ROBOTICS & PERCEPTION GROUP

Universität Zürich[UZH]

Lecture 06
Point Feature Detection and Matching
Part 2

Davide Scaramuzza

Mini-project

Goal: implement a Visual Odometry (VO) pipeline

- Groups: 1 to 4 students
- Hand-in:
  - **Code** (Matlab, or alternatively runnable on Ubuntu 14.04)
  - **Report** (free-form, 5 pages max)
- Goal of report: show us what work you did, what failed and what worked...

Grading:

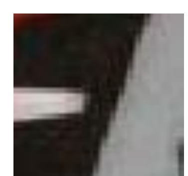
- 4.5-5.5: working VO pipeline (grade depends on accuracy)
- 5.5-6: working VO pipeline with extra features (not covered during the exercises)
- < 4.5: pipelines that don't work. The grade will be based on the report.

Scale changes

- How can we match image patches corresponding to the same feature but belonging to images taken at different scales?
  - Possible solution: rescale the patch!

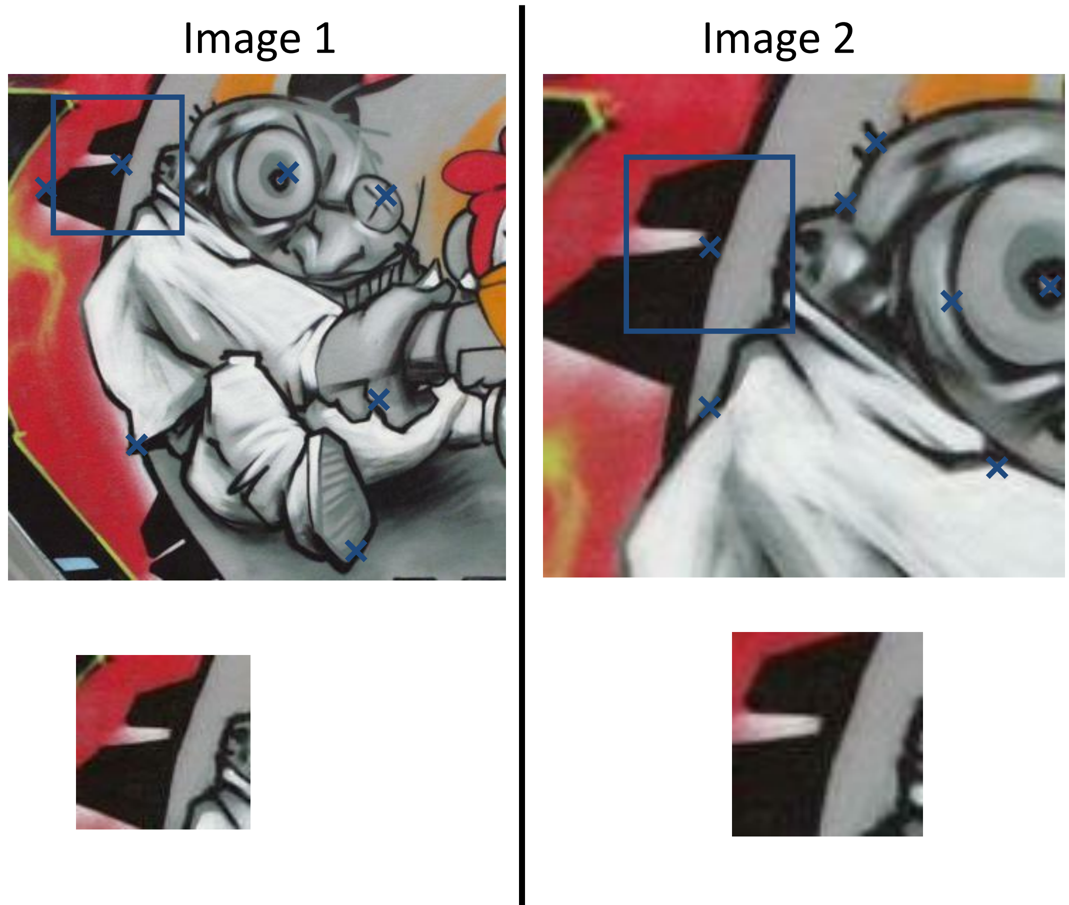
Image 1

Image 2



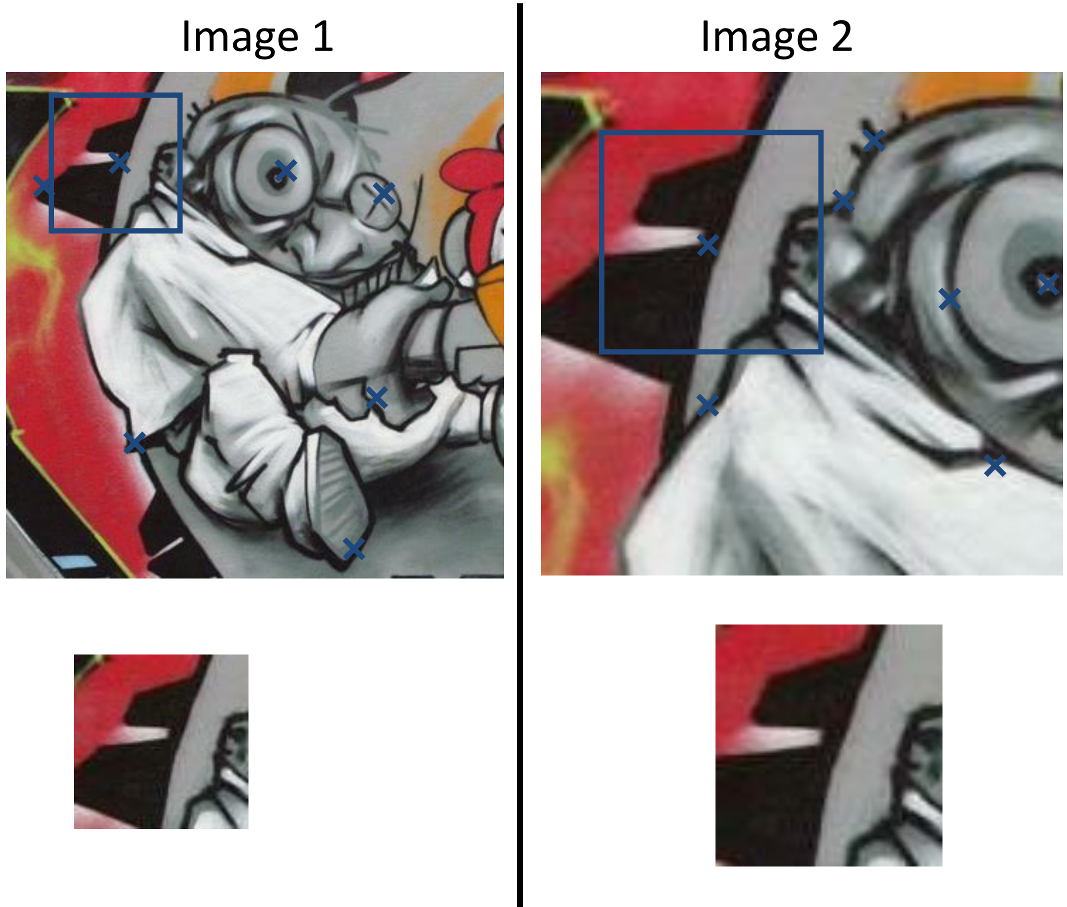
Scale changes

- How can we match image patches corresponding to the same feature but belonging to images taken at different scales?
  - Possible solution: rescale the patch!



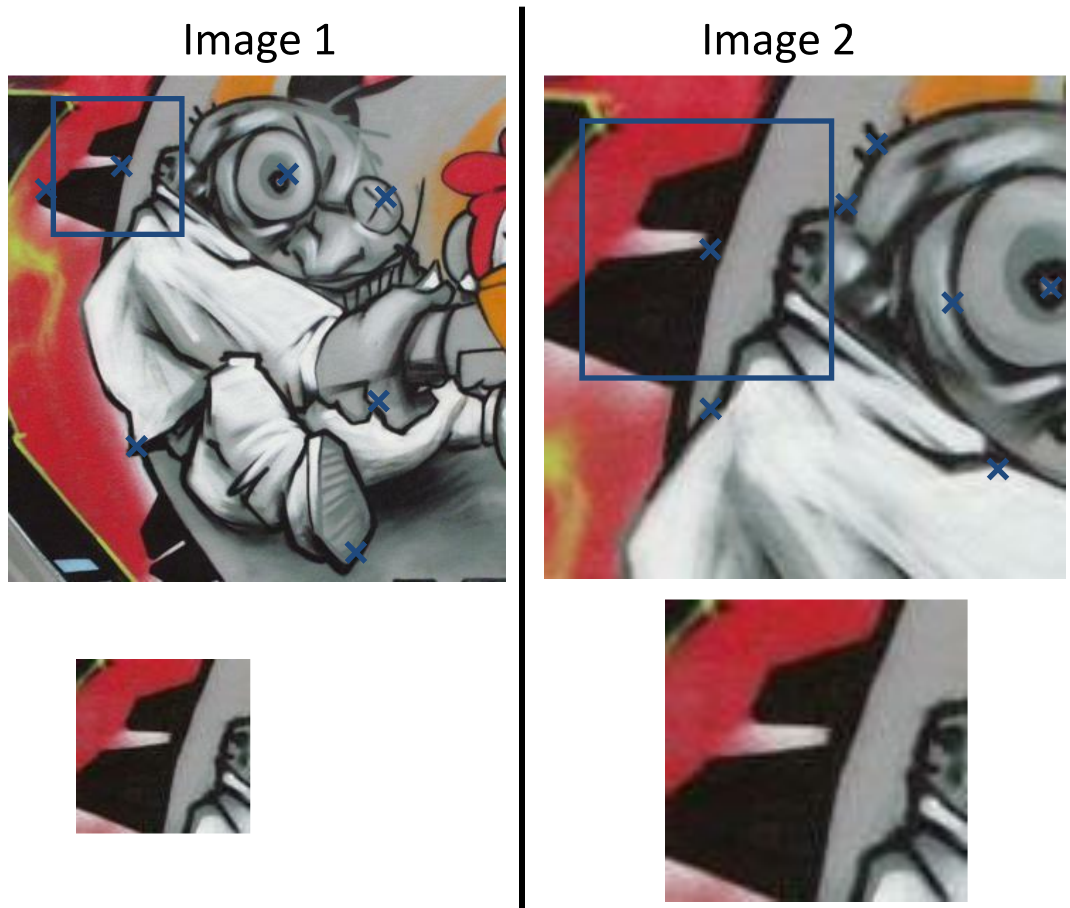
Scale changes

- How can we match image patches corresponding to the same feature but belonging to images taken at different scales?
  - Possible solution: rescale the patch!



Scale changes

- How can we match image patches corresponding to the same feature but belonging to images taken at different scales?
  - Possible solution: rescale the patch!



Scale changes

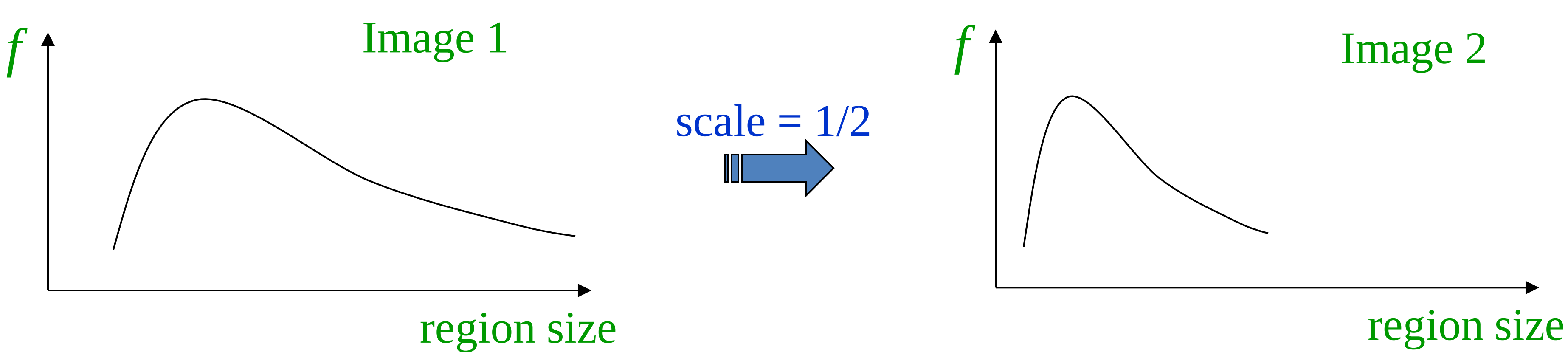
- Scale search is time consuming (needs to be done individually for all patches in one image)
  - Complexity would be $(NM)^2$ (assuming that we have N features per image and M scale levels for each image)
- Possible solution: assign each feature its own “scale” (i.e., size).
  - What’s the optimal scale (i.e., size) of the patch?

Automatic Scale Selection

- Solution:
  - Design a function on the image patch, which is “scale invariant” (i.e., which has the same value for corresponding regions, even if they are at different scale*s*)

Can this function be the Cornerness Response function?
Answer: no! Why? What kind of behavior does it have?

