ROBOTICS & PERCEPTION GROUP

Universität Zürich[UZH]

Lecture 05
Feature Extraction 1

Prof. Dr. Davide Scaramuzza

sdavide@ifi.uzh.ch

REMINDER: Lab Exercise 2 - Today

- At 14:15 in room 2.A.01
- Harris corner detector
- Download: http://rpg.ifi.uzh.ch/docs/teaching/ex02_harris.zip

Exercise 2: Harris Corner Detection

1 Harris Corner Detection Summary

A corner in an image can be defined as the intersection of two or more edges. Corners are features with high repeatability.

1.1 The basic concept of corner detection

One of the earliest corner detectors was invented by Moravec [3]. He defined a corner as a point where there is a large intensity variation in every direction. An intuitive explanation of his corner detection algorithm is given in Figure 1. Intuitively, one could recognize a corner by looking through a small window centered on the pixel. If the pixel lies in a "flat" region (i.e., a region of uniform intensity), then the adjacent windows will look similar. If the pixel is along an edge, then adjacent windows in the direction perpendicular to the edge will look different, but adjacent widows in a direction parallel to the edge will result only in a small change. Finally, if the pixel lies on a corner, then none of the adjacent windows will look similar. Moravec used the Sum of Squared Differences (SSD) as a measure of the similarity between two patches. A low SSD score indicates more similarity. If this number is locally maximal, then a corner is present.

1.2 The Harris corner detector

Harris and Stephens [2] improved Moravec's corner detector by considering the partial derivatives of the SSD score instead of using shifted windows.

Let I be a grayscale image. Consdering taking an image patch centered on (x, y) and shifting it by $(\Delta x, \Delta y)$. The Sum of Squared Differences (SSD) between these two patches is given by:

$$SSD(x, y) = \sum_{x,y \in P} (I(x, y) - I(x + \Delta x, y + \Delta y))^2. \quad (1)$$

$I(x + \Delta x, y + \Delta y)$ can be approximated by a first-order Taylor expansion. Let I_x and I_y be the partial derivatives of I, such that

$$I(x + \Delta x, y + \Delta y) = I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y. \quad (2)$$

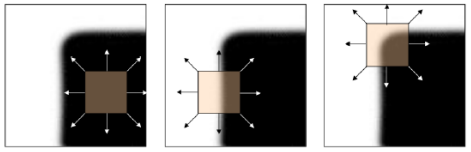


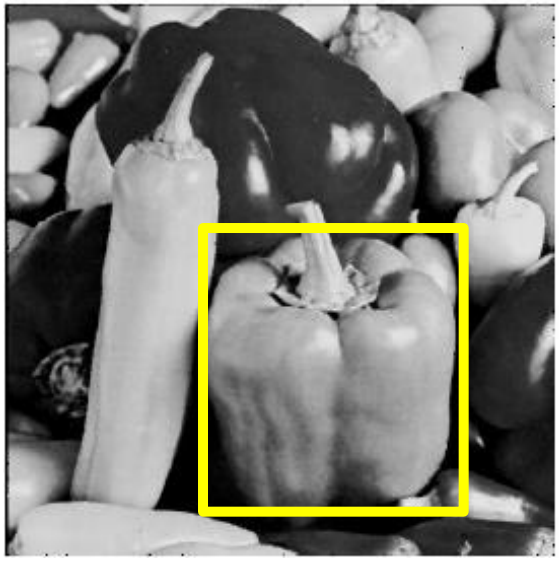
Figure 1: Illustration of flat regions (left), edge regions (middle) and corner region (right).

Outline

- Filters for feature extraction
- Point-feature extraction: today and next lecture
- Line extraction algorithms: next lecture

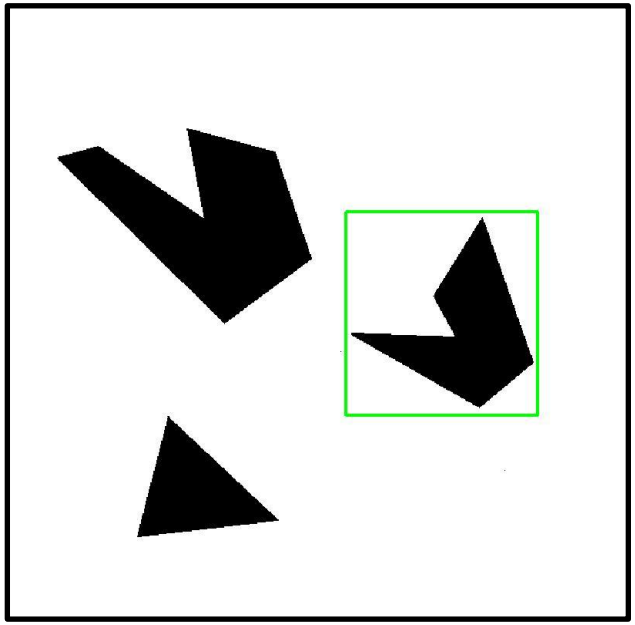
Filters for features

- Previously, thinking of filtering as a way to remove or reduce **noise**
- Now, consider how filters will allow us to abstract higher-level "**features**".
  - Map raw pixels to an intermediate representation that will be used for subsequent processing
  - Goal: reduce amount of data, discard redundancy, preserve what's useful
    - Template matching
    - Edge detection
    - Feature extraction
      - lines
      - points

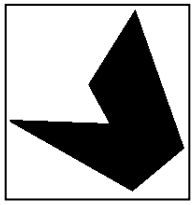


Filters as Templates

- Find locations in an image that are similar to a ***template***
- If we look at filters as **templates**, we can use correlation to detect these locations



