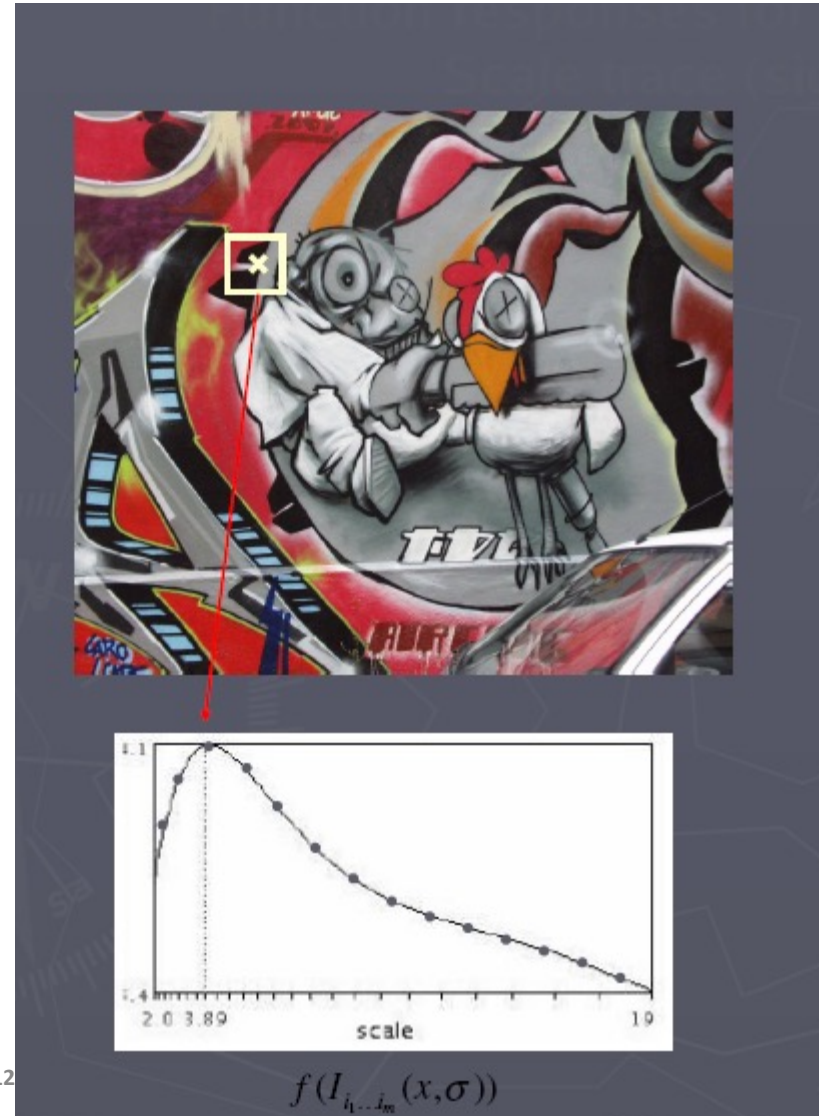


Automatic scale detection

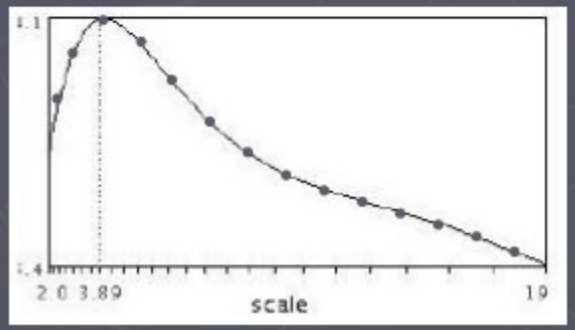
Automatic scale detection

$$f(I_{i_1 \ldots i_m}(x,\sigma))$$

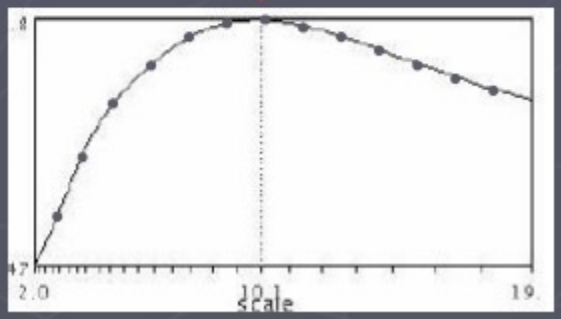
Automatic scale detection



Automatic scale detection

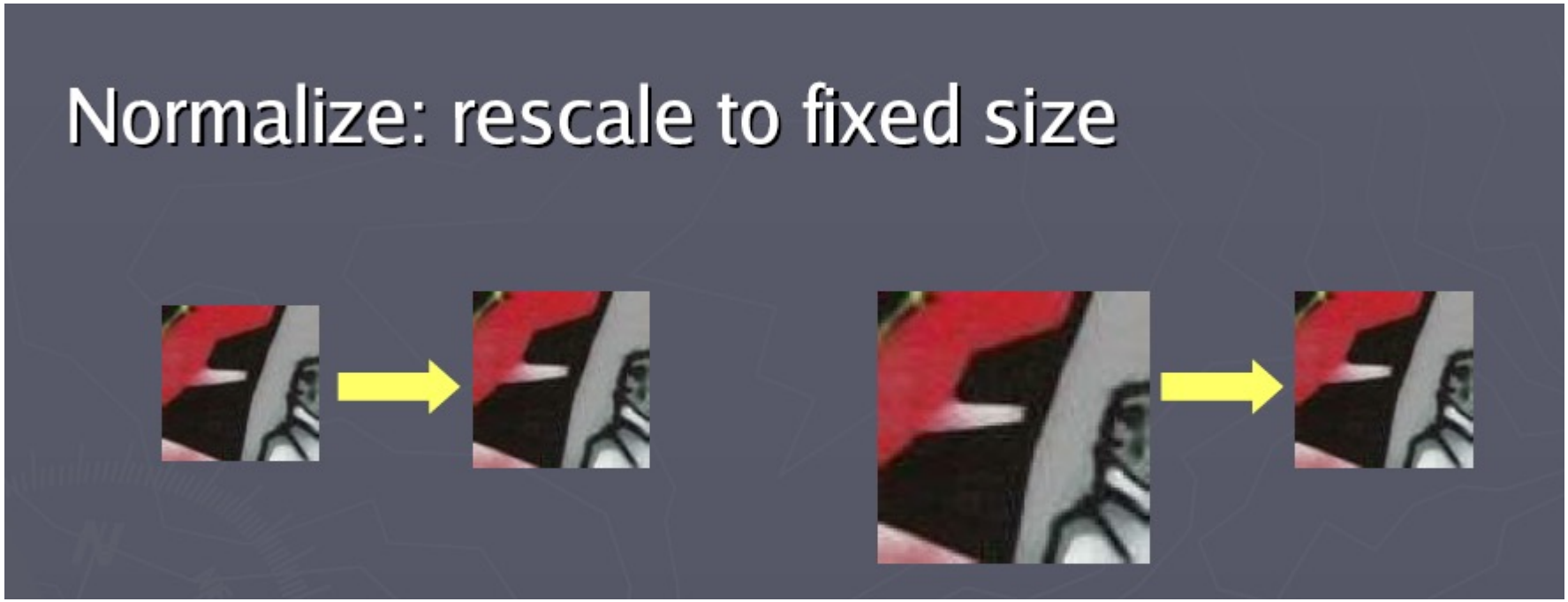


$f(I_{i_1 \ldots i_m}(x,\sigma))$



$f(I_{i_1 \ldots i_m}(x',\sigma'))$

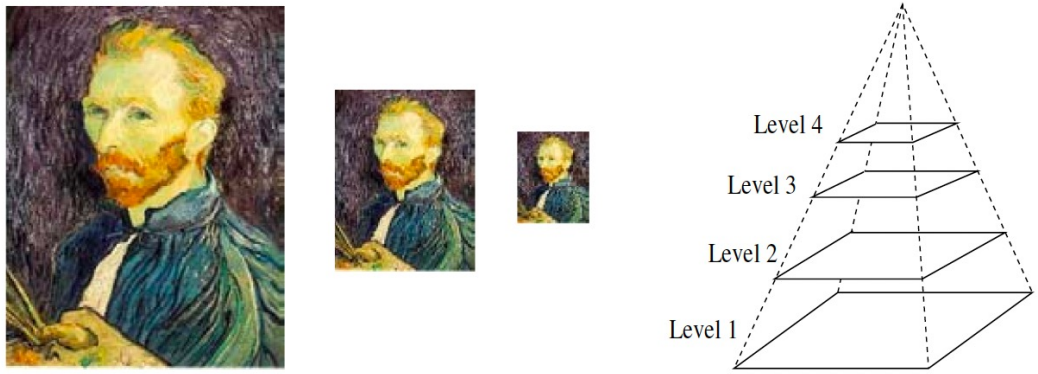
Automatic scale detection



Patch Reescaled patch Patch Reescaled patch

SIFT Detector

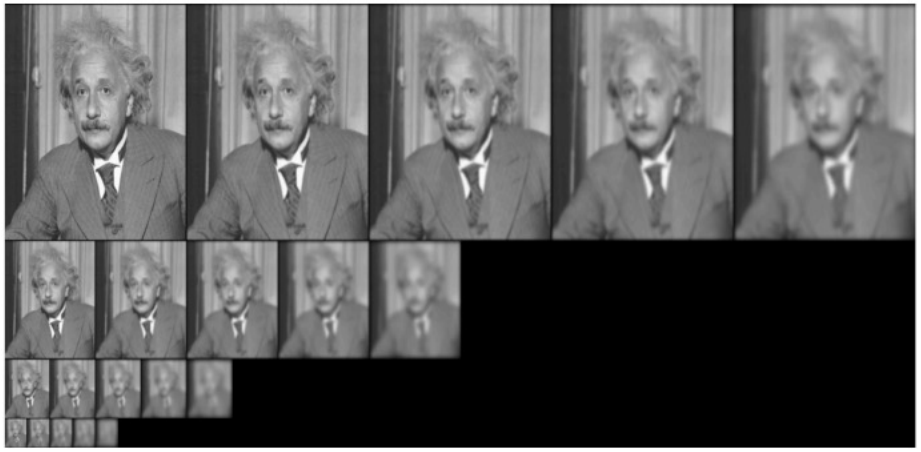
Instead of increasing the size of the patch R for a given image, one may **reduce the size of the image and fix the size of R through the scales**. This technique builds a **pyramid of scales**.



- Level 1 corresponds to original image, level 2 to subsampled image, and so on.
- For each scale significant patches with size R are extracted.