  - For a point in one image, we can consider it as a function of region size (patch width)



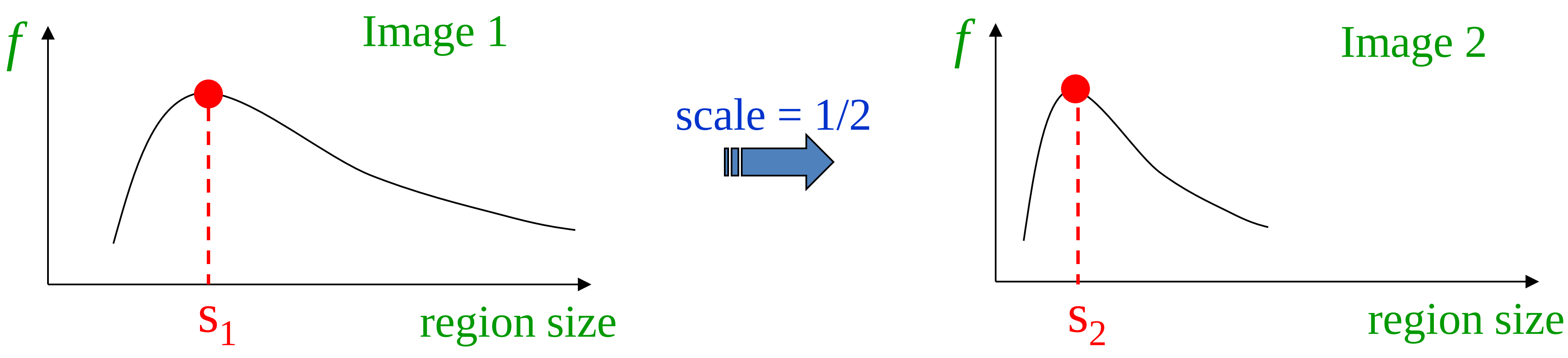
Automatic Scale Selection

- Common approach:

Take a local maximum of this function

Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

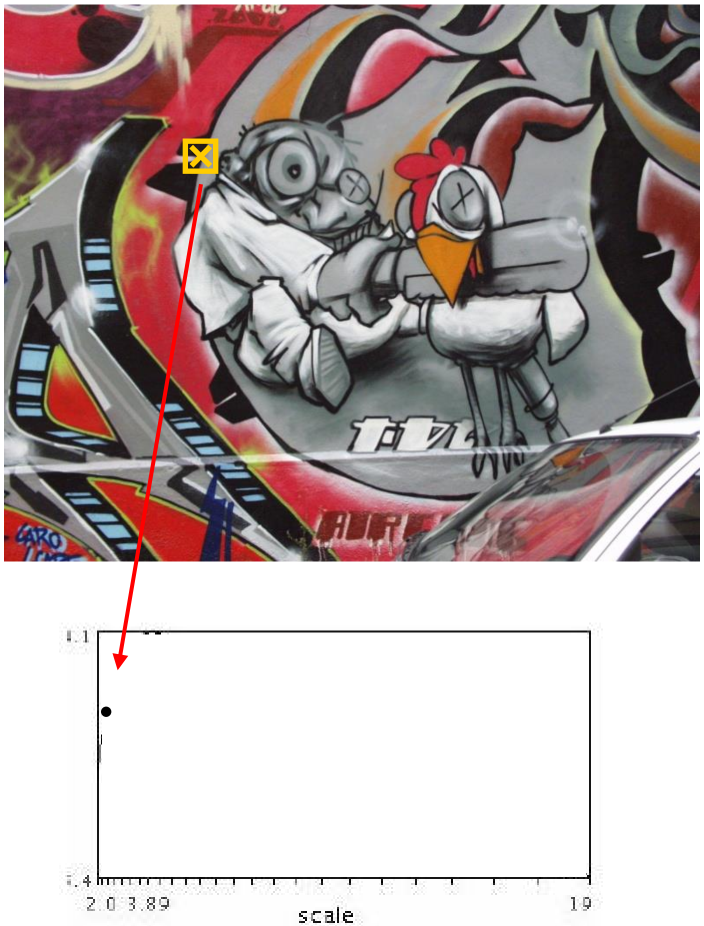
Important: this scale invariant region size is found in each image independently!



Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Image 1



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

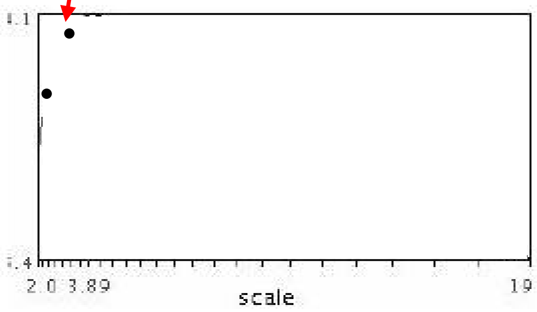
Image 2

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

Automatic Scale Selection

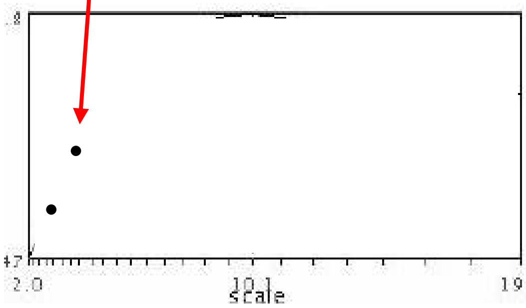
- Function responses for increasing scale (scale signature)

Image 1



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

Image 2

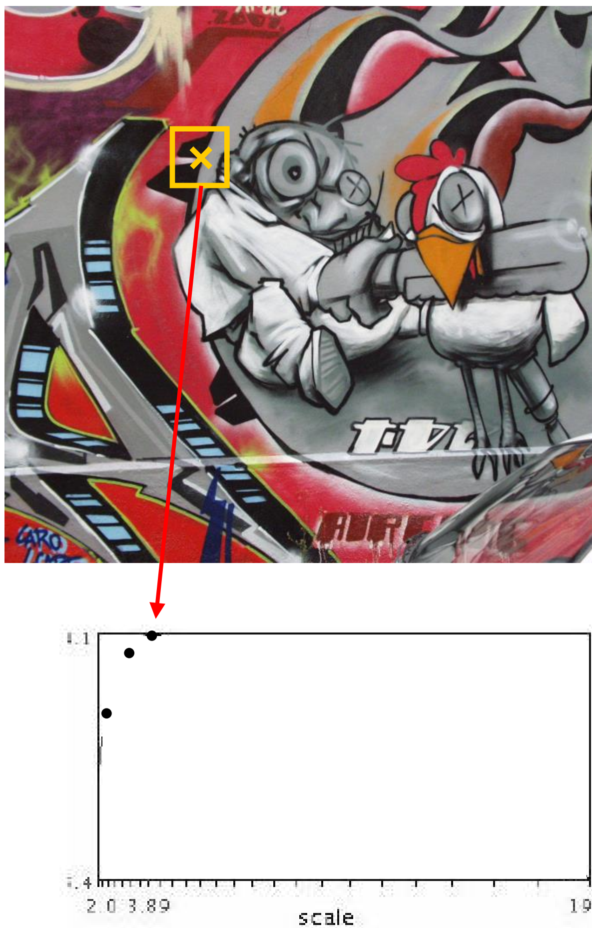


$f(I_{i_1 \ldots i_m}(x', \sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Image 1



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

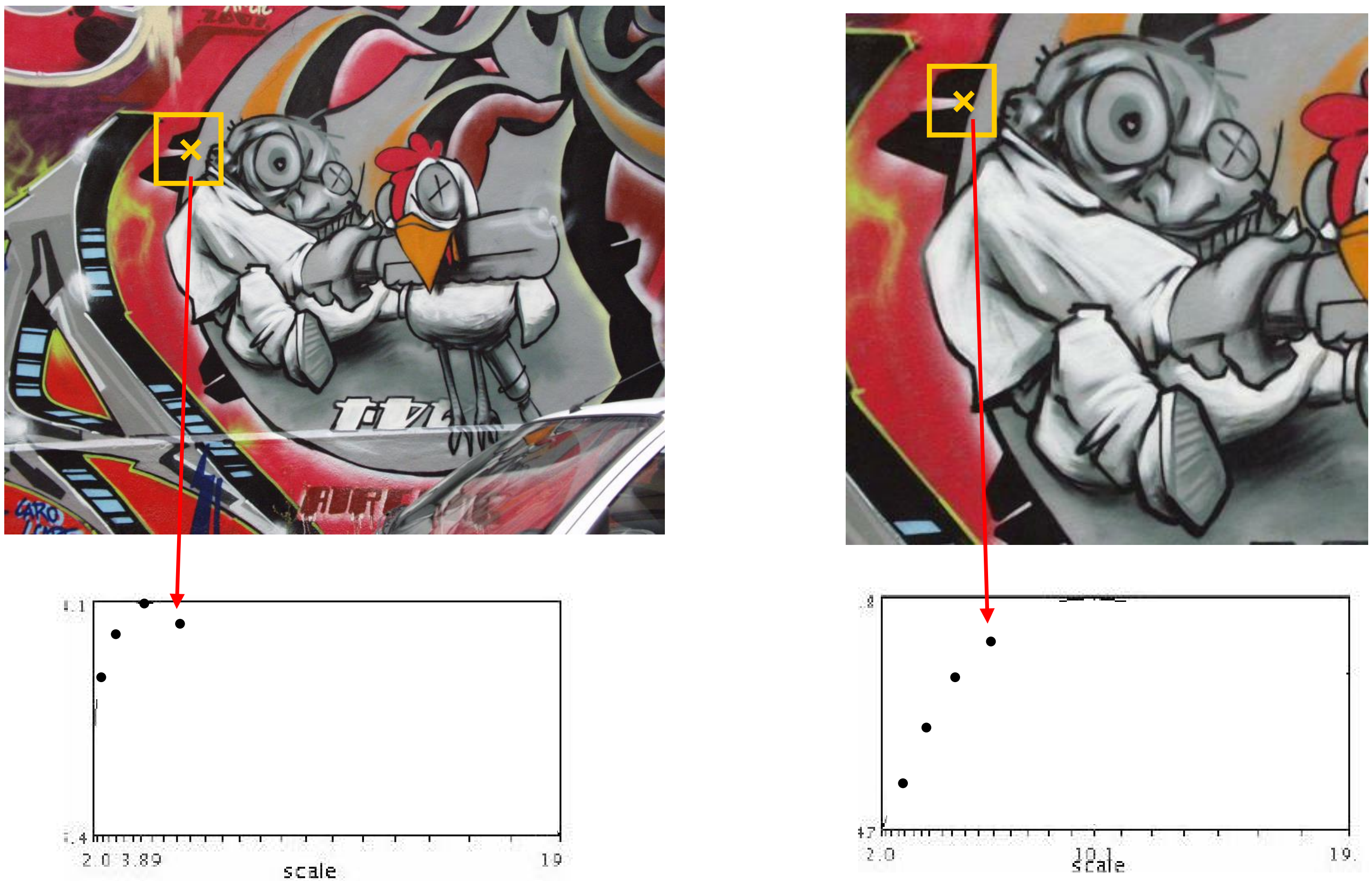
Image 2

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

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Image 1 | Image 2

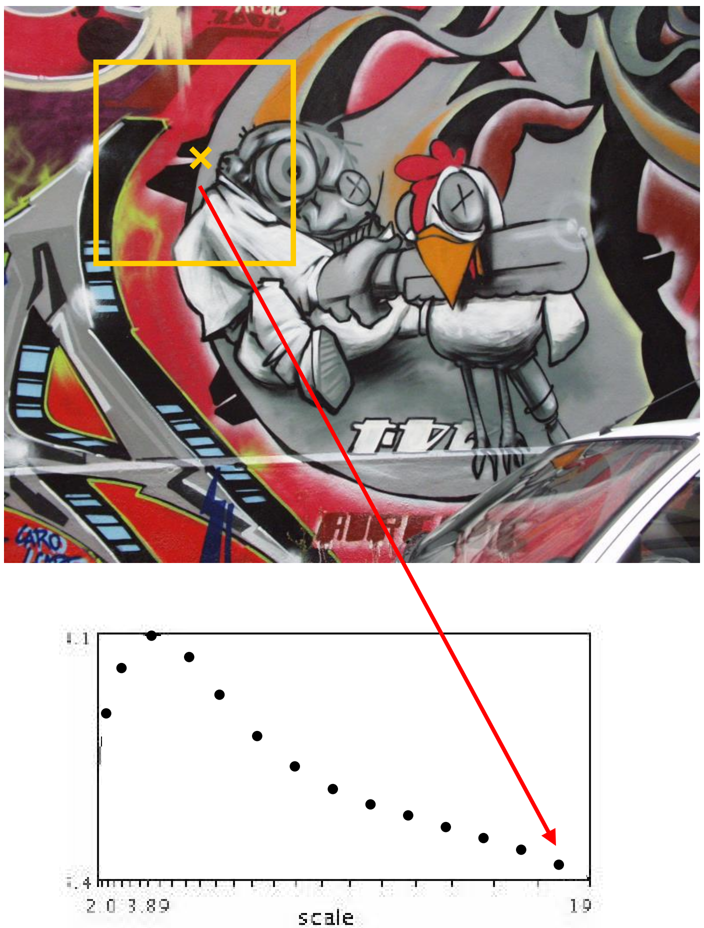


$f(I_{i_1 \ldots i_m}(x,\sigma))$ | $f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

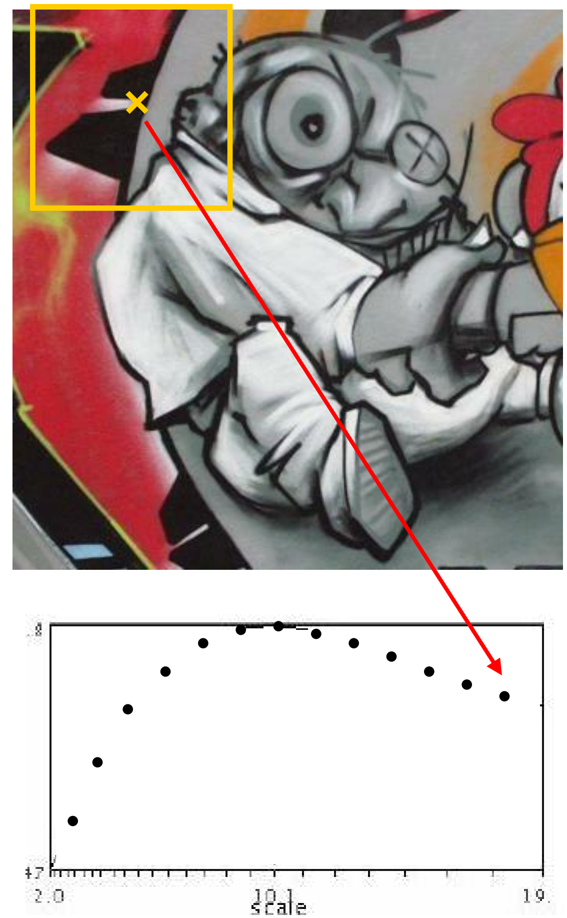
- Function responses for increasing scale (scale signature)