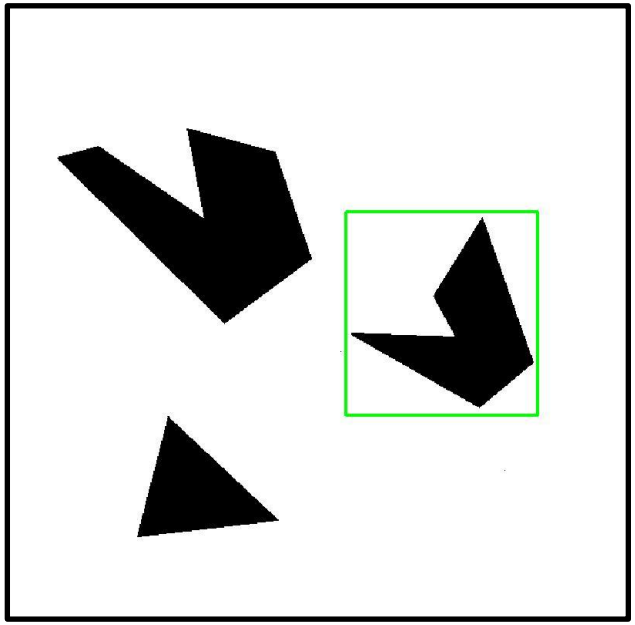
Detected template



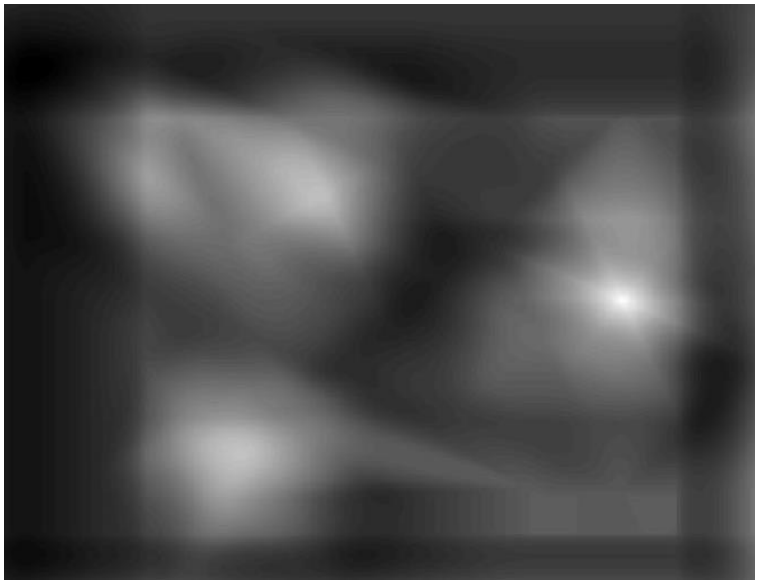
Template

Template matching

- Find locations in an image that are similar to a ***template***
- If we look at filters as **templates**, we can use correlation to detect these locations



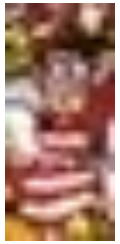
Detected template



Correlation map

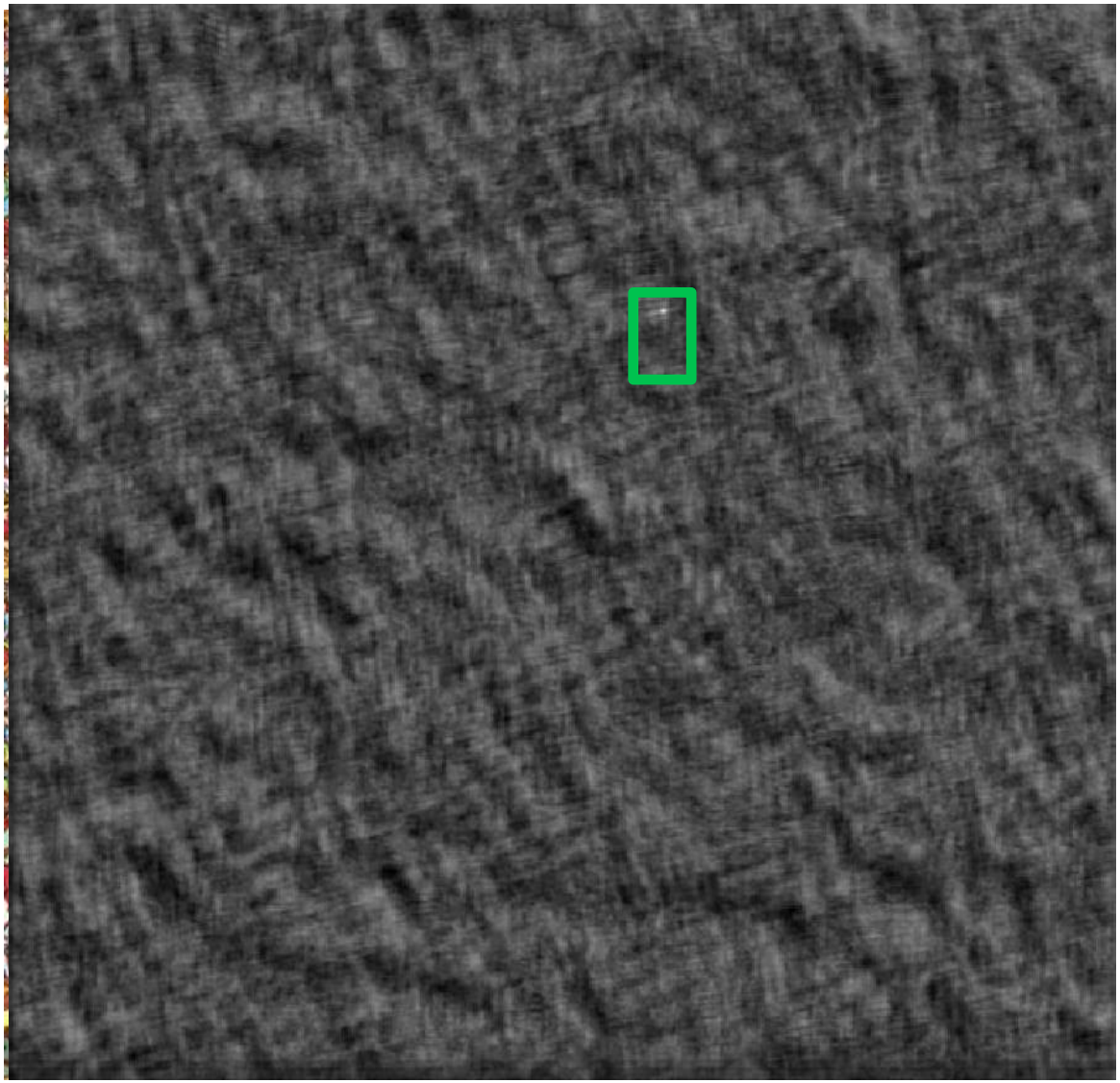
Where’s Waldo?