- This technique reduces computational effort (with respect to increasing the size of R with a fixed image).

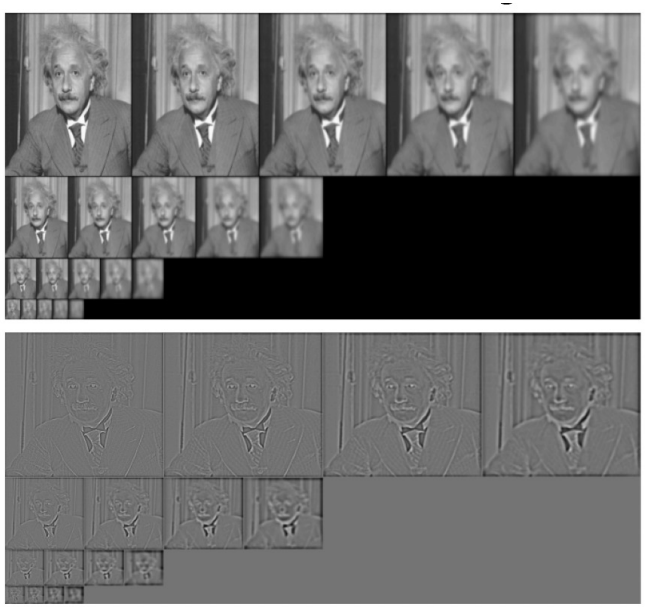
SIFT Detector

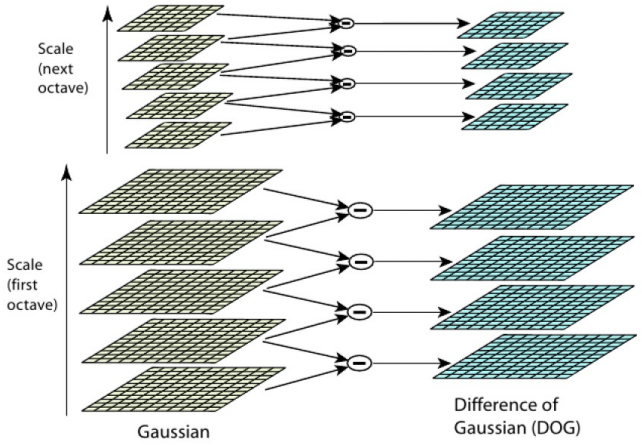
- Select some candidate keypoints using scale-space extrema detection.
- Instead of checking each possible patch R for each possible scale, we select some interest candidate points. For that issue, images are first convolved at each scale with Gaussian filters.



SIFT Detector

- The **Difference-of-Gaussian** images are computed from the difference of blurred images.





Candidate keypoints are identified as local maxima or minima of the Difference-of-Gaussian images across scale.

SIFT Detector

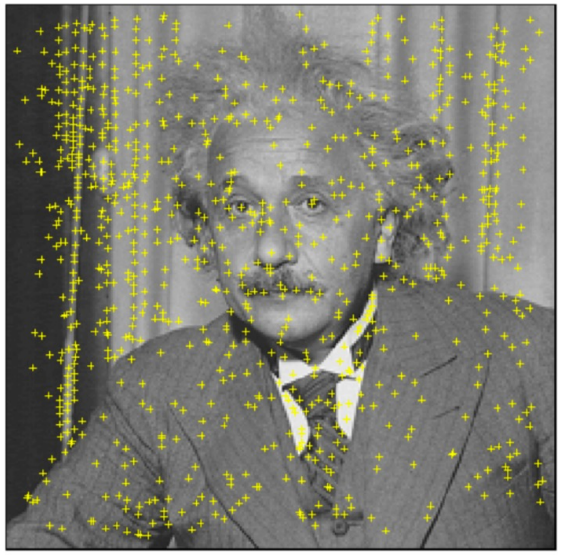
- Select some candidate keypoints using **scale-space extrema** detection.
- The candidate keypoints are tested for significance by a two-step procedure:

1. Test if the absolute value of the **Difference-of-Gaussian** image D at the minimum or maximum $x\hat{}$ is greater than a threshold:

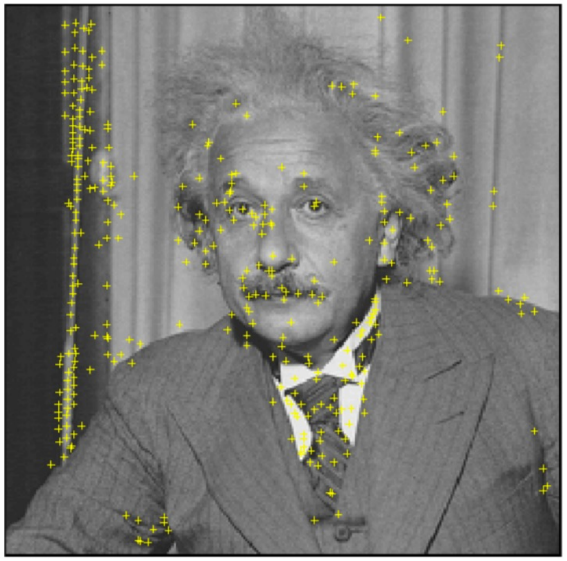
   $|D(x\hat{})| > \gamma$

   This allows to reduce sensitivity to noise.
2. Test if the autocorrelation of the Difference-of-Gaussian images D at $x\hat{}$ indicates significance. This step allows to remove badly localized candidates (i.e. at a straight edge).

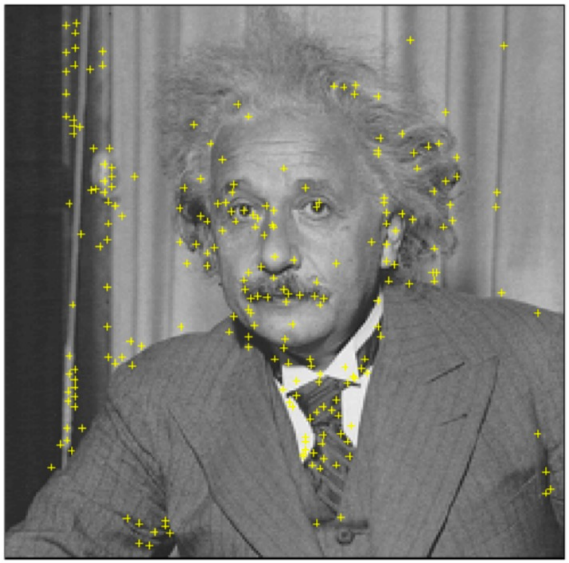
SIFT Detector



Candidate points



Points with $|D(\hat{x})| > \gamma$



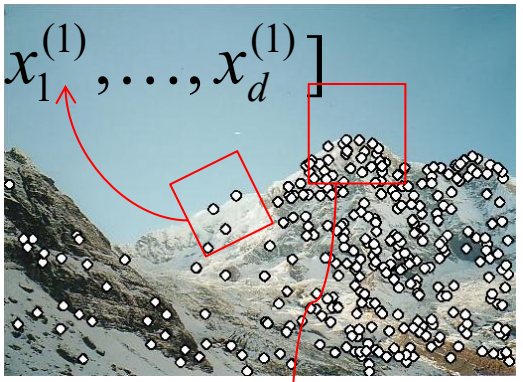
Significant points

Local features: main components

1) Detection: Identify the interest points

2) **Description**: Extract feature descriptor vector surrounding each interest point.

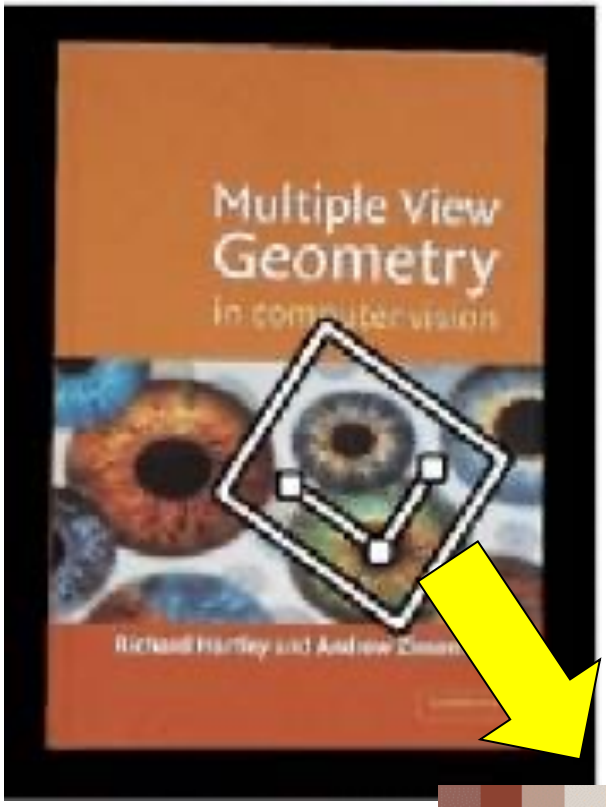
$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$

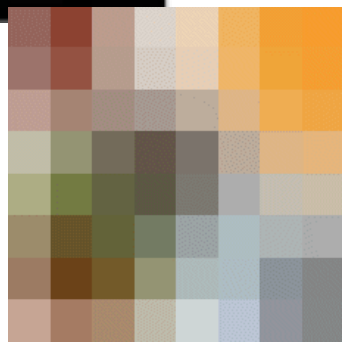


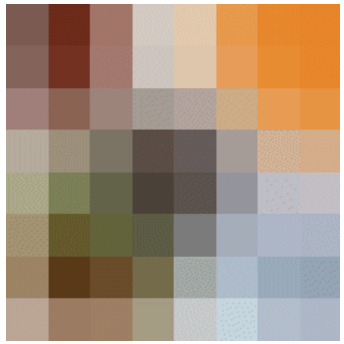
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) Matching: Determine correspondence between descriptors in two views