Image 1



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

Image 2

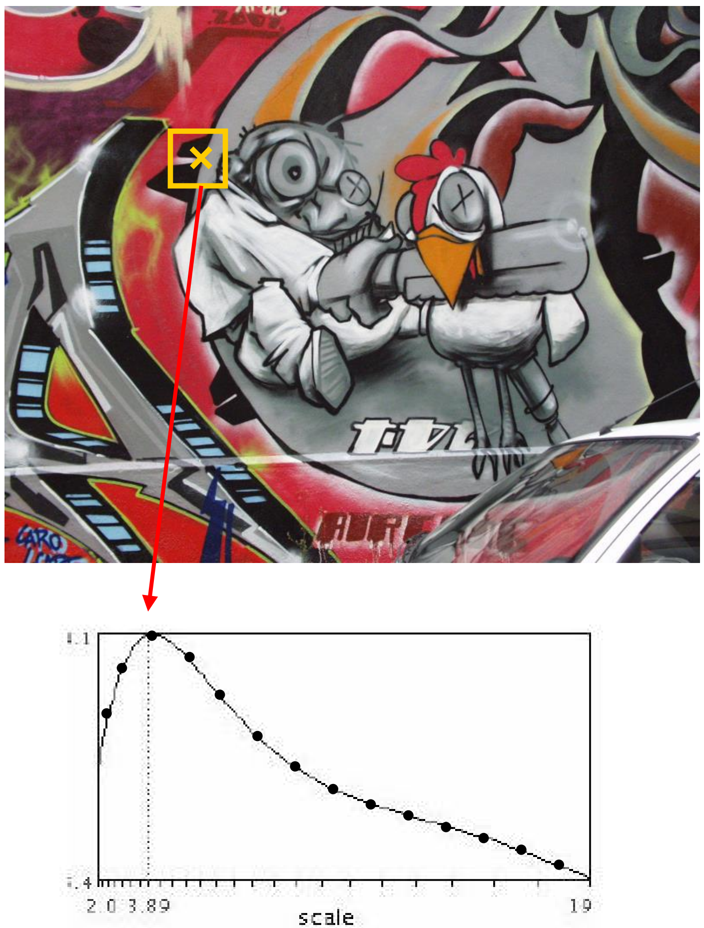


$$f(I_{i_1 \ldots i_m}(x', \sigma))$$

Automatic Scale Selection

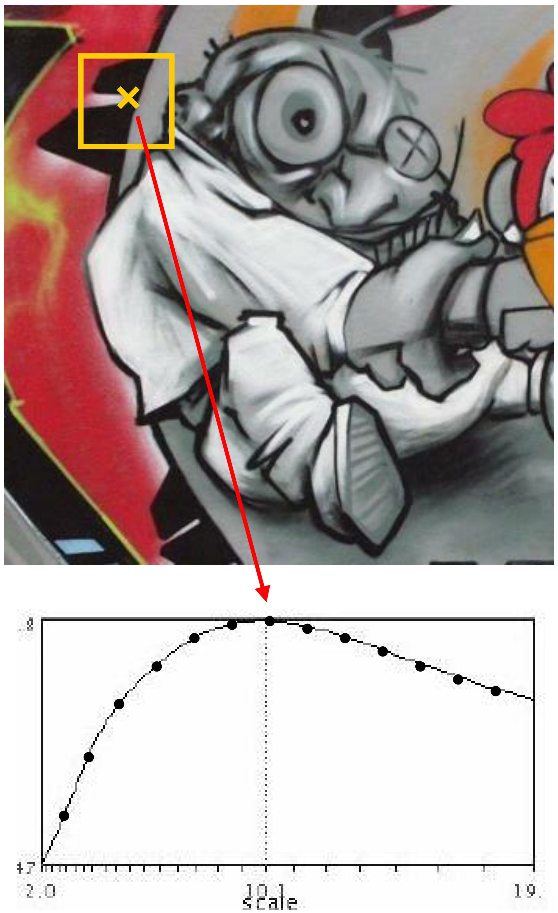
- Function responses for increasing scale (scale signature)

Image 1



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

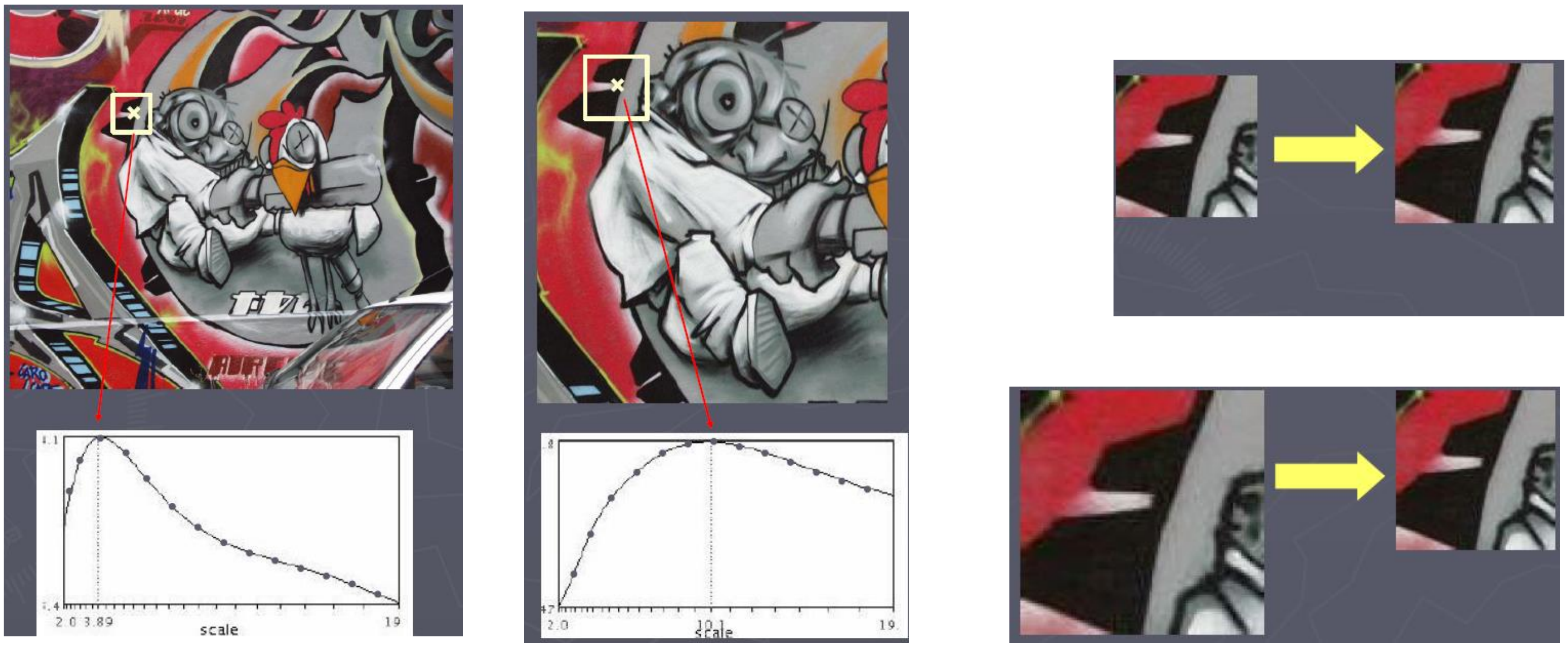
Image 2



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

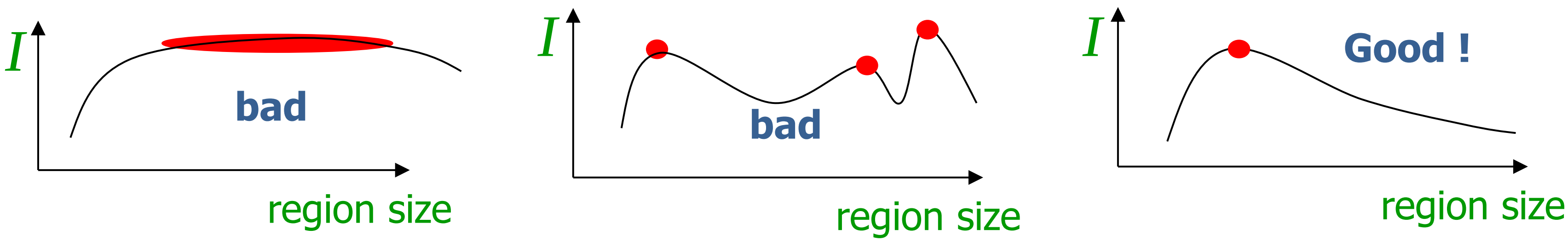
Automatic Scale Selection

- When the right scale is found, the patch must be normalized



Scale Invariant Detection: Robustness

- A “good” function for scale detection should have a single & sharp peak



A cornerness response function would exhibit this “flat ”behavior, why?

- Sharp, local intensity changes are good regions to monitor in order to identify the scale

⇨ Blobs and corners are the ideal locations!

Scale Invariant Detection

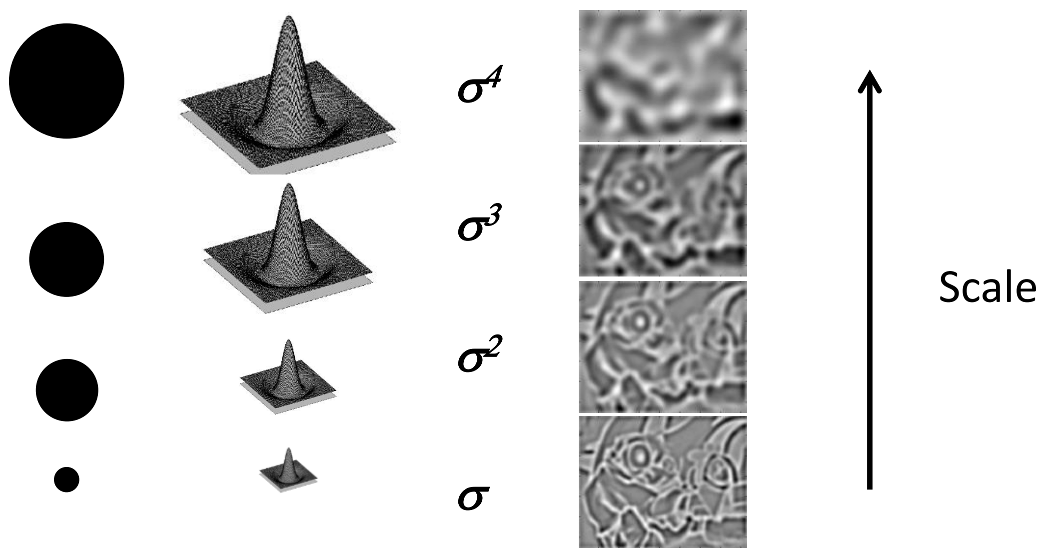
- Functions for determining scale: convolve image with kernel to identify sharp intensity discontinuities

$$f = \text{Kernel} * \text{Image}$$

- Laplacian of Gaussian kernel:

$$LoG = \nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

- Correct scale is found as local maxima across consecutive smoothed images



Scale Invariant Detection

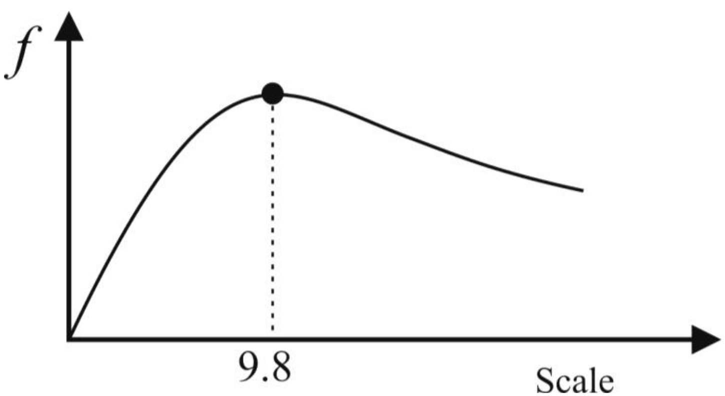
- Functions for determining scale: convolve image with kernel to identify sharp intensity discontinuities

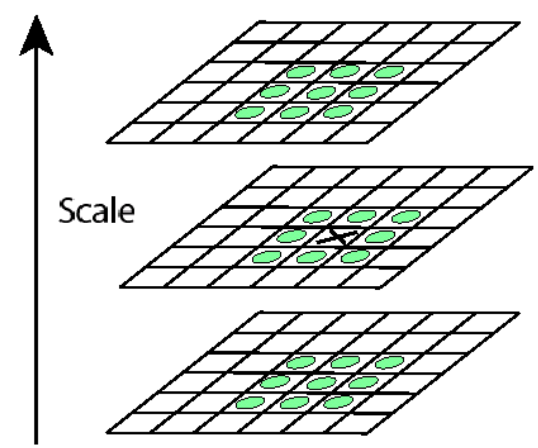
$$f = \text{Kernel} * \text{Image}$$

- Laplacian of Gaussian kernel:

$$LoG = \nabla^2 G(x, y) = \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2}$$