Scene

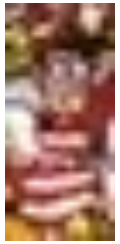


Template

Where’s Waldo?



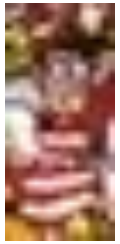
Scene



Template

Where’s Waldo?

Scene



Template

Template matching

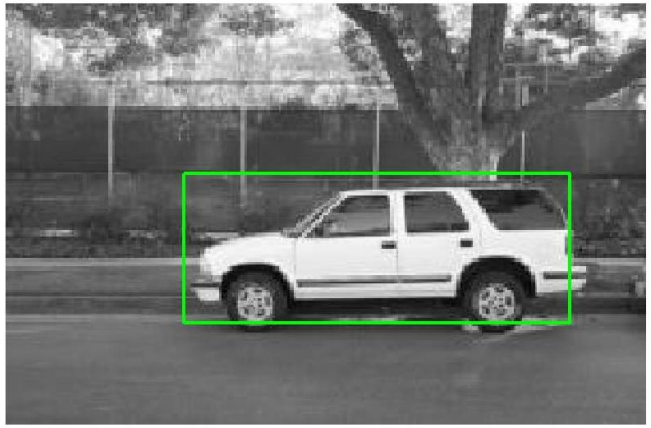
- What if the template is not identical to the object we want to localize?

Scene

Template

Template matching

- What if the template is not identical to the object we want to localize?
- Match can be meaningful if **scale**, **orientation**, **illumination**, and general appearance are right



Detected template

Template

Similarity measures

- Sum of Squared Differences (**SSD**)

$$SSD = \sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(H(u,v)-F(u,v)\right)^2$$

- Sum of Absolute Differences (**SAD**) (used in optical mice)

$$SAD = \sum_{u=-k}^{k}\sum_{v=-k}^{k}\left|H(u,v)-F(u,v)\right|$$

- Zero-mean Normalized Cross Correlation (**ZNCC**)

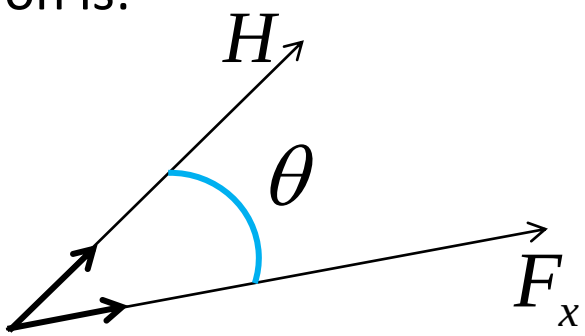
$$ZNCC = \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(H(u,v)-\mu_H\right)\left(F(u,v)-\mu_F\right)}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(H(u,v)-\mu_H\right)^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(F(u,v)-\mu_F\right)^2}}$$

$$\begin{cases}\mu_H = \dfrac{\sum_{u=-k}^{k}\sum_{v=-k}^{k}H(u,v)}{(2k+1)^2} \\ \mu_F = \dfrac{\sum_{u=-k}^{k}\sum_{v=-k}^{k}F(u,v)}{(2k+1)^2}\end{cases}$$

Correlation as an inner product

- Considering the filter H and the portion of the image F as vectors ⇨ their correlation is:

$$\langle H, F \rangle = \|H\| \|F\| \cos\theta$$



- In **ZNCC** we consider the unit vectors of H and F, hence we measure their similarity based on the angle θ. Alternatively, ZNCC maximizes $cos\theta$

$$\cos\theta = \frac{\langle H, F \rangle}{\|H\| \|F\|} = \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(H(u,v)-\mu_H\right)\left(F(u,v)-\mu_F\right)}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(H(u,v)-\mu_H\right)^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(F(u,v)-\mu_F\right)^2}}$$

Summary on filters

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
- Derivatives
  - Opposite signs used to get high response in regions of high contrast
  - Sum to 0 → no response in constant regions
  - High absolute value at points of high contrast
- Filters act as templates
  - Highest response for regions that “look the most like the filter”
  - Dot product as correlation

Outline

- Filters for feature extraction
- Point-feature extraction: today and next lecture
- Line extraction algorithms: next lecture