Geometric transformations

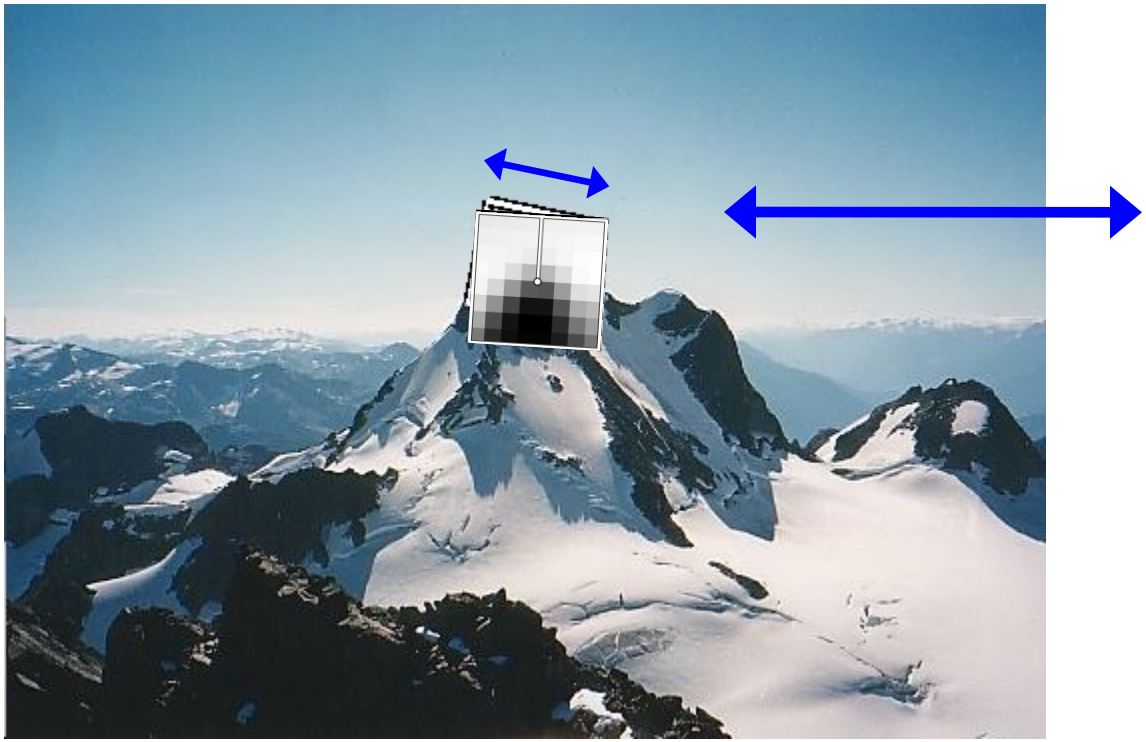






e.g. scale, translation, rotation

Making the descriptor rotationally invariant



- Rotate patch according to its dominant gradient orientation
  - It can be shown that the dominant direction is given by the first eigenvector (e_1, if $|\lambda_1|>|\lambda_2|$, and e_2, vice versa) of ***M***.

- This puts the patches into a **canonical orientation**.

Image from Matthew Brown

Illumination invariance



What does happen when image illumination change?

- Image gradient magnitude?
- Image gradient direction?

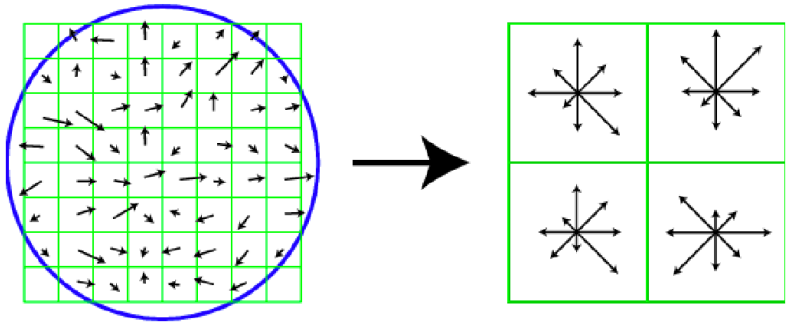
=> Define Sift descriptors based on Gradient direction.

SIFT Descriptor

[Lowe 2004]

Scale Invariant Feature Transform (SIFT)

- Uses histogram of gradients (relative to computed orientation) to describe pixels around the keypoint
- Gradients are computed in the corresponding scale-space image.



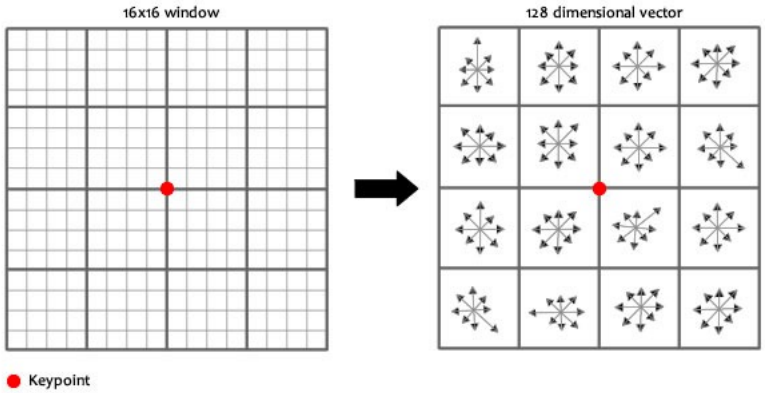
(a) image gradients (b) keypoint descriptor

The figure shows an 8x8 patch and 2x2 descriptor array with 8 bin histograms.

SIFT Descriptor

[Lowe 2004]

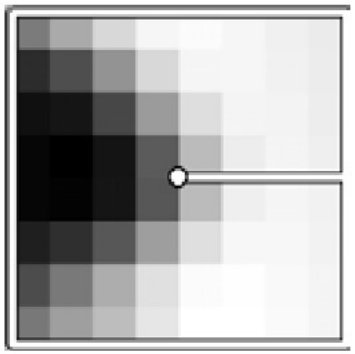
- Original SIFT descriptor uses 16x16 patches and 4x4 array of 8 bin histograms.
- Thus, each SIFT descriptor is made up of 4x4x8 = 128 non-negative numbers.



SIFT Descriptor

[Lowe 2004]

- Scale Invariant Feature Transform (SIFT)
  - Not robust to non-linear illumination changes.
  - Useful when dealing with images that do not get a lot distorted due to rotations or viewpoint changes.

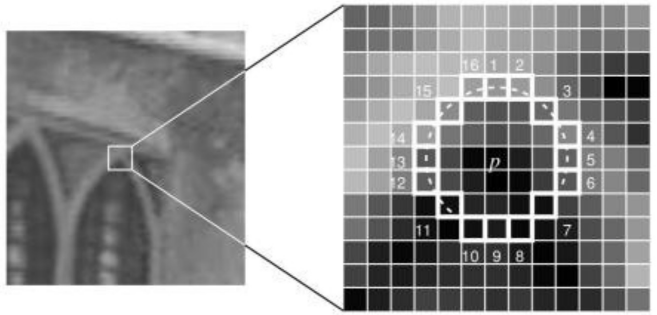


Explain the invariance

ORB feature descriptor (Approximation of SIFT)

- ORB (Oriented FAST (Features from Accelerated Segment Test) and Rotated BRIEF (Binary robust independent elementary feature))
- ORB advantages:
  - No licensing restrictions
  - Fast, computational efficient
  - Real time and low power device

- ORB is based on FAST corner detector
  - Compare intensity between center pixel and those in circular ring around the center



Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary R. Bradski , **ORB: An efficient alternative to SIFT or SURF**, ICCV 2011. doi: 10.1109/ICCV.2011.6126544.
https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html

ORB feature descriptor (Approximation of SIFT)

- Shortcomings of the ORB detection approach:
  1. No quality measure (“cornerness”).

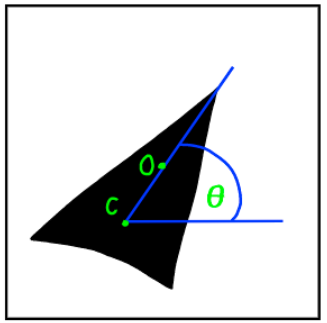
  Solution: use Harris cornerness measure

  2. Not scale invariant.

  Solution: use SIFT scale invariance (scale pyramid of the image).

  3. Not rotation invariant.

  Solution: calculate intensity centroid.
    - Corner’s intensity is offset from image center.