- Correct scale is found as local maxima across consecutive smoothed images



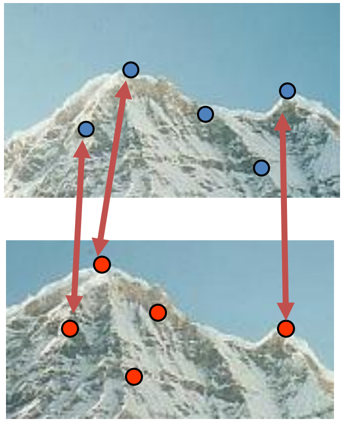


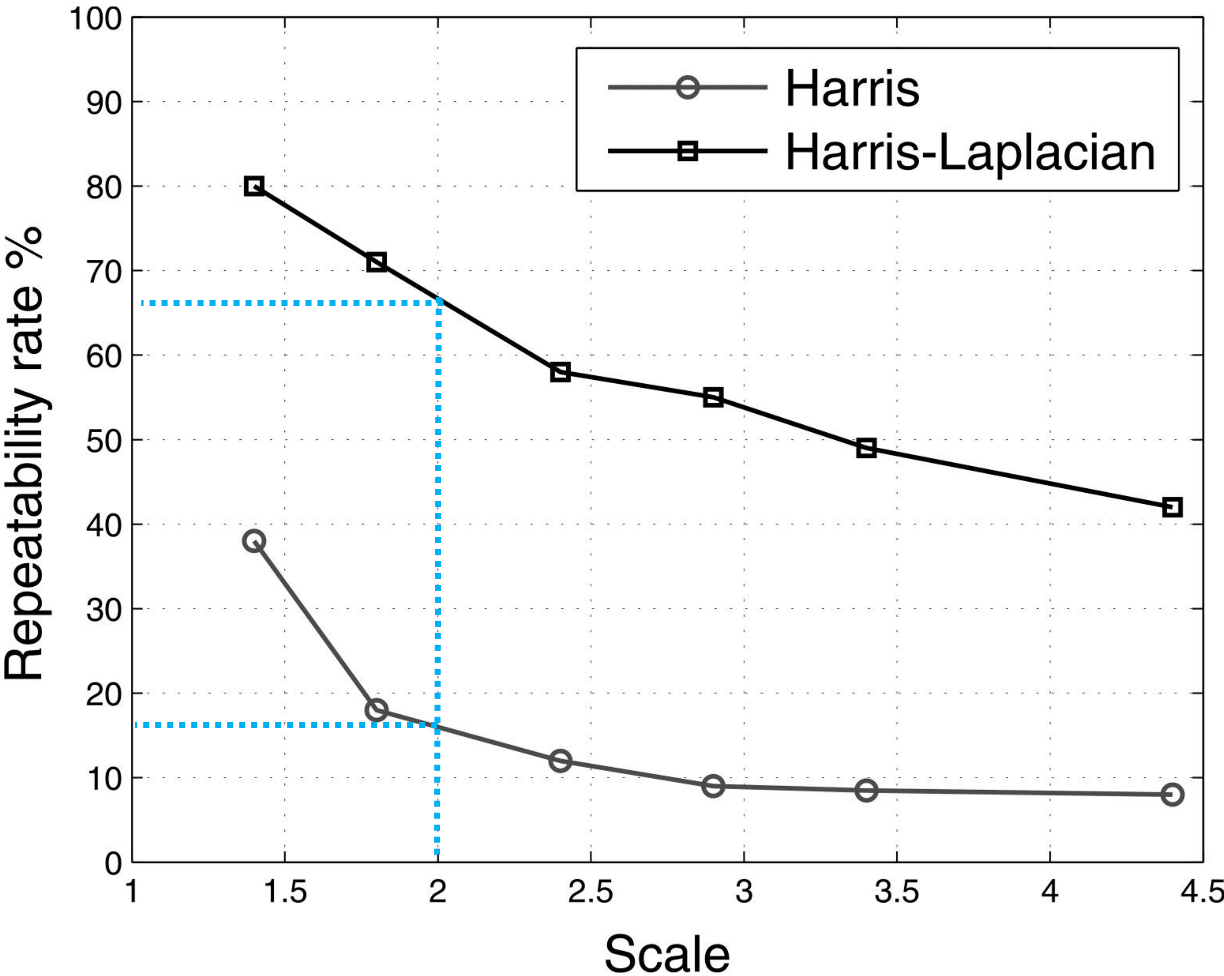
Scale Invariant Detectors

- Experimental evaluation of detectors w.r.t. scale change

Repeatability=

$$\frac{\text{\# correspondences detected}}{\text{\# correspondences present}}$$





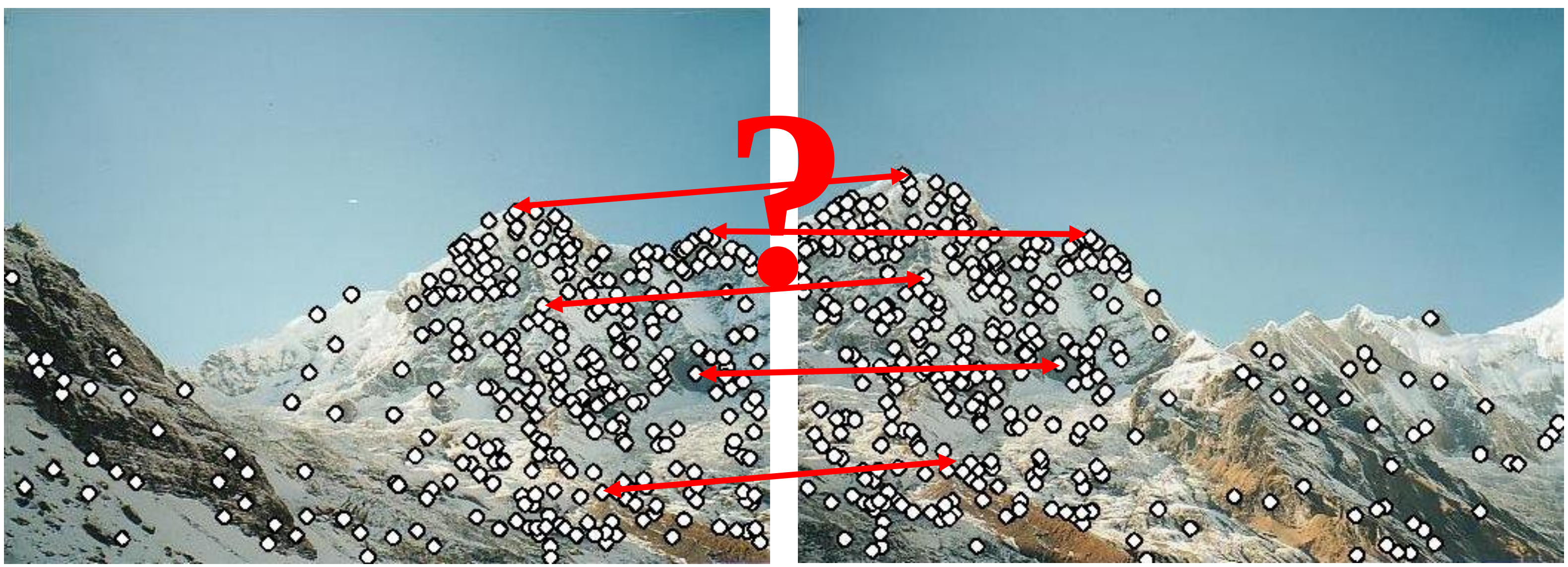
Main questions

- Where will the interest points come from?
  - What are salient features that we'll *detect* in multiple views?
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?

Feature descriptors

- We know how to detect points
- Next question:

How to *describe* them for matching?



- **Simplest descriptor**: **intensity values** within a squared patch or gradient histogram
- Alternative: **Histograms of Oriented Gradients** (like in SIFT, see later)
- Then, descriptor matching can be done using **(Z)SSD, (Z)SAD, or (Z)NCC**

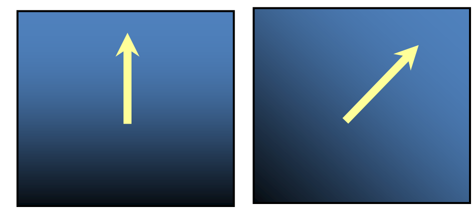
Feature descriptors

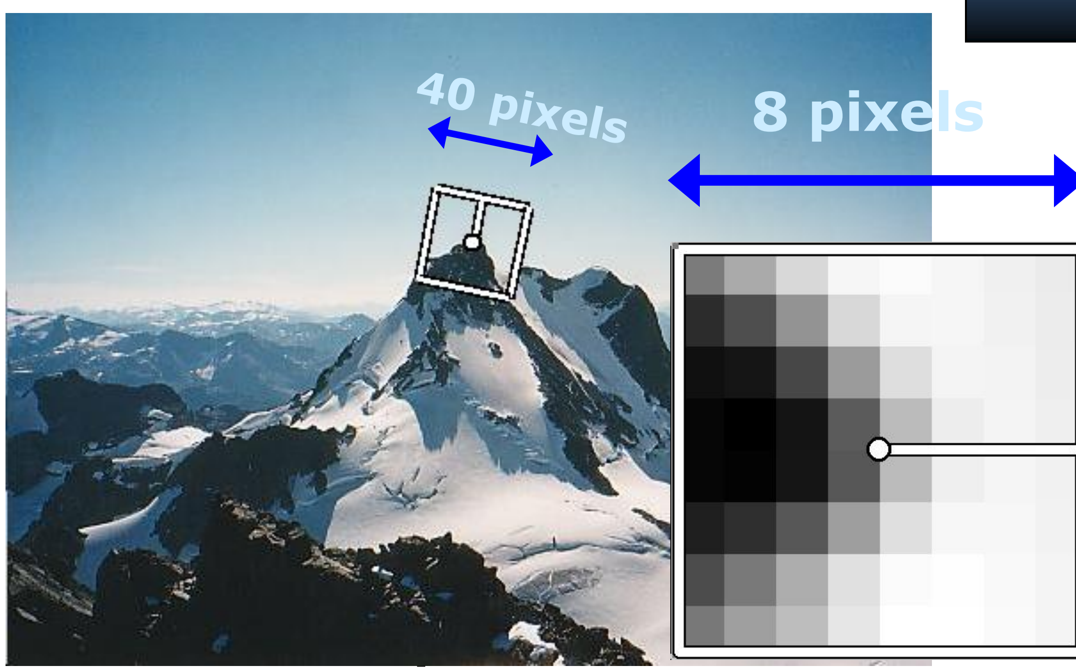
- We'd like to find the same features regardless of the **transformation** (*rotation, scale, view point,* and *illumination)*
  - **Most feature methods** are designed to be invariant to
    - translation,
    - 2D rotation,
    - scale
  - Some of them can also handle
    - **Small view-point invariance** (3D rotation) (e.g., SIFT works up to about 60 degrees)
    - **Linear illumination changes**

How to achieve invariance

Step 1: Re-scaling and De-rotation

- **Find correct scale** using LoG operator
- **Rescale the patch** to a default size (e.g., 8x8 pixels)
- **Find local orientation**
  - Dominant direction of gradient for the image patch (e.g., Harris eigenvectors)
- **De-rotate patch**
  - This puts the patches into a canonical orientation

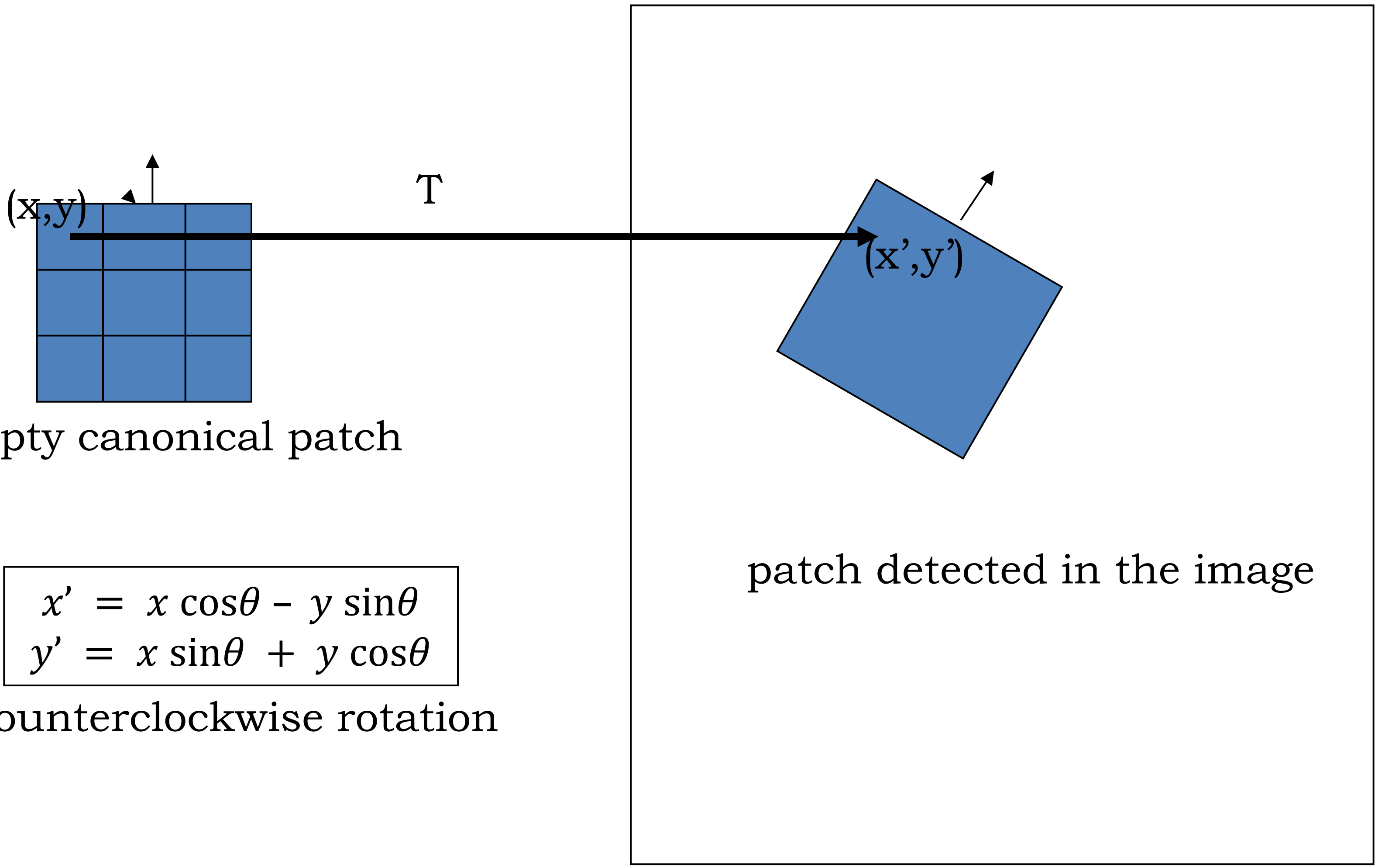




Implementation Concern:
How do you rotate a patch?

- Start with an “empty” patch whose dominant direction is “up”.
- For each pixel in your patch, compute the position in the detected image patch. It will be in floating point and will fall between the image pixels.
- Interpolate the values of the 4 closest pixels in the image, to get a value for the pixel in your patch.

Rotating a Patch

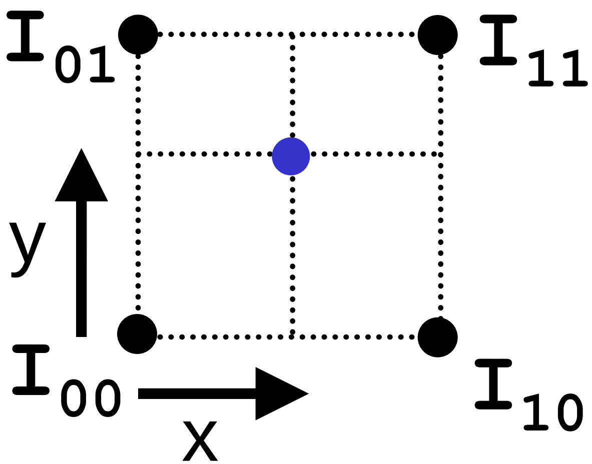


$$T \quad \boxed{\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}}$$

counterclockwise rotation

Using Bilinear Interpolation

- Use all 4 adjacent samples

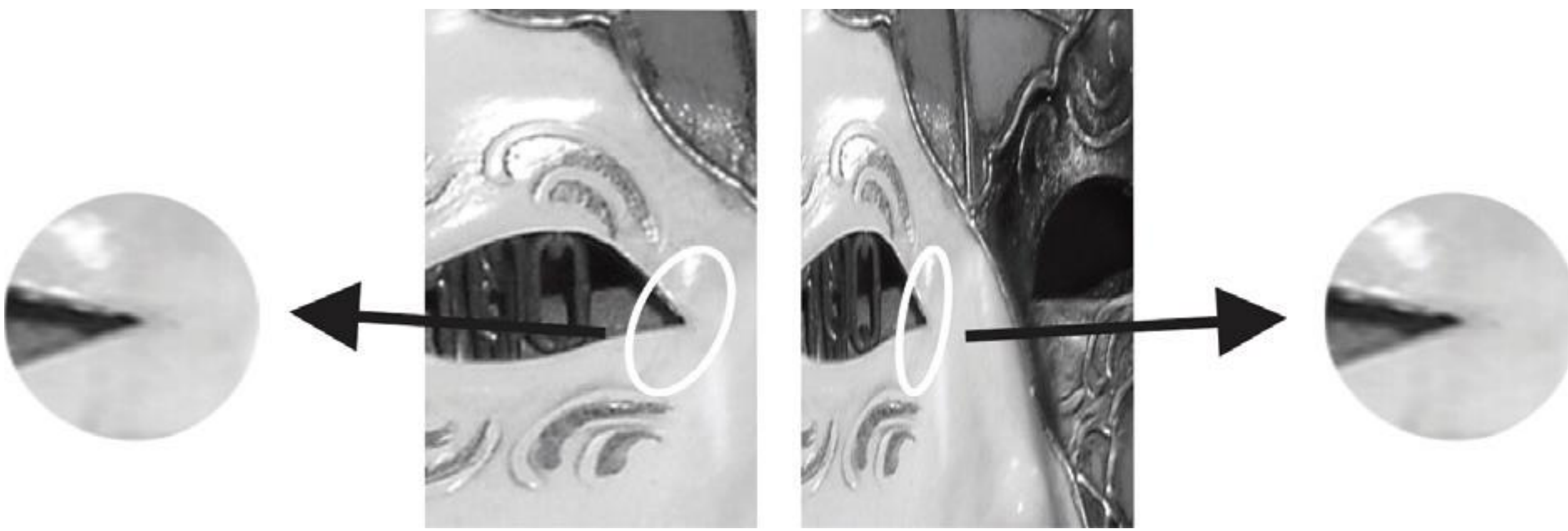


$$
\begin{aligned}
I = \; & (1-x) && (1-y) && I_{00} && + \\
& (1-x) && y && I_{01} && + \\
& x && (1-y) && I_{10} && + \\
& x && y && I_{11} &&
\end{aligned}
$$