Example: Point feature extraction and matching for robust place recognition

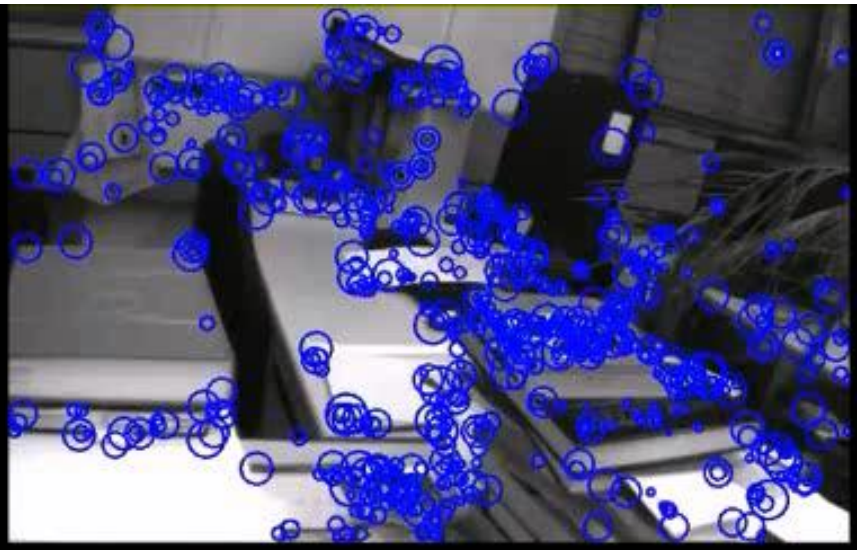
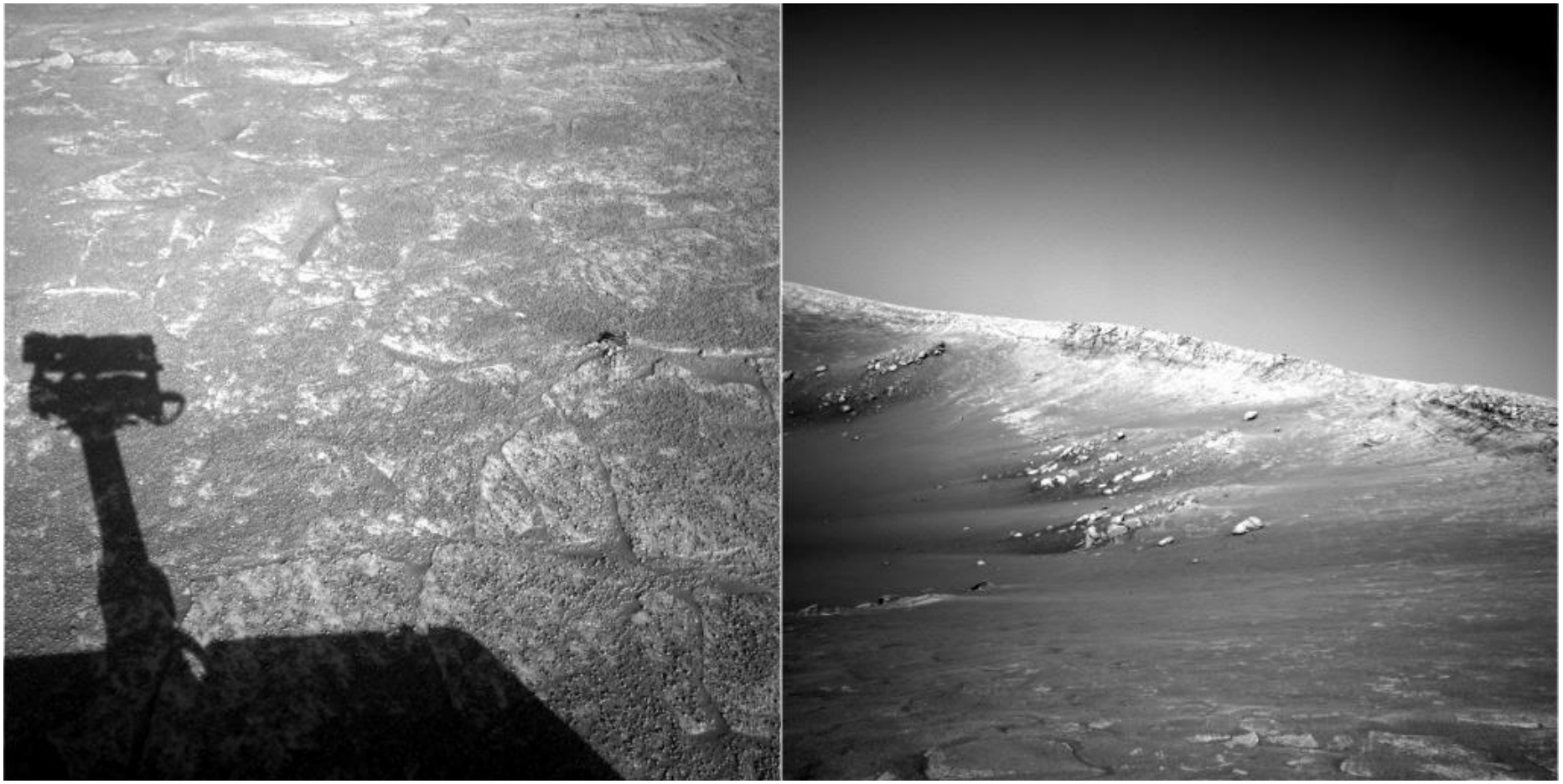


Image matching: why is it hard?