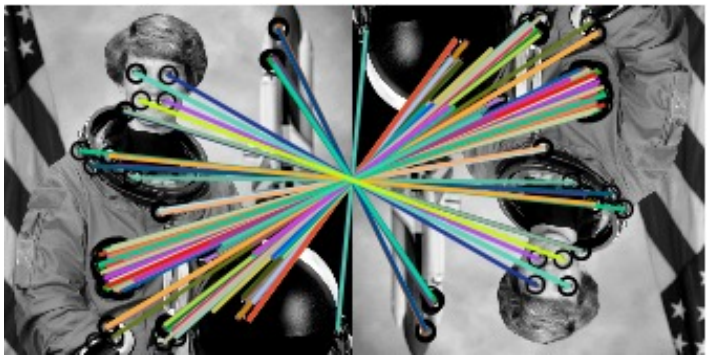


Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary R. Bradski , **ORB: An efficient alternative to SIFT or SURF**, ICCV 2011. doi: 10.1109/ICCV.2011.6126544.
https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html

ORB feature descriptor

- Scikit-image

```
>>descriptor_extractor = ORB(n_keypoints=200)
>>descriptor_extractor.detect_and_extract(img1)
>>keypoints1 = descriptor_extractor.keypoints
>>descriptors1 =descriptor_extractor.descriptors
…
>>matches12 = match_descriptors(descriptors1, descriptors2)
>>plot_matches(fig, img1, img2, keypoints1, keypoints2, matches12)
```

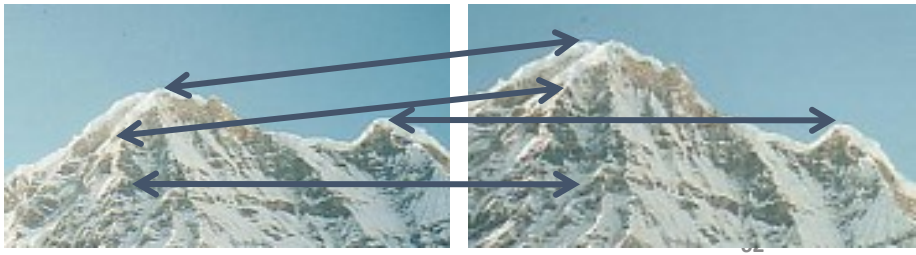


Local features: main components

1) Detection: Identify the interest points

2) Description:Extract vector feature descriptor surrounding each interest point.

3) **Matching**: Determine correspondence between descriptors in two views



Matching local features

Matching local features

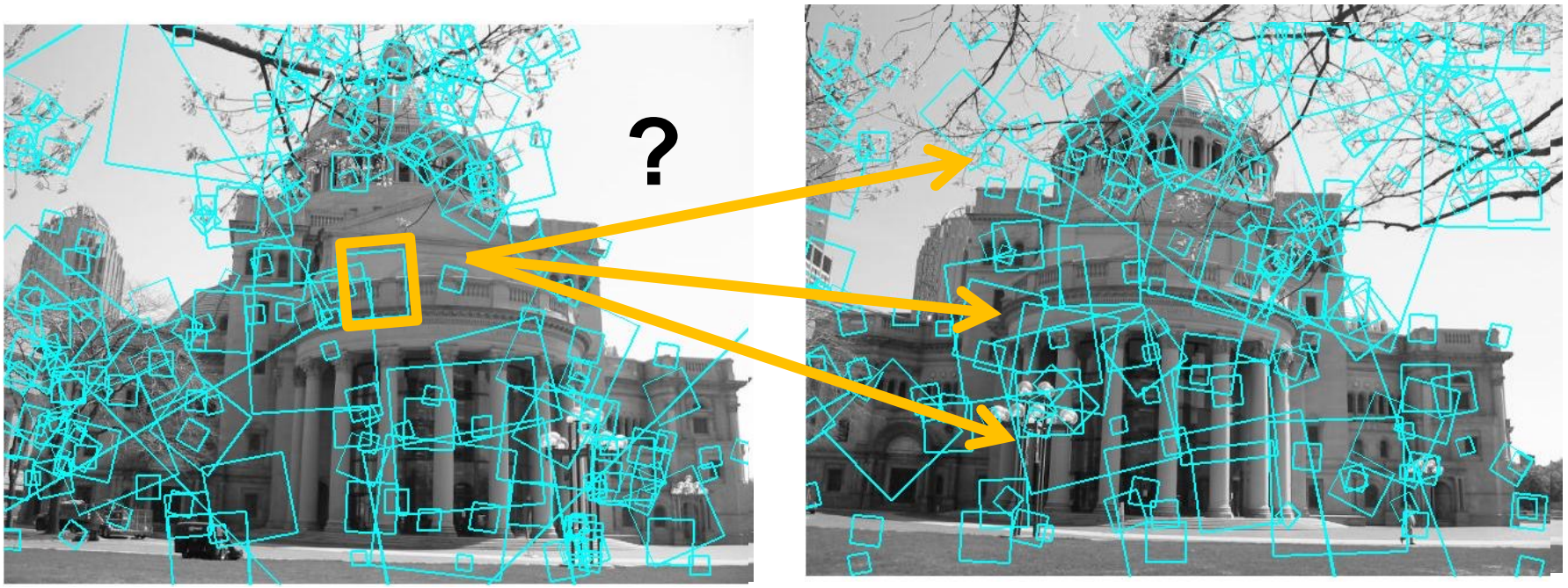


Image 1 Image 2

To generate **candidate matches**, find patches that have the most similar appearance (e.g., lowest Sum of Squared Distance (SSD)).

Simplest approaches:

- Compare all them
- Take the closest (or closest k, or within a thresholded distance).

Ambiguous matches

Image 1

Image 2

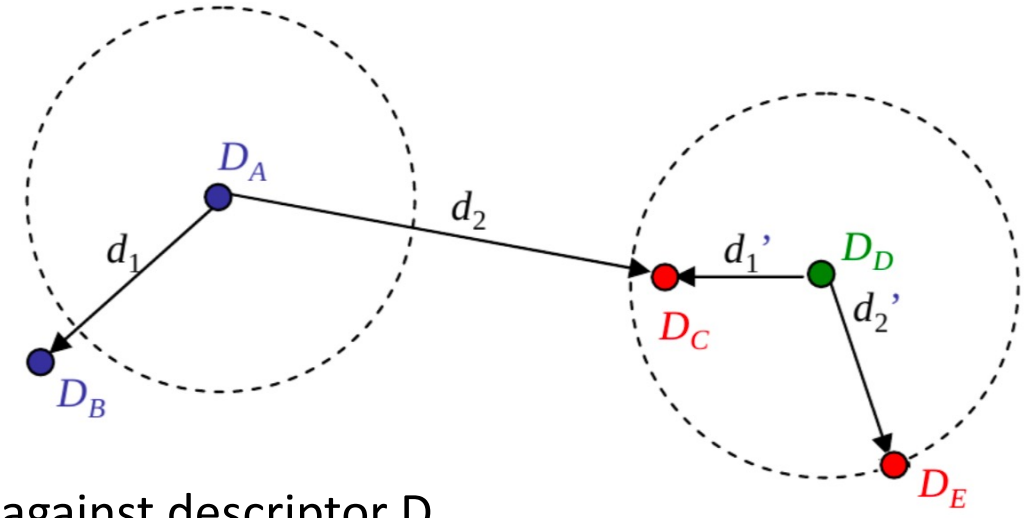
At what SSD value do we have a good match?