How to achieve invariance

Step 2: Affine Un-warping (to achieve slight view-point invariance)

- The second moment matrix *M* can be used to identify the two directions of fastest and slowest change of intensity around the feature.
- Out of these two directions, an elliptic patch is extracted at the scale computed by with the LoG operator.
- The region inside the ellipse is normalized to a circular one



How to achieve invariance

Example: de-rotation, re-scaling, and affine un-warping

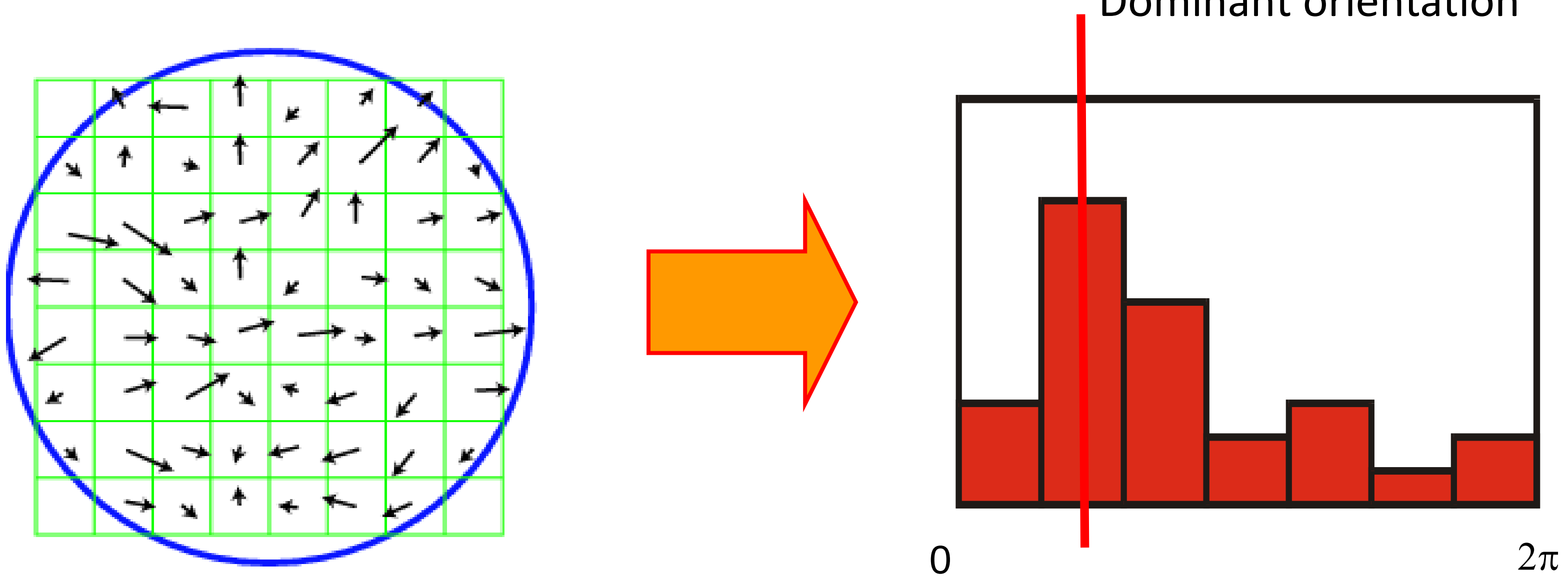
Feature descriptors

- Disadvantage of patches as descriptors:
  - Very small errors in rotation, scale, view-point, and illumination can affect matching score significantly
  - Computationally expensive (need to unwarp every patch)

- Better solution **today**: build descriptors from Histograms of Oriented Gradients (HOGs)

HOG descriptor (Histogram of Oriented Gradients)

- Compute a histogram of orientations of intensity gradients
- Peaks in histogram: dominant orientations
- **Keypoint orientation = histogram peak**
  - If there are multiple candidate peaks, **construct a different keypoint for each such orientation**
- **Rotate patch** according to this angle
  - This puts the patches into a canonical orientation

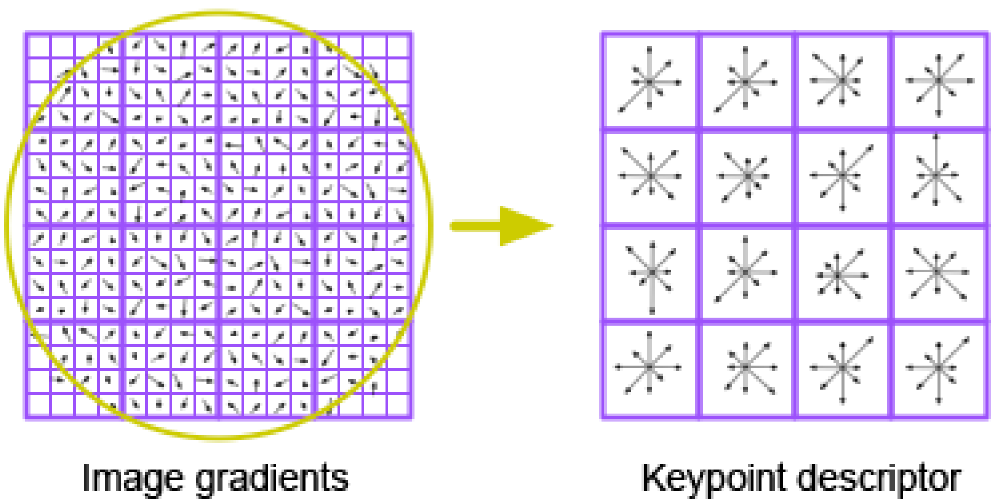


Rotation and Scale Normalization

- Rotate the window to standard orientation
- Scale the window size based on the scale at which the point was found.

SIFT descriptor

- **S**cale **I**nvariant **F**eature **T**ransform
- Invented by David Lowe [IJCV, 2004]
- Descriptor computation:
  - Divide patch into 4x4 sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
  - Resulting SIFT descriptor: 4x4x8 = 128 values
  - Descriptor Matching: Euclidean-distance between these descriptor vectors (i.e., SSD)



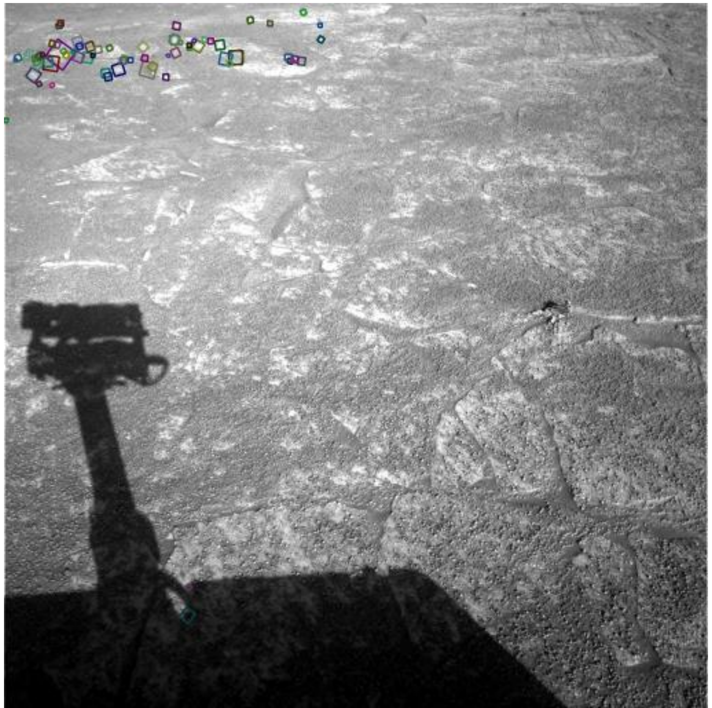
David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* , 2004.

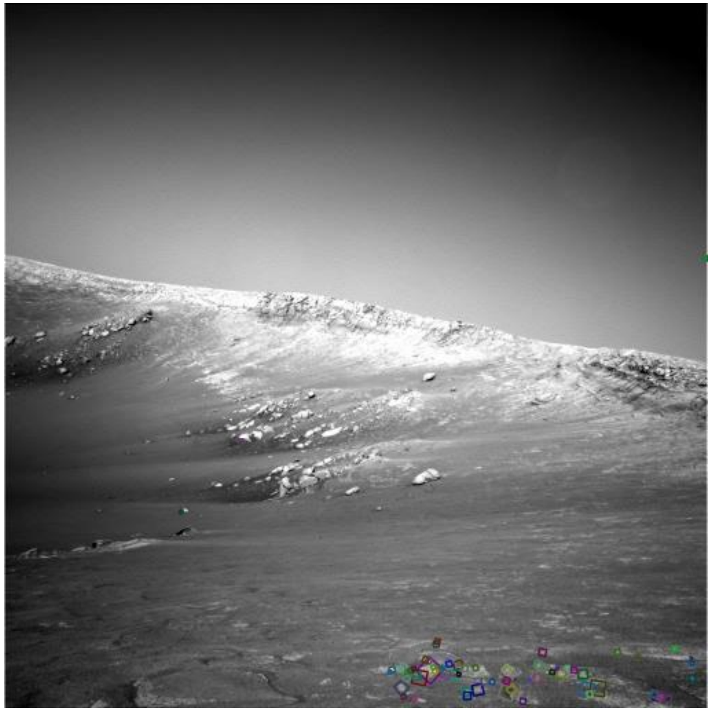
Intensity Normalization

- The descriptor values are normalized such that the L2 norm is 1. This guarantees that the descriptor is invariant to linear illumination changes.

Feature descriptors: SIFT

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night (below)
  - Fast and efficient—can run in real time
  - Original SIFT code (binary files): http://people.cs.ubc.ca/~lowe/keypoints

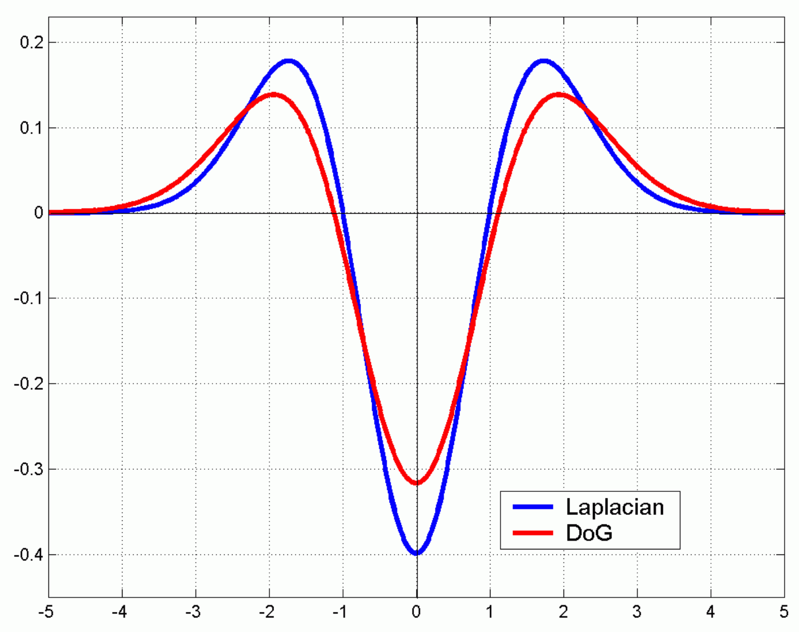




Scale Invariant Detection

Like to Harris Laplacian but Laplacian of Gaussian kernel is approximated with Difference of Gaussian (DoG) kernel (computationally cheaper):

$$LOG \approx DoG = G_{k\sigma}(x, y) - G_{\sigma}(x, y)$$

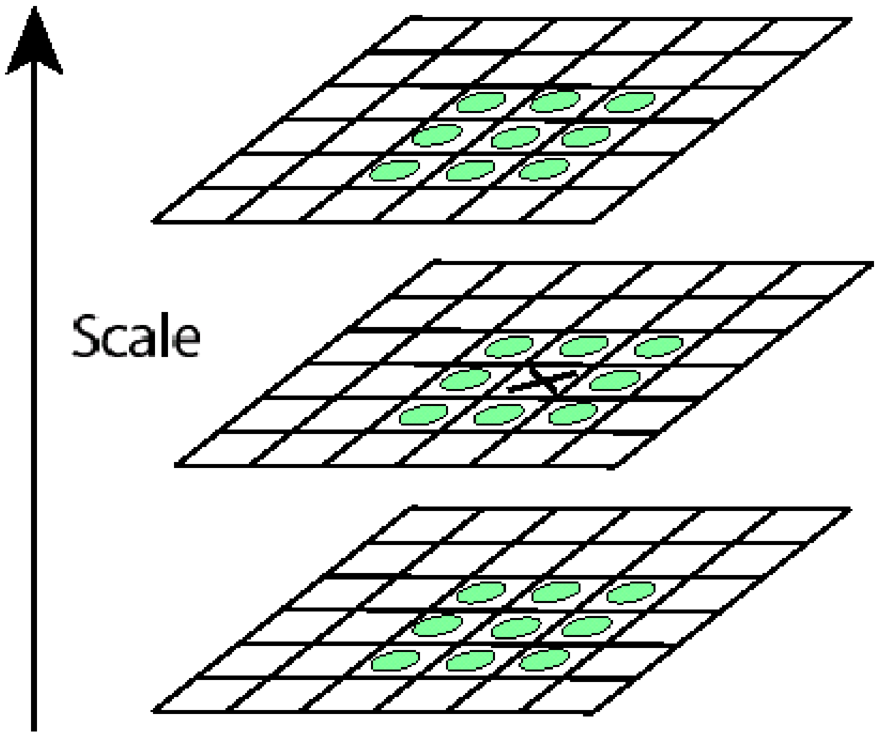


SIFT detector (location + scale)

SIFT keypoints: local extrema in both location and scale of the DoG

- Detect maxima and minima of difference-of-Gaussian in scale space

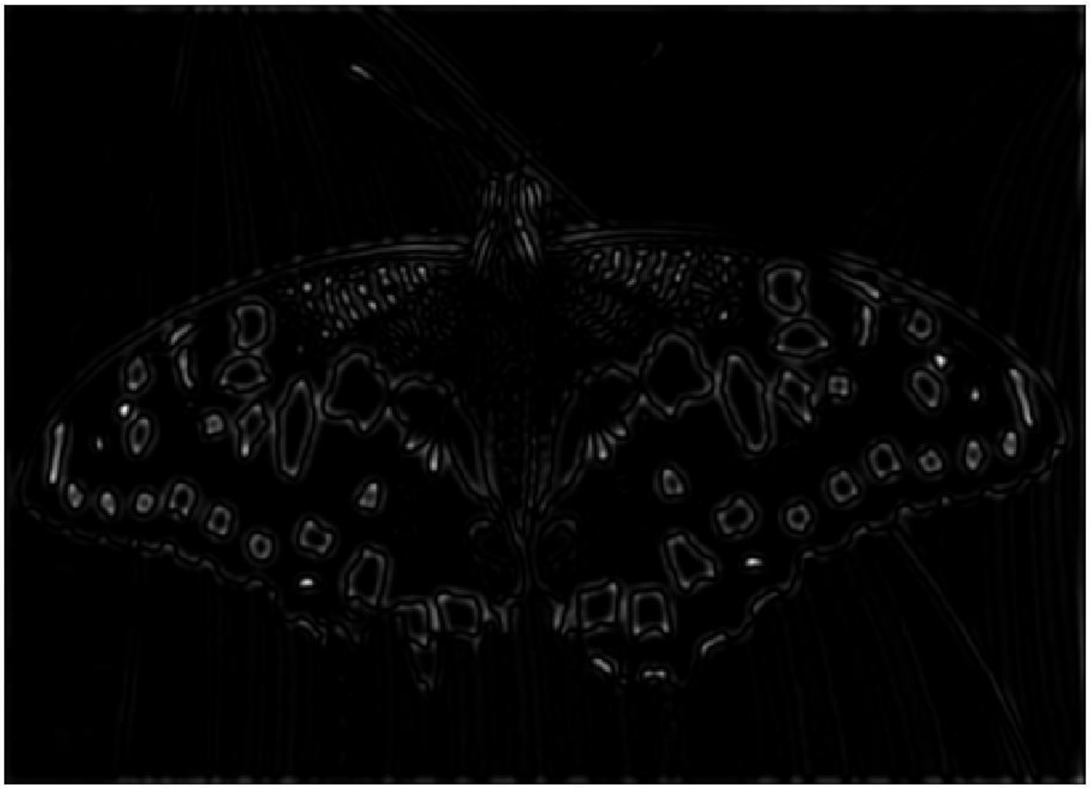
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below



For each max or min found, output is the **location** and the **scale**.