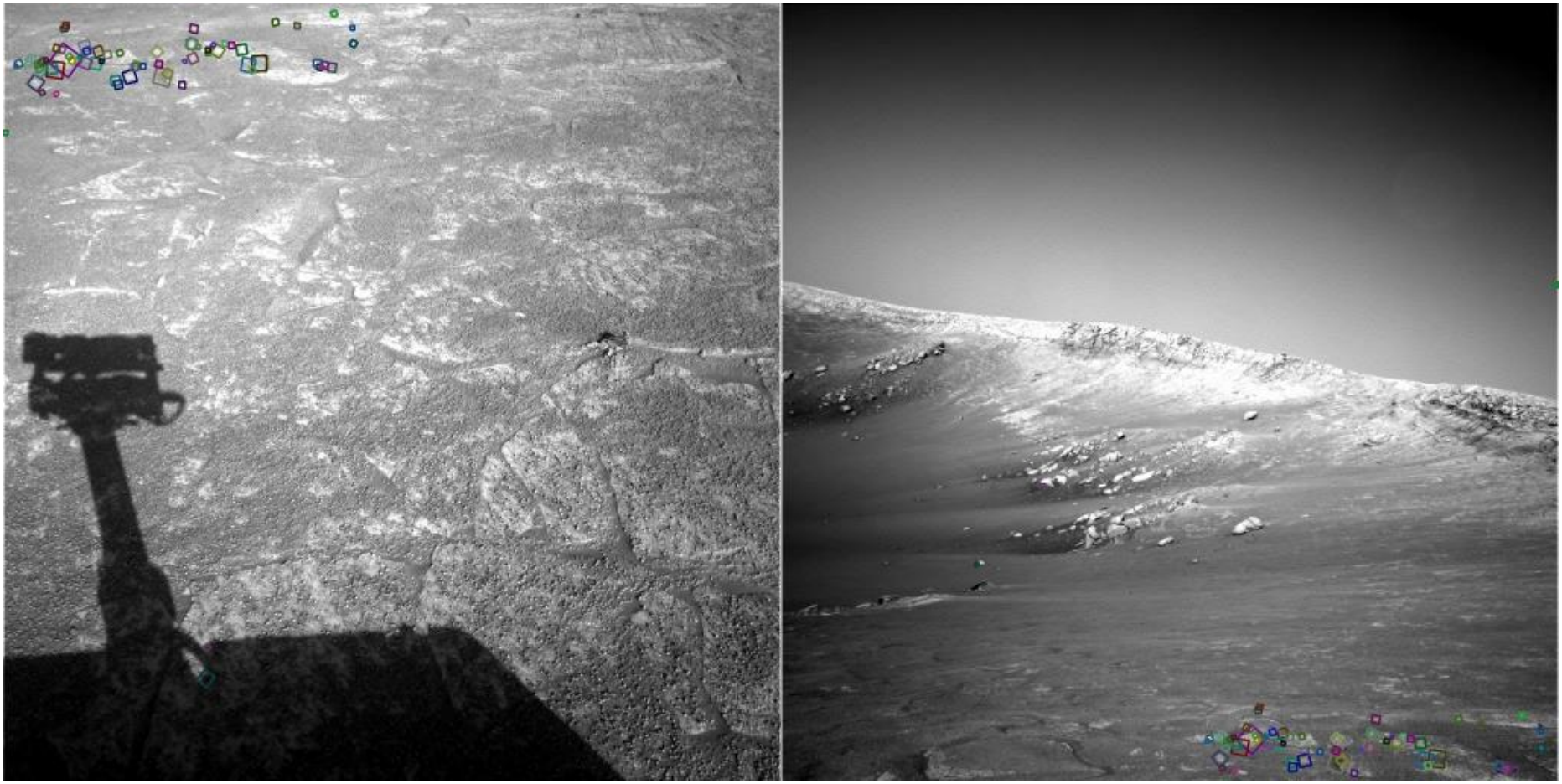
Image matching: why is it hard?



NASA Mars Rover images

Image matching: why is it hard?

Answer below



NASA Mars Rover images with SIFT feature matches

Applications: Build a Panorama

This panorama was generated using **AUTOSTITCH** (freeware)
(Build your own: http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html)

Feature points are used also for:

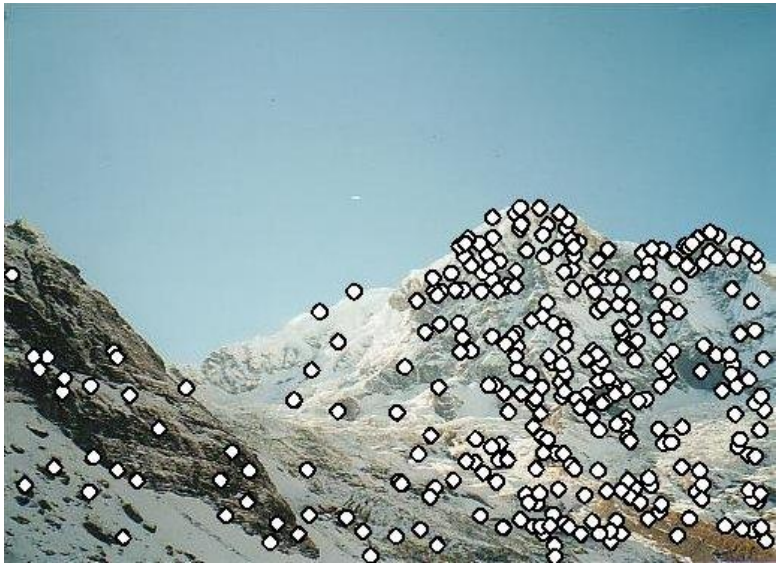
- Robot navigation
- Object recognition
- 3D reconstruction
- Motion estimation (structure from motion)
- Indexing and database retrieval ⇨ Google Images or http://tineye.com
- …