Ambiguous matches: Nearest neighbor distance ratio (NNDR)

- We may use a threshold (dashed circle in the figure) to select the candidate matches.
  - The useful range of thresholds can vary a lot depending on the kind of image.

Example:

Consider the points of the same color as a correct match.



- Descriptor D_A fails to match against descriptor D_B.
- D_D is incorrectly matched against two descriptors: D_C and D_E (D_D should match only against one descriptor).

Ambiguous matches: Nearest neighbor distance ratio (NNDR)

- To add robustness to matching, we can consider the **ratio**: d_1 / d_2, where d_1=distance to best match (**nearest neighbor**) and d_2=distance to second best match (**the second nearest neighbor**).
- If ratio is low, first match looks good.
- If ratio is high, it could be ambiguous match.

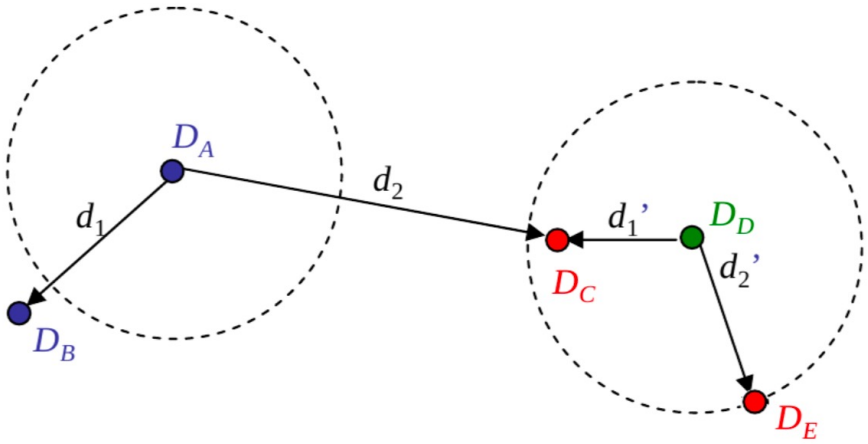
Example:

- Ratio d_1/d_2 is small,

thus D_A matches with D_B.

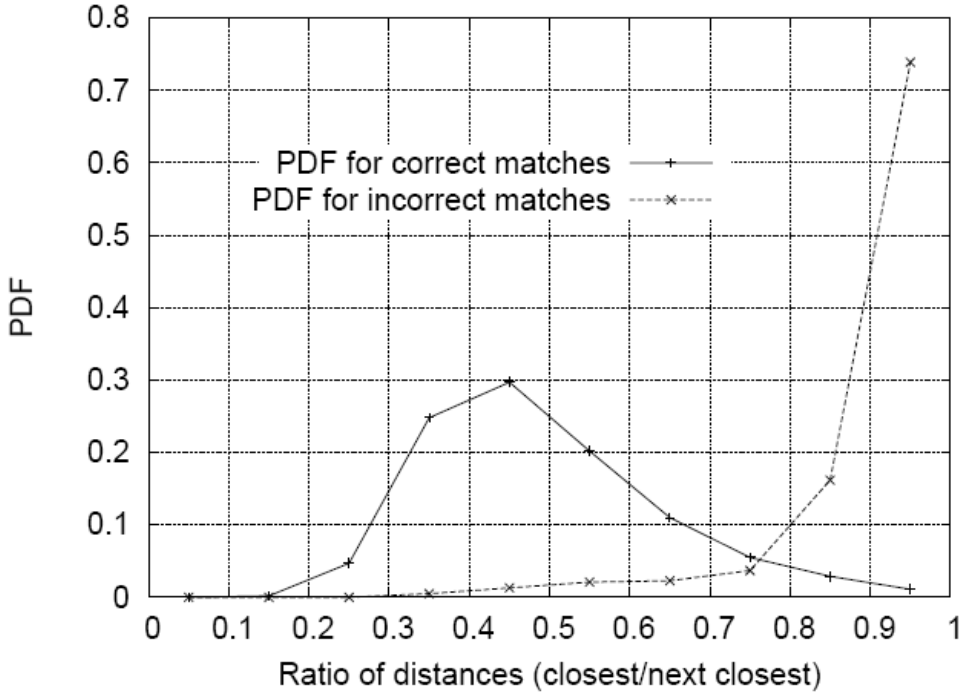
- But d'_1/d'_2 is large,

thus D_D does not match D_C .



Matching SIFT Descriptors

- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2nd nearest descriptor



$d(x, y_1)/d(x, y_2)$

Lowe IJCV 2004

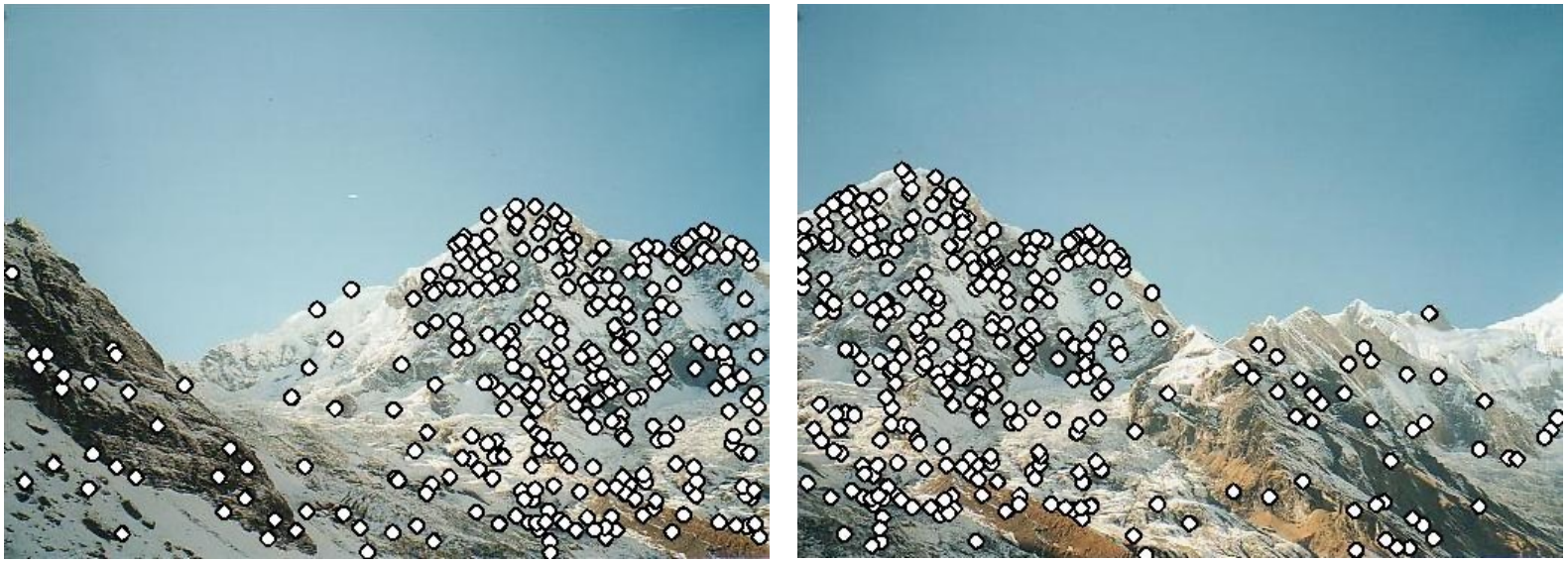
Robust feature-based alignment for Panorama creation.