Scale-space detection: Example

Scale-space detection: Example

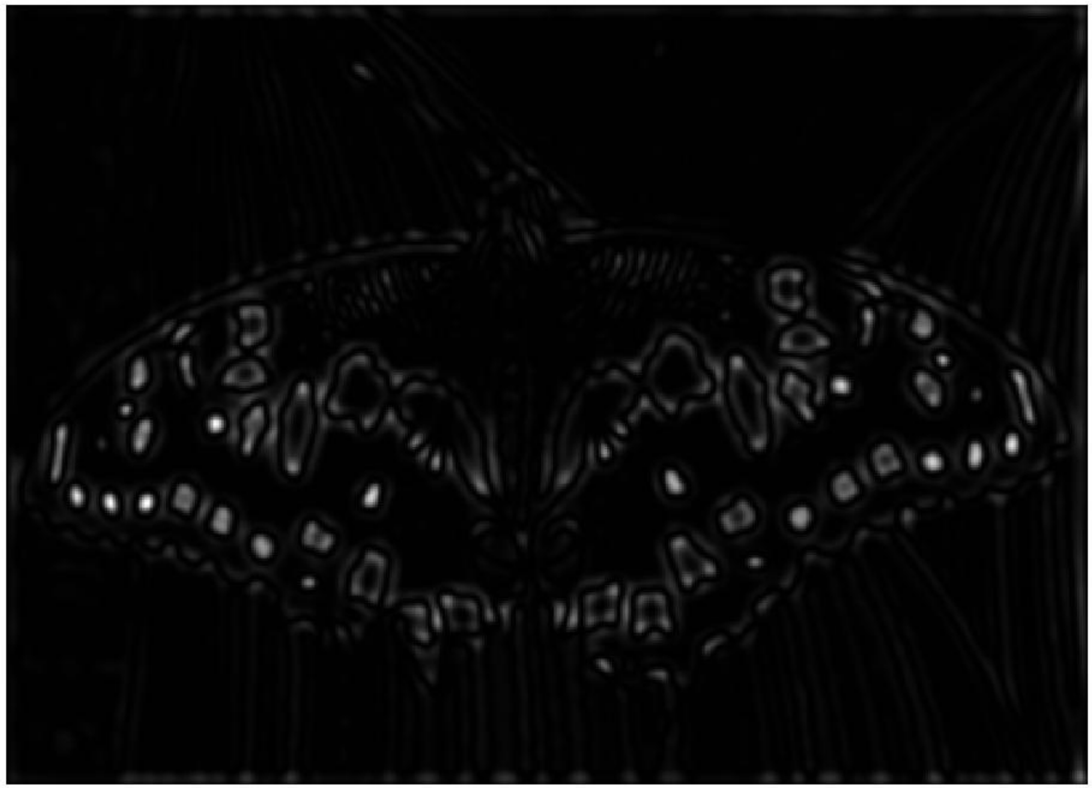


sigma = 2

Scale-space detection: Example

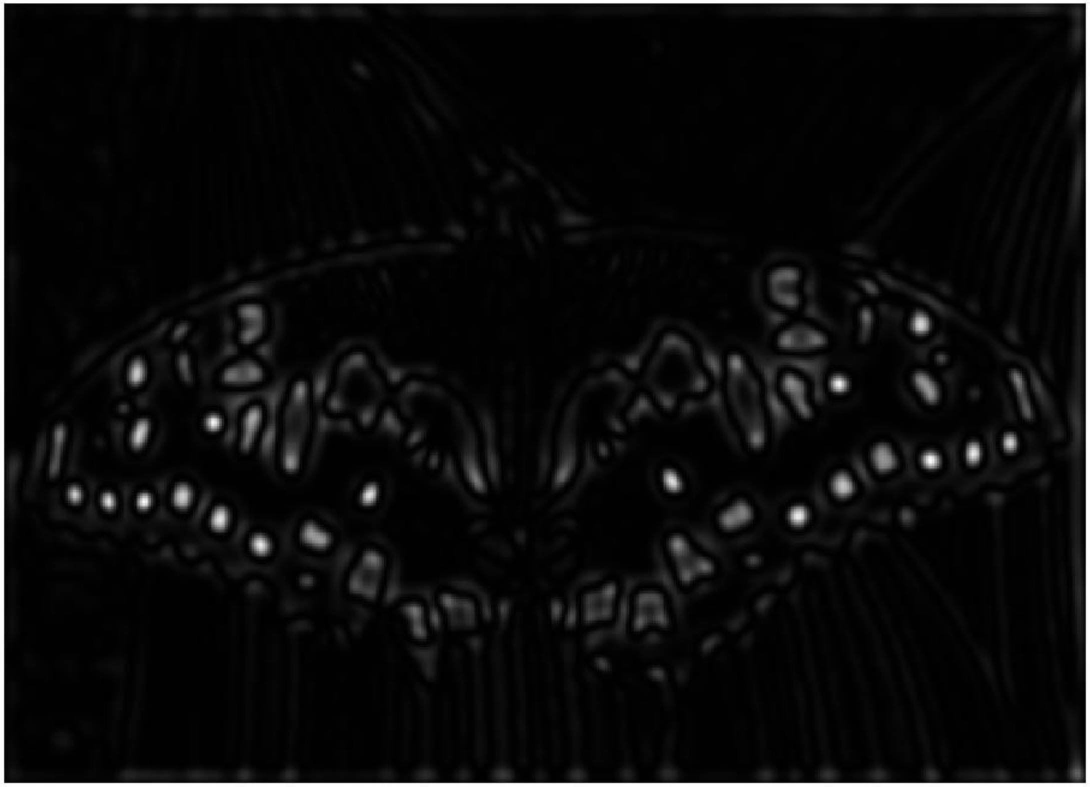
sigma = 2.5018

Scale-space detection: Example



sigma = 3.1296

Scale-space detection: Example



sigma = 3.9149

Scale-space detection: Example

sigma = 4.8972

Scale-space detection: Example

sigma = 6.126

Scale-space detection: Example

sigma = 7.6631

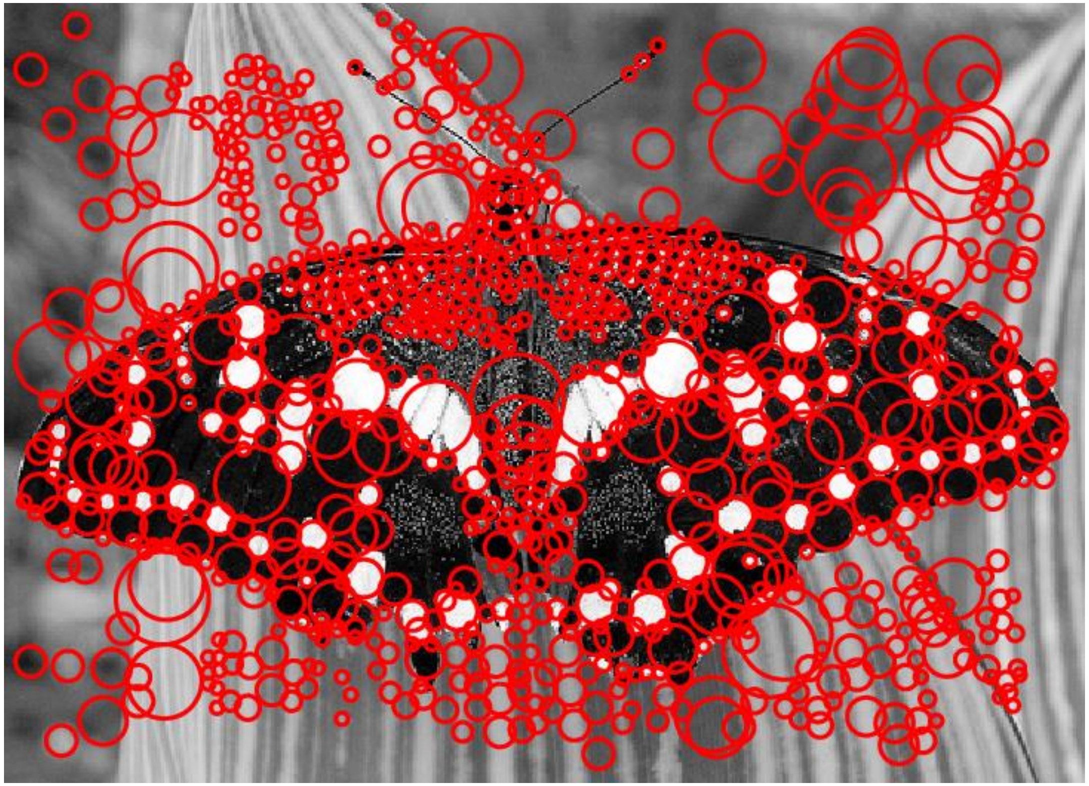
Scale-space detection: Example

sigma = 9.5859

Scale-space detection: Example

sigma = 11.9912

Scale-space detection: Example

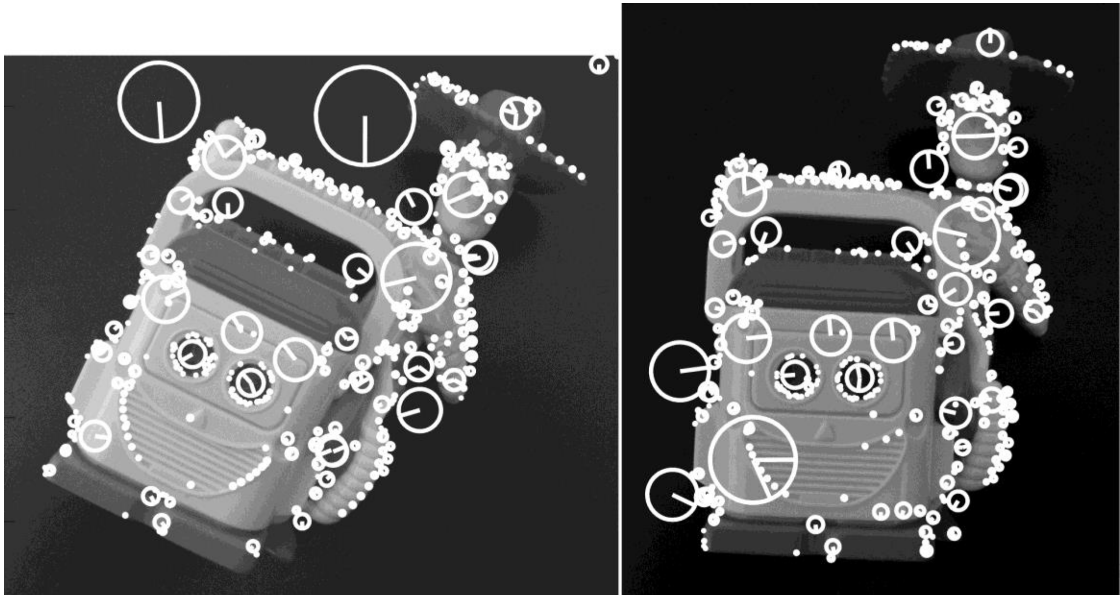


SIFT Features: Summary

- **SIFT: Scale Invariant Feature Transform** [Lowe, IJCV 2004]
- An approach to **detect and describe** regions of interest in an image.