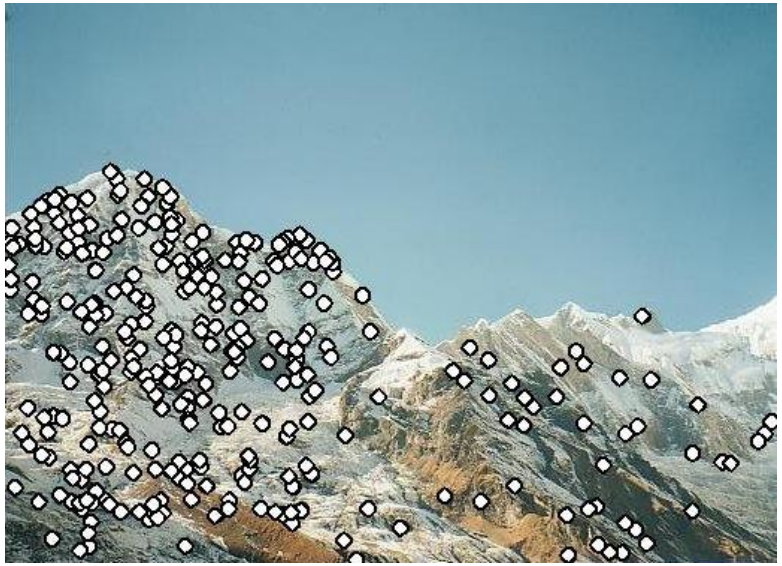
Local features and alignment

- We need to match (align) images
- How would you do it by eye?

Local features and alignment

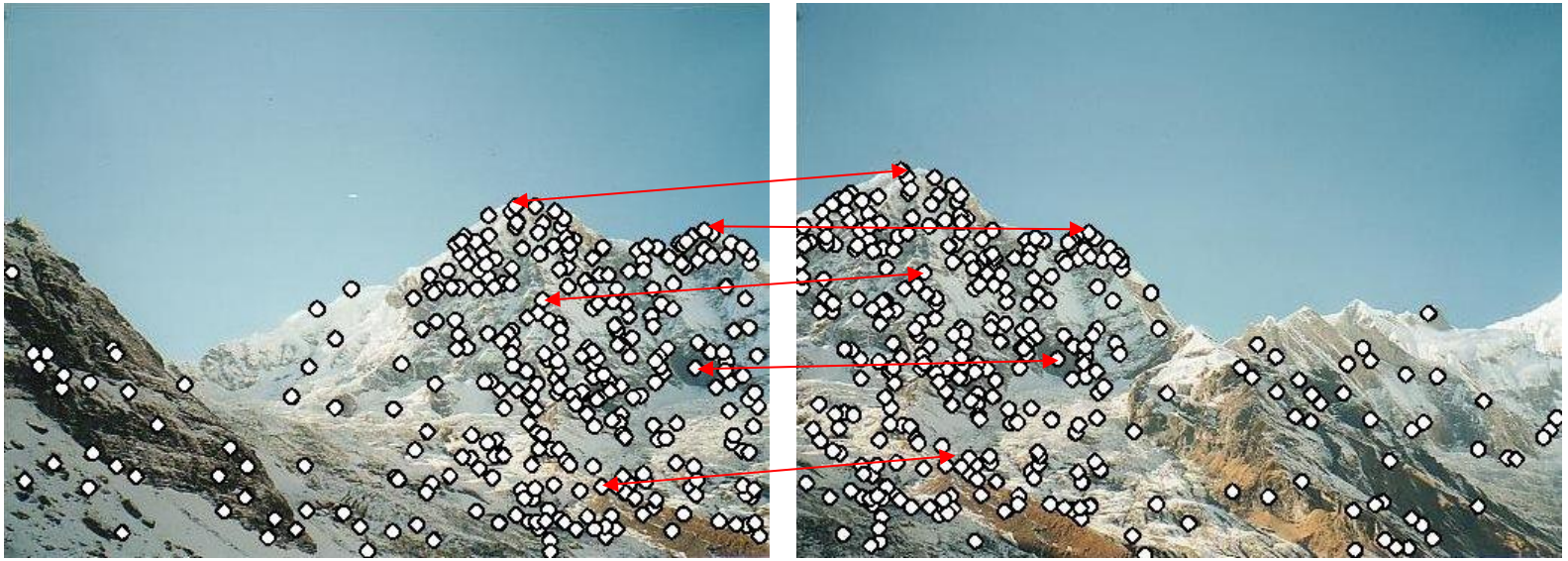
- Detect feature points in both images





Local features and alignment

- Detect feature points in both images
- Find corresponding pairs

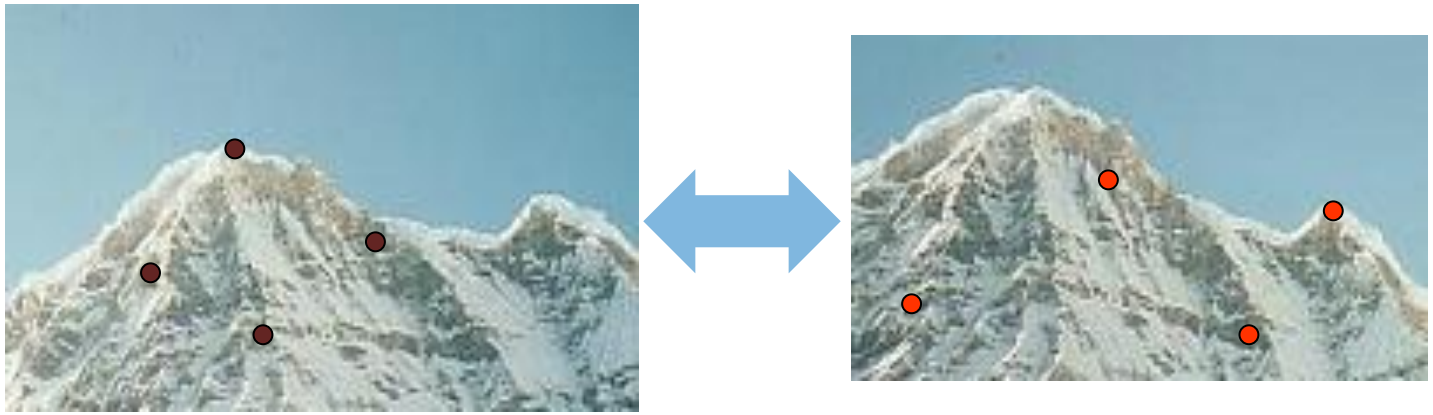


Local features and alignment

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

Matching with Features

- Problem 1:
  - Detect the **same** points **independently** in both images, if they are in the field of view