Overall strategy:

1. Extract features (detect and describe)
2. Find some useful matches (*putative matches)*
3. Compute the best transformation (affine, translation,…)
4. Apply this transformation for panorama creation

Robust feature-based alignment

Robust feature-based alignment



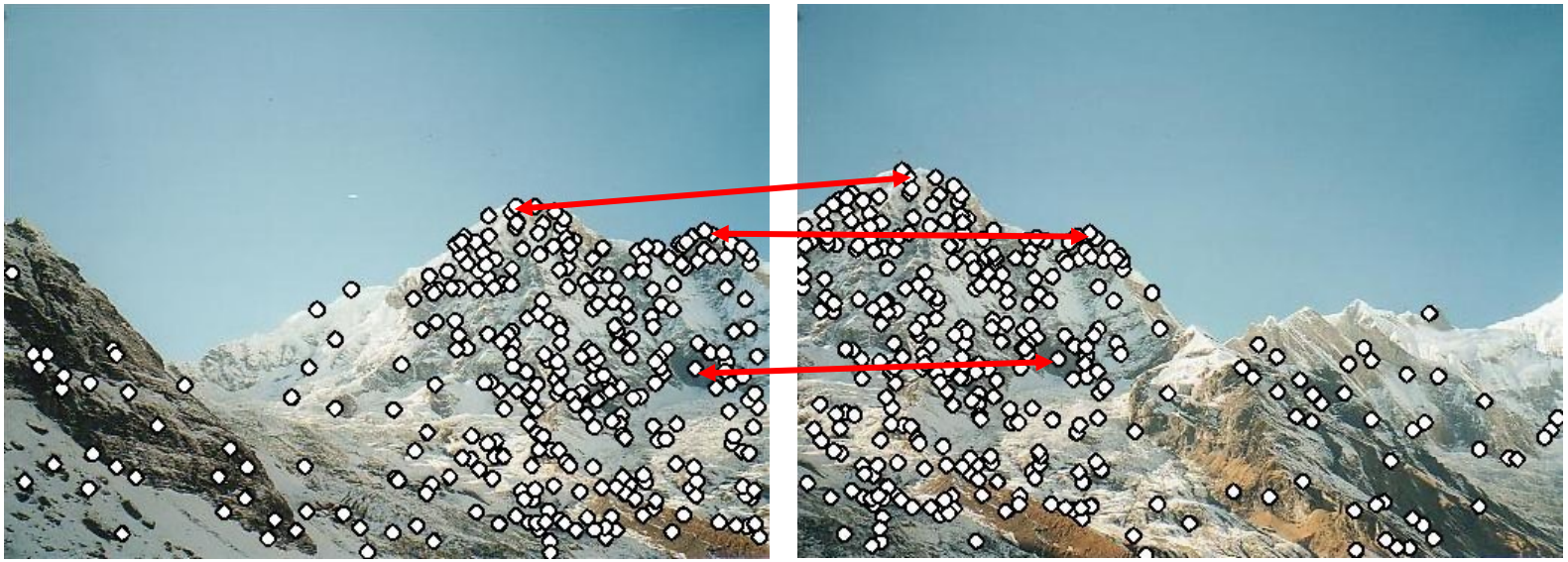
- Extract features in both images (independently)

Robust feature-based alignment

- Extract features
- Compute *putative matches*

How can we be robust with respect the false correspondences?

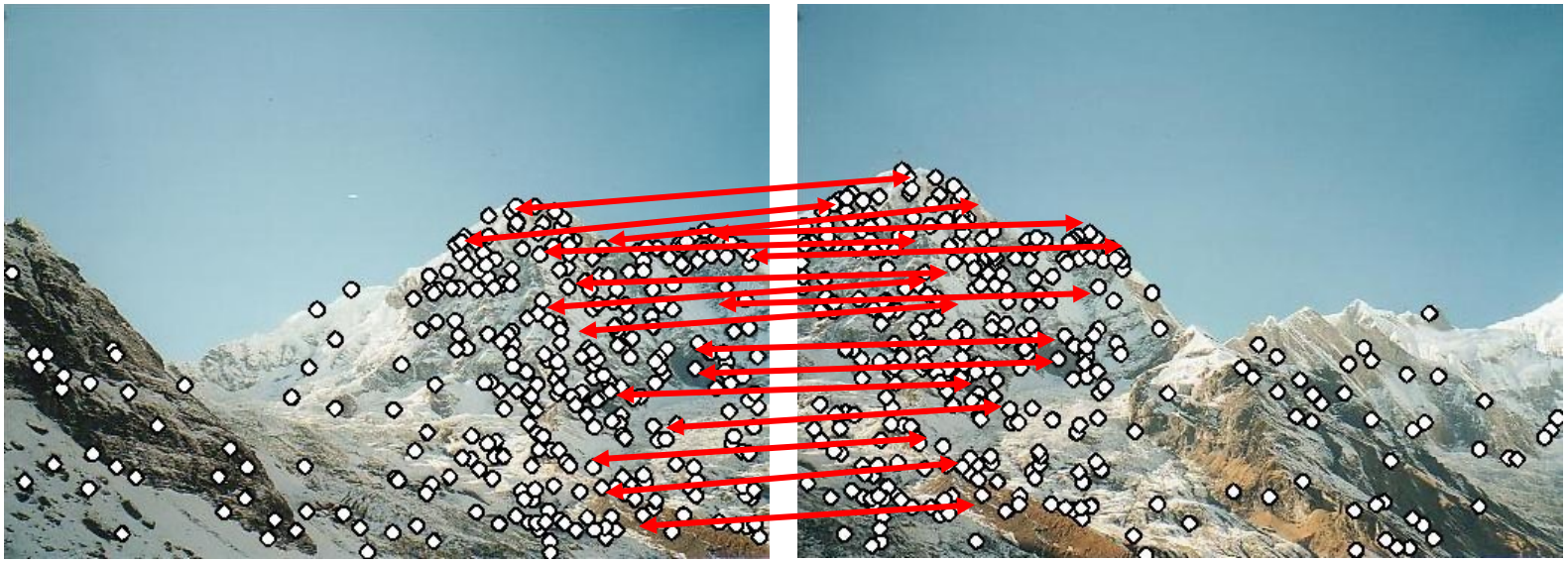
Robust feature-based alignment



- Extract features
- Compute *putative matches*
- *Hypothesize* transformation *T* from putative matches (typically with least squares).

We assume that the whole image is transformed according to a particular motion model (translation, affine transformation)

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- *Hypothesize* transformation *T* from putative matches (all or e.g. the best 10 matches)
- *Verify* transformation (search for other matches consistent with *T*)

Robust feature-based alignment

- Extract features
- Compute *putative matches*
- *Hypothesize* transformation *T* from putative matches
- *Verify* transformation (search for other matches consistent with *T*)
- Apply transformation

Robust feature-based alignment

Take into account that:

- With the least squares method we assume that the noise follows a Gaussian distribution.
  - Here the noise are the matchings that do not follow the motion model called **outliers**.
- More robust versions of least squares are required when the outliers degrade the model. This may happen if the number of outliers is large.