- SIFT features are **reasonably invariant** to changes in **rotation, scaling**, and **small changes in viewpoint and illumination**

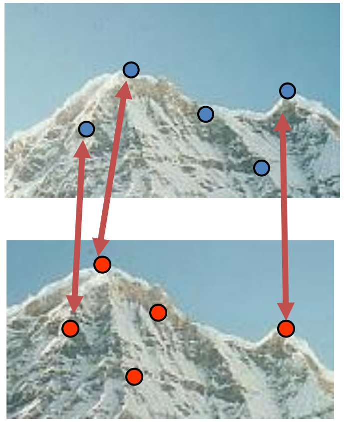
- **Real-time but still slow (10 Hz on an i7 laptop)**
  - Expensive steps are the scale detection and descriptor extraction

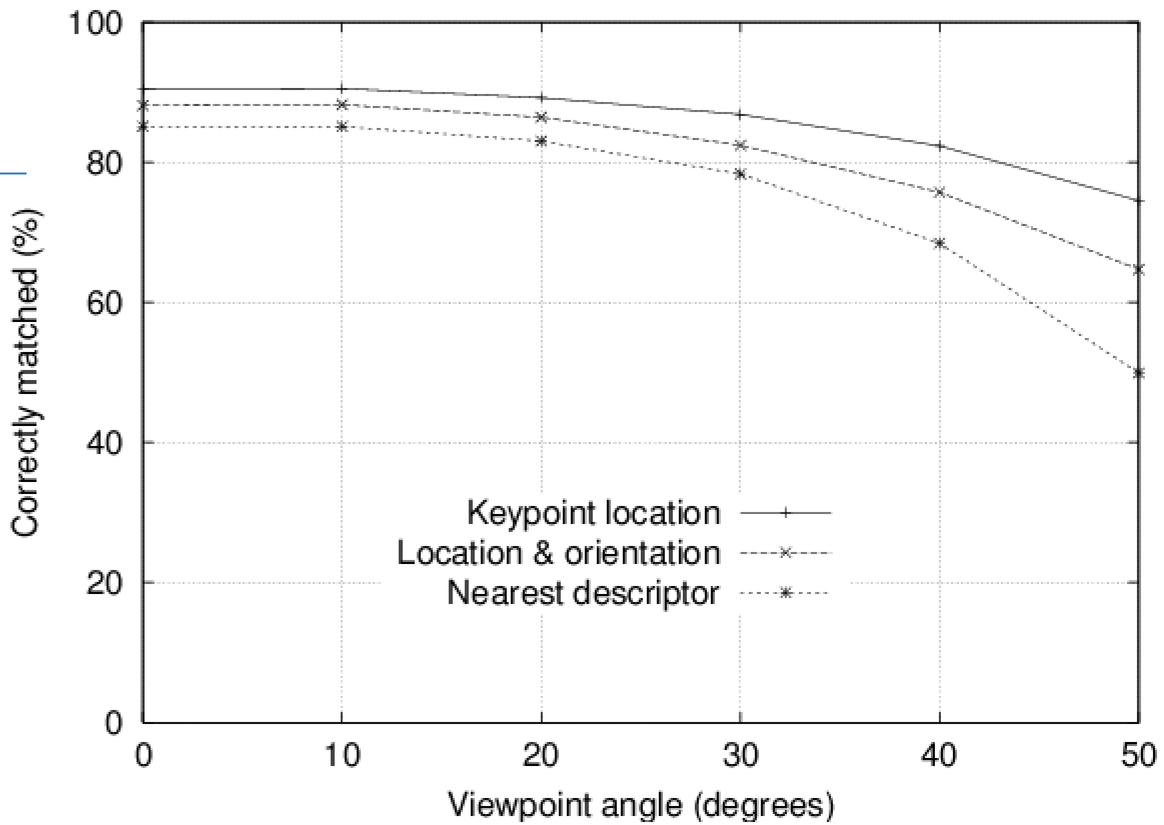


SIFT repeatability vs. viewpoint angle

Repeatability=

$$\frac{\text{\# correspondences detected}}{\text{\# correspondences present}}$$

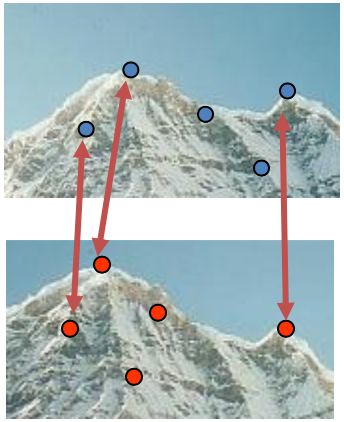


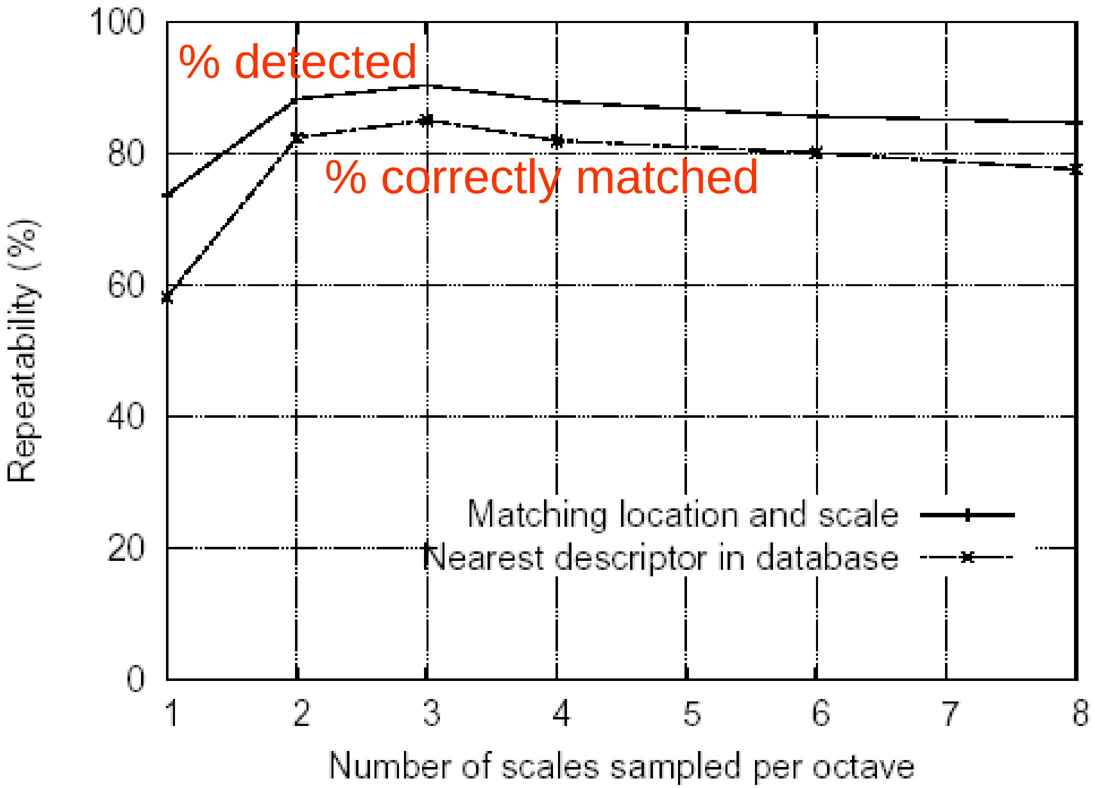


SIFT repeatability vs. Scale

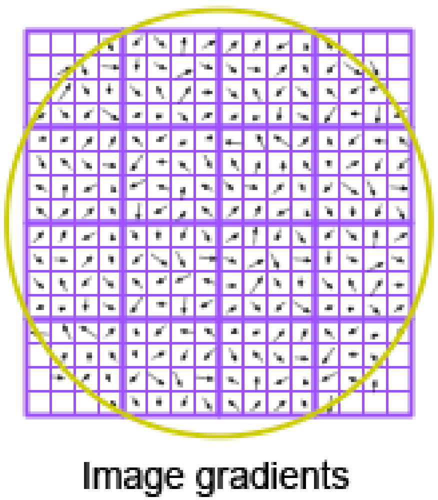
Repeatability=

$$\frac{\#\ \text{correspondences detected}}{\#\ \text{correspondences present}}$$





Influence of Number of Orientations and Sub-patches



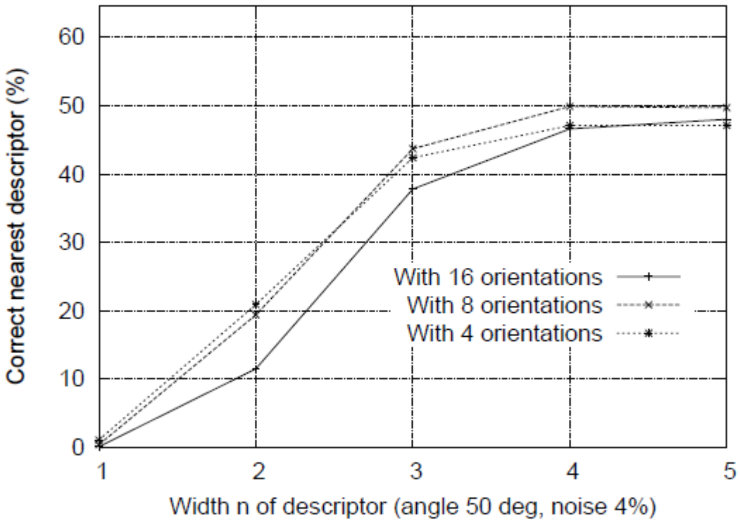
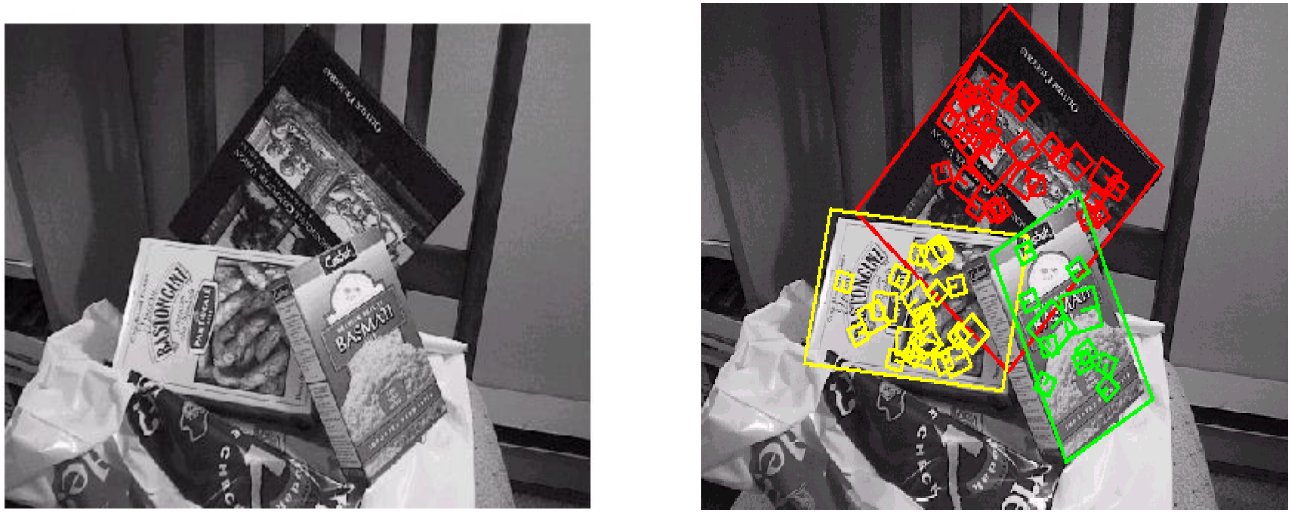


Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affine viewpoint change of 50 degrees and addition of 4% noise.

How many parameters are used to define a SIFT feature?

- Descriptor: 128 parameters
- Location (pixel coordinates of the center of the patch): 2D vector
- Scale (i.e., size) of the patch: 1 scalar value
- Orientation (i.e., angle of the patch): 1 scalar value

SIFT for Object recognition



SIFT for Panorama Stitching

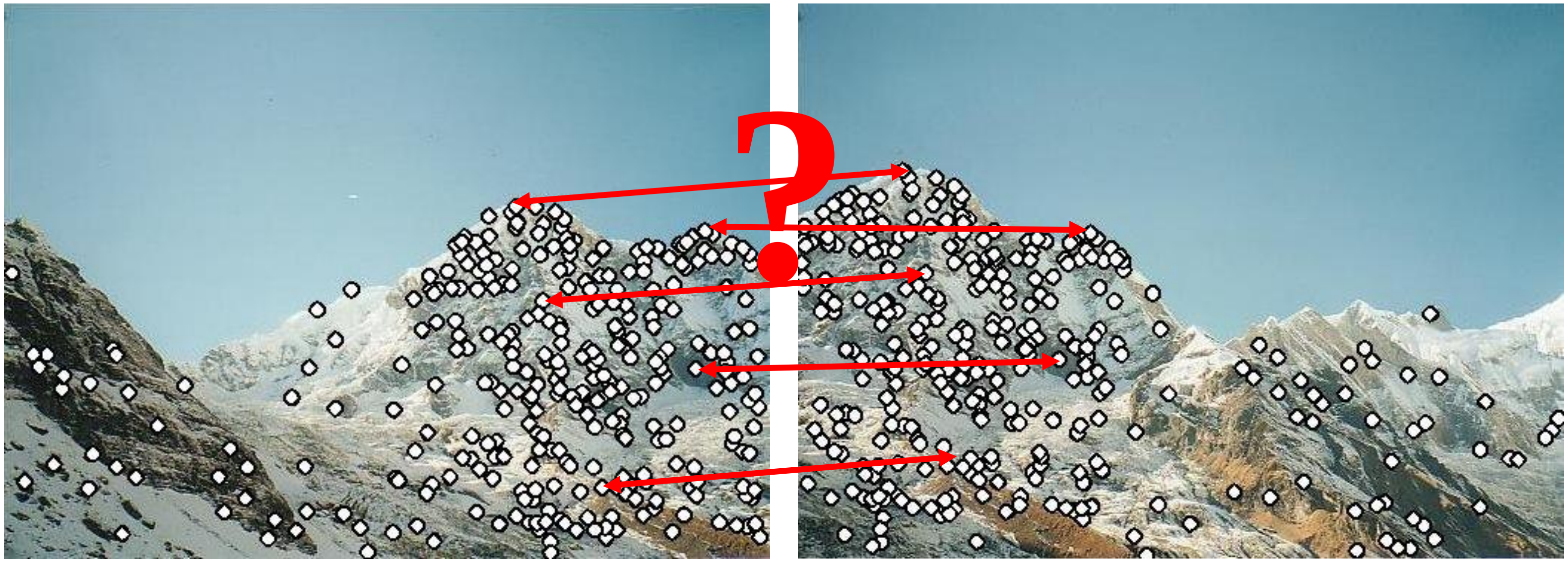
AutoStitch: http://matthewalunbrown.com/autostitch/autostitch.html

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

Main questions

- Where will the interest points come from?
  - What are salient features that we'll *detect* in multiple views?
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?

Feature matching



Feature matching

- Given a feature in I_1, how to find the best match in I_2?
  1. Define distance function that compares two descriptors
     - SSD
     - SAD
     - NCC
  2. Brute-force matching: Test all the features in I_2, find the one with min distance

- Problem with distance: can give good scores to very ambiguous (bad) matches!
- Better approach: ratio distance = $d(f_1, f_2) / d(f_1, f_2') <$ Threshold (e.g., 0.8)
  - f_2 is best match of f_1 in I_2
  - f_2' is 2^{nd} best match of f_1 in I_2
  - gives small values for ambiguous matches

SIFT Feature matching: ratio distance

The inventor of the SIFT recommends to use a threshold on 0.8. Where does this come from?

"A threshold of 0.8, eliminates 90% of the false matches while discarding less than 5% of the correct matches."

"This figure was generated by matching images following random scale and orientation change, a depth rotation of 30 degrees, and addition of 2% image noise, against a database of 40,000 keypoints."