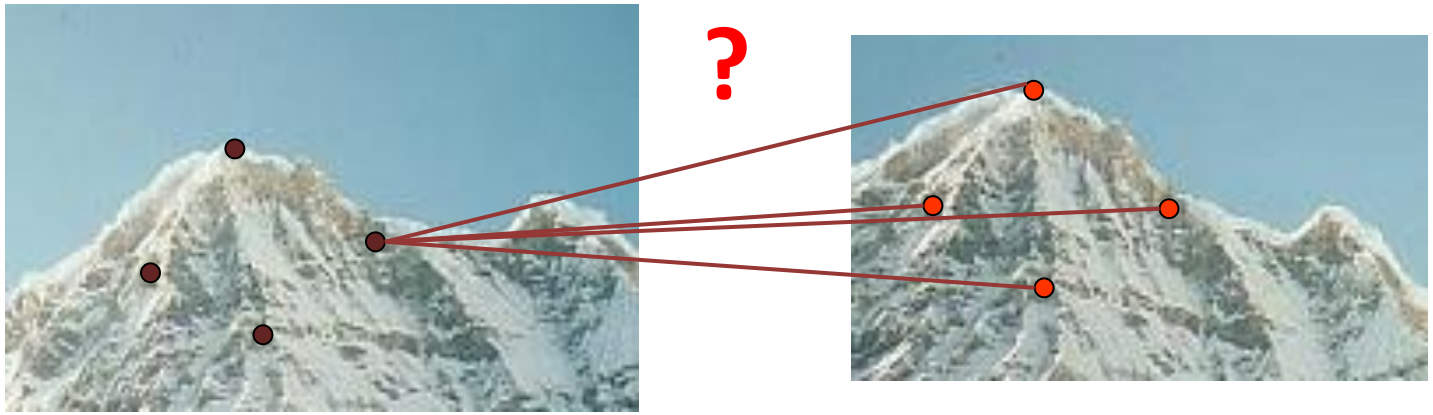


no chance to match!

We need a **repeatable** feature detector

Matching with Features

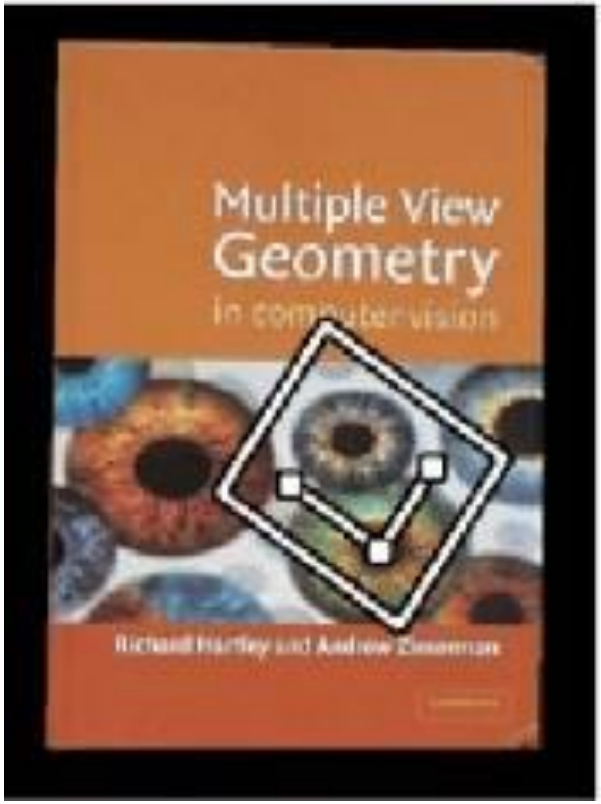
- Problem 2:
  - For each point, identify its correct correspondence in the other image(s)



We need a **reliable** and **distinctive** feature descriptor

Geometric changes

- Rotation
- Scale (i.e., zoom)
- View point (i.e, perspective changes)