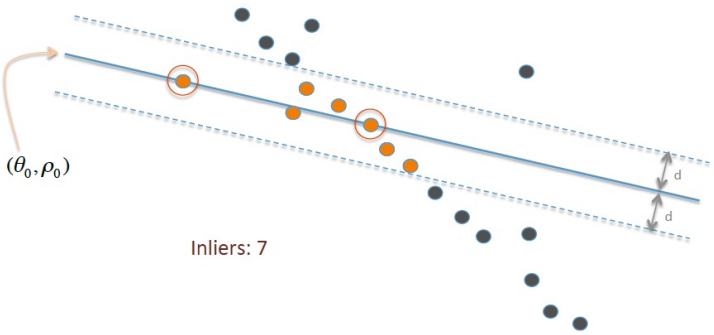
Other techniques are required. For instance, we may use

- Robust energies (e.g.. take the absolute value rather than the square of the error). Gradient descent techniques are then needed since no closed solution exists.
- Apply least squares but use alternative techniques rather than simply select e.g. the 10 best matchings.
  - Let's see a method called **RANSAC**.

Ransac (Random Sample Consensus) method

Some remarks:

- Fitting a line (model) is easy if we know which points belong and which do not.
- If we had a proposed line (model), we could guess which points belongs to that line (model): these are called **inliers.**



The point p is an inlier if error in p < d

- RANSAC: randomly pick some points to define your line (model). Repeat enough times until you find a good one.
- A good one means one that have many inliers.

Mosaic construction by the Ransac method

The algorithm:

1. Select a **random** subset of of k correspondences (hypothetical inliers).

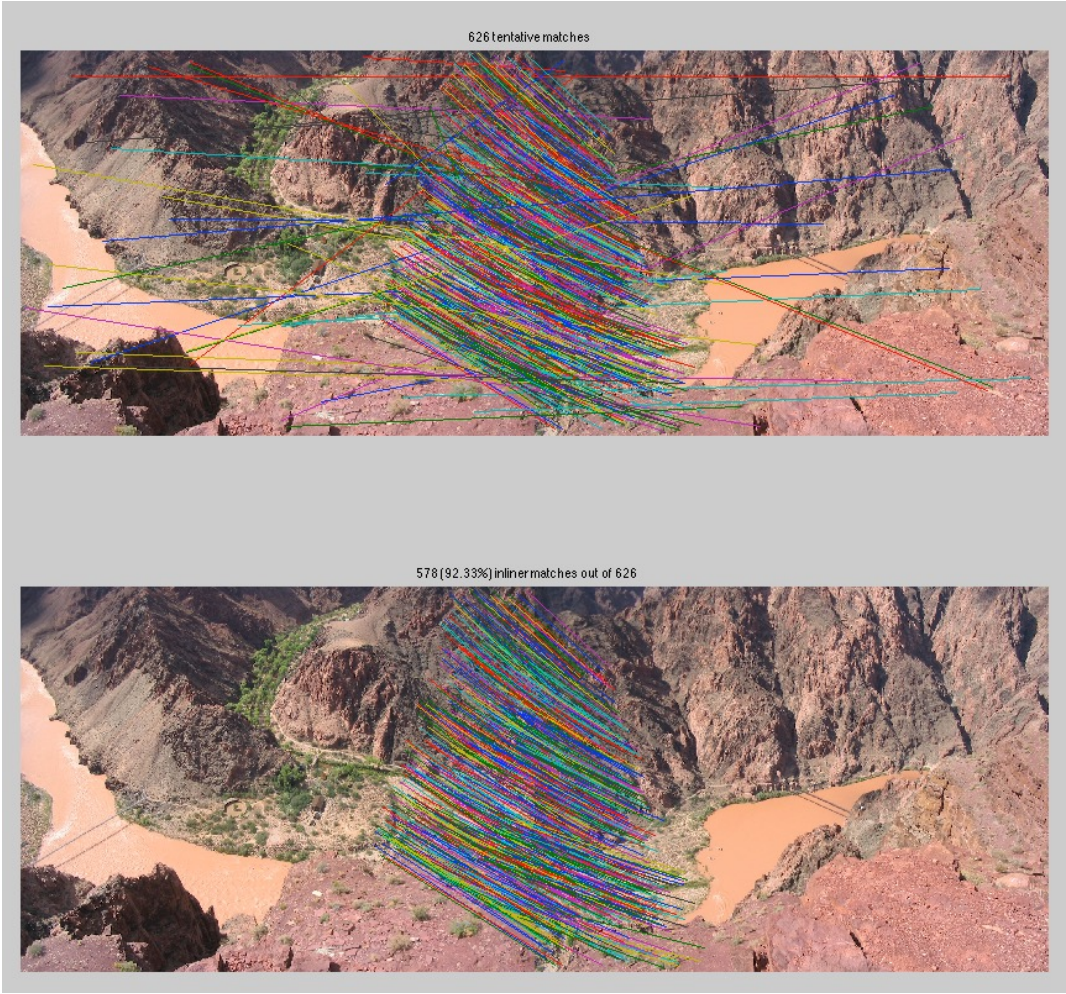
2. A transformation model is fitted to this random subset.

3. All other data are then tested against the fitted model.
   - Those points that fit the estimated model well, according to some model-specific loss function, are considered as part of the consensus set.
   - The estimated model is reasonably good if sufficiently many points have been classified as part of the consensus set.

4. This procedure is repeated a fixed number of times (S).
   - Each time producing either a model which is rejected because too few points are part of the consensus set, or a refined model together with a corresponding consensus set size.
   - In the latter case, we keep the refined model if its consensus set is larger than the previously saved model.

Construct the mosaic through SIFT and Ransac

Construct the mosaic through SIFT and Ransac



Construct the mosaic through SIFT and Ransac

Summary: SIFT descriptor [Lowe 2004]

- Robust matching technique
  - Fast and efficient—can run in real time
  - Can handle changes in rotation and viewpoint (Up to about 60 degree out of plane rotation)
  - Can handle significant changes in illumination (sometimes even day vs. night)
  - Based on histograms of Gradients in patches for robustness to small shifts and translations
- Partially invariant to occlusion, clutter

References

- Source for slides are from:
  - K. Grauman, L. Lazebnik, Steve Seitz, B. Leibe, J. Tompkin, James Hays, Derek Hoiem, Navneet Dalal and Bill Triggs
  - Petia Radeva, Lluís Garrido
- Readings:
  - Szeliski: Computer Vision: Algorithms and applications, Chapter 4, Feature extraction and matching
  - Article: https://medium.com/software-incubator/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf

Lecture videos

Matching with Features

- From 432 – Intro. Until 438 – End.

Harris corner:

From 439 – Intro. Until 455 – End.

SIFT:

- From 465 - Last Time Until 477 – End.

RANSAC:

- From 498 – Intro Until 510 – End.

From *Introduction to Computer Vision*:

https://www.udacity.com/course/introduction-to-computer-vision--ud810

Final Test

- **Go to:** *Socrative.com*
  - Choose Student Login
  - Room Name: COMPUTERVISION
  - Enter your name: It can be anonymous.

COMPUTATIONAL VISION: Image Features (SIFT)

Master in Artificial Intelligence

Department of Mathematics and Computer Science

UNIVERSITAT DE BARCELONA