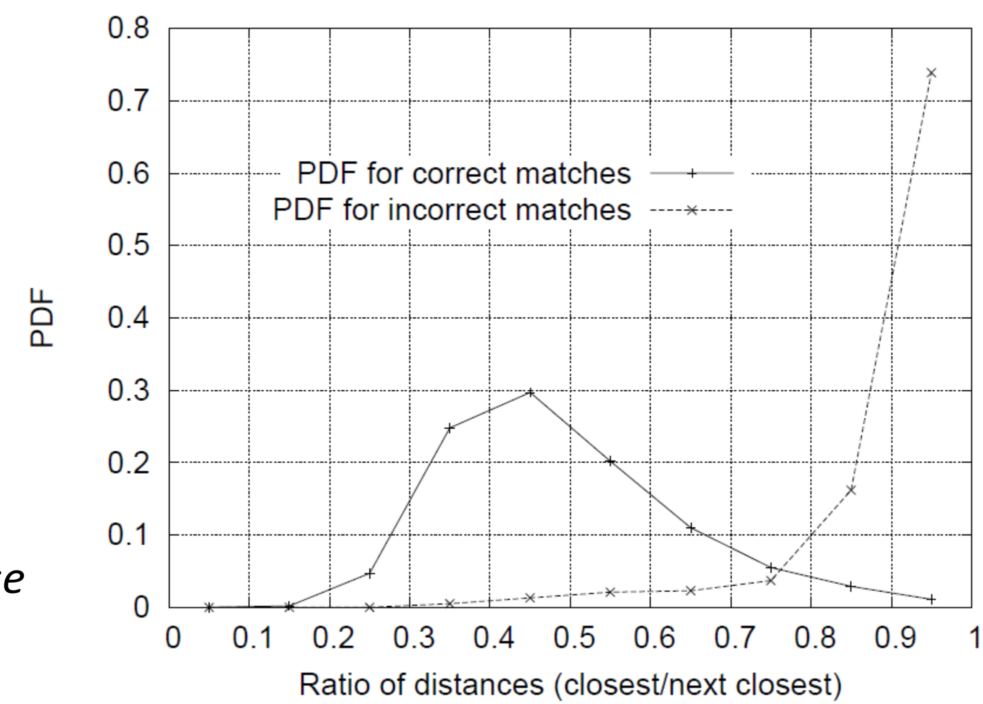


Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.

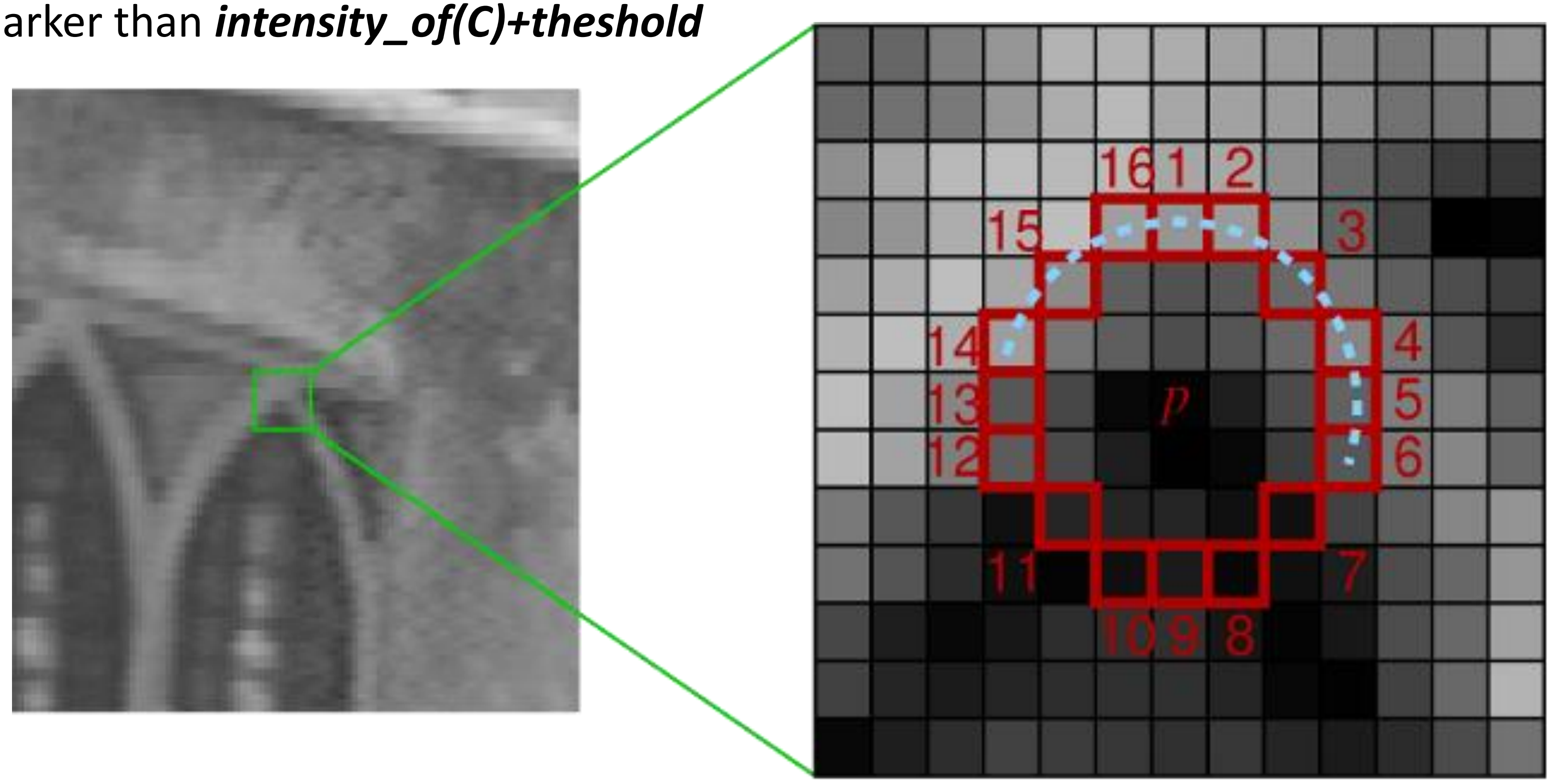
SURF [Bay et al., ECCV 2006]

- ***S***peeded ***U***p ***R***obust ***F***eatures
- Based on ideas similar to **SIFT**
- Approximated computation for detection and descriptor
- Results comparable with SIFT, plus:
  - Faster computation
  - Generally shorter descriptors

Bay, Tuytelaars, Van Gool, " Speeded Up Robust Features ", ECCV 2006

FAST detector [Rosten et al., ECCV'05]

- **FAST**: **F**eatures from **A**ccelerated **S**egment **T**est
- Studies intensity of pixels on circle around candidate pixel ***C***
- ***C*** is a FAST corner **if** a set of ***N*** contiguous pixels on circle are:
  - all brighter than ***intensity_of(C)+theshold***, or
  - all darker than ***intensity_of(C)+theshold***



- Typical FAST mask: test for **9** contiguous pixels in a **16**-pixel circle
- **Very fast detector** - in the order of 100 Mega-pixel/second

Rosten, Drummond, Fusing points and lines for high performance tracking, IEEE International Conference on Computer Vision, 2005

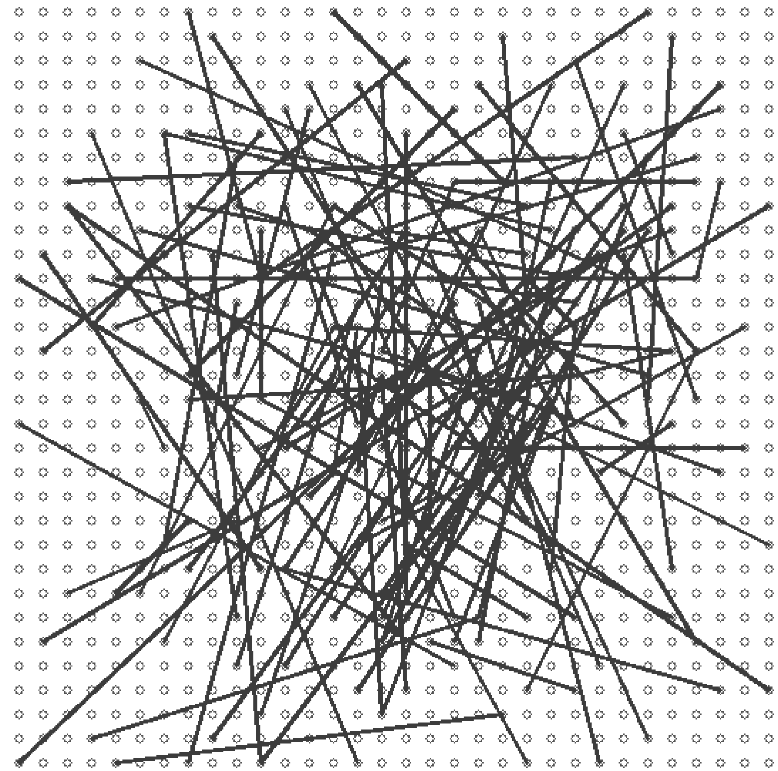
BRIEF descriptor [Calonder et. al, ECCV 2010]

- **B**inary **R**obust **I**ndependent **E**lementary **F**eatures
- Goal: high speed (in description and matching)

- **Binary** descriptor formation:
  - Smooth image
  - **for each** detected keypoint (e.g. FAST),
  - ***sample*** 256 intensity pairs **p**=(p_1, p_2) within a squared patch around the keypoint
  - **for each pair p**
    - **if** $I_{p_1} < I_{p_2}$ **then *set*** bit **p** of descriptor to **1**
    - **else *set*** bit **p** of descriptor to **0**

- The pattern is generated randomly only once; then, the same pattern is used for all patches

- Not scale/rotation invariant
- Allows **very fast** Hamming Distance matching: count the number of bits that are different in the descriptors matched

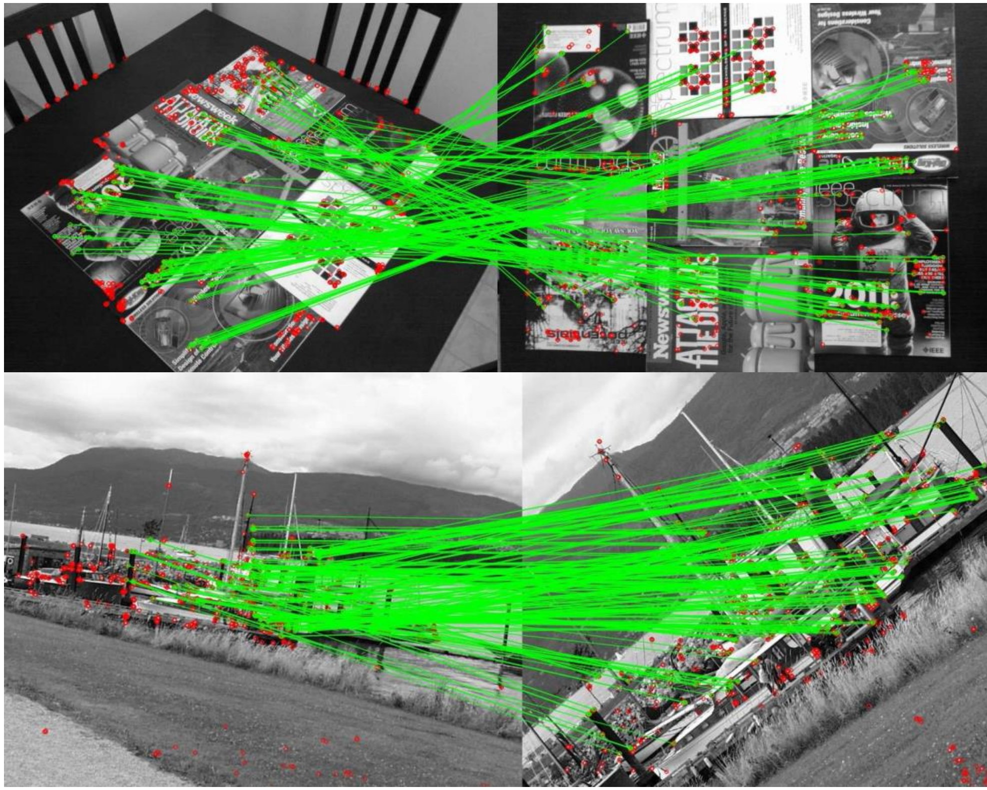


Pattern for intensity pair samples – generated randomly

Calonder, Lepetit, Strecha, Fua, BRIEF: Binary Robust Independent Elementary Features, ECCV'10]

ORB descriptor [Rublee et al., ICCV 2011]

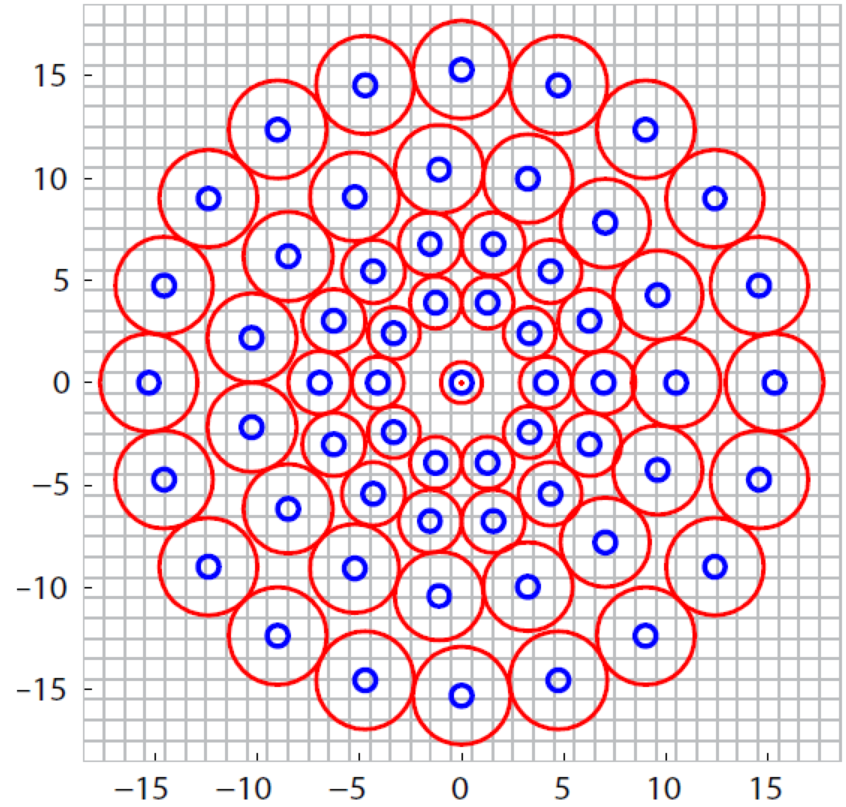
- ***O***riented FAST and ***R***otated ***B***RIEF
- Keypoint detector based on **FAST**
- **BRIEF** descriptors are *steered* according to keypoint orientation (to provide rotation invariance)
- Good Binary features are learned by minimizing the correlation on a set of training patches.



BRISK descriptor [Leutenegger, Chli, Siegwart, ICCV 2011]

- **B**inary **R**obust **I**nvariant **S**calable **K**eypoints
- Detect corners in scale-space using FAST
- Rotation and scale invariant

- **Binary**, formed by pairwise intensity comparisons (like BRIEF)
- **Pattern** defines intensity comparisons in the keypoint neighborhood
- **Red circles**: size of the smoothing kernel applied
- **Blue circles**: smoothed pixel value used
- Compare short- and long-distance pairs for orientation assignment & descriptor formation
- Detection and descriptor speed: ~10 times faster than SURF
- Slower than BRIEF, but scale- and rotation- invariant



Recap Table

Detector	Descriptor	Localization Accuracy	Relocalization & Loop closing	Efficiency
Harris	Patch	++++	+	+++
Shi-Tomasi	Patch	++++	+	+++
SIFT	SIFT	+++	++++	+
SURF	SURF	+++	++++	++
FAST	BRIEF ORB BRISK	++++	+++	++++