Illumination changes

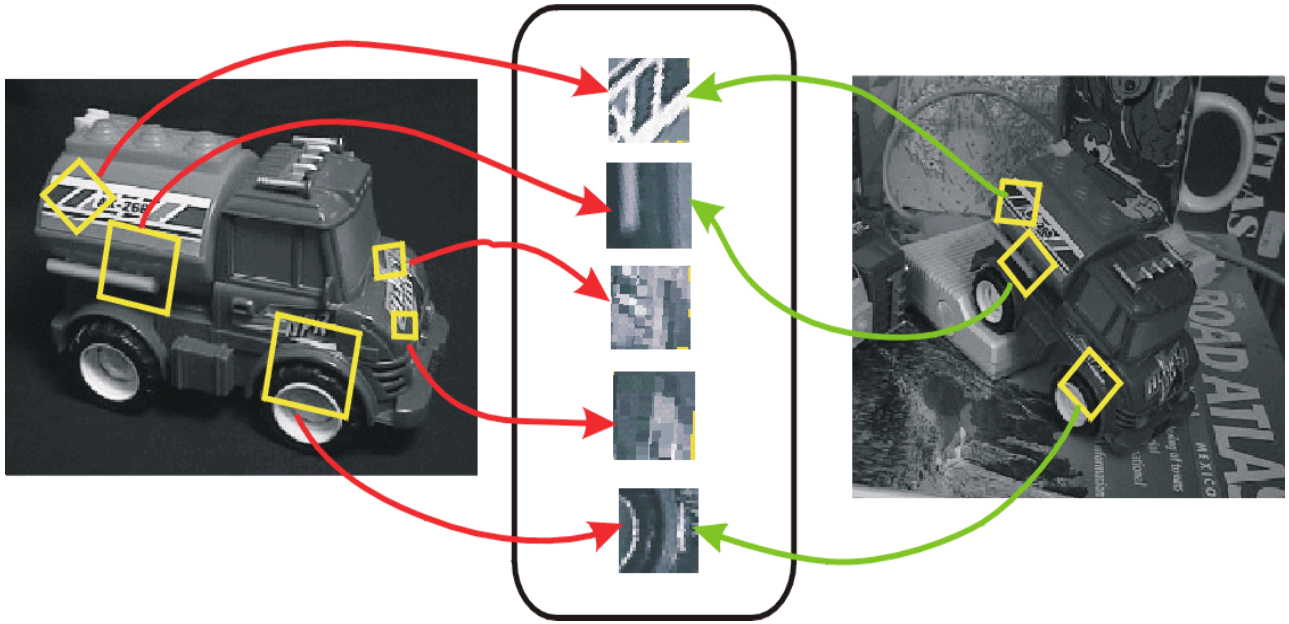
Invariant local features

Subset of local feature types designed to be invariant to common geometric and photometric transformations.

Basic steps:

1) Detect distinctive interest points

2) Extract invariant descriptors

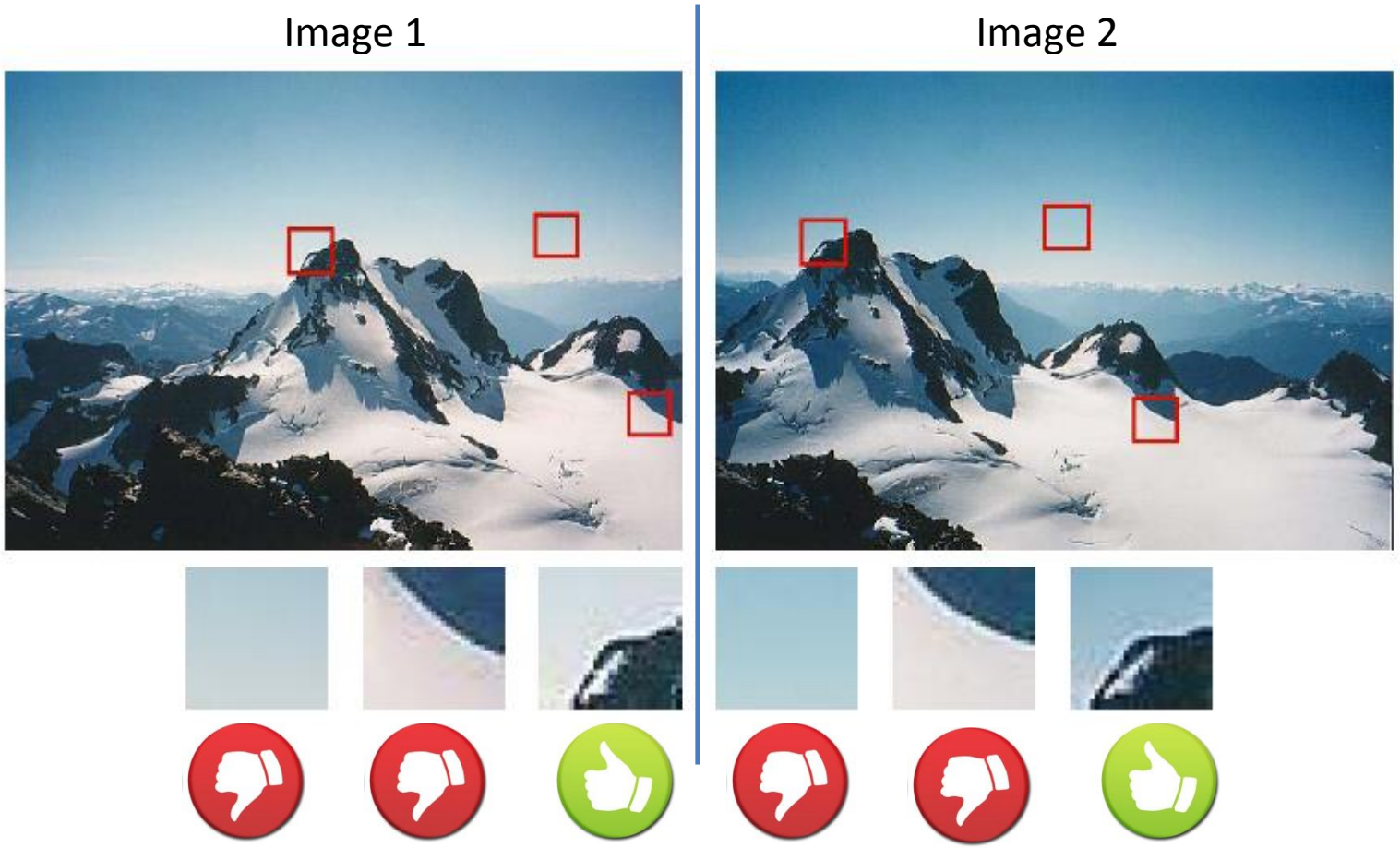


Main questions

- What features are *salient ? (i.e.,* that can be re-*detected* from other views)
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?

What is a distinctive feature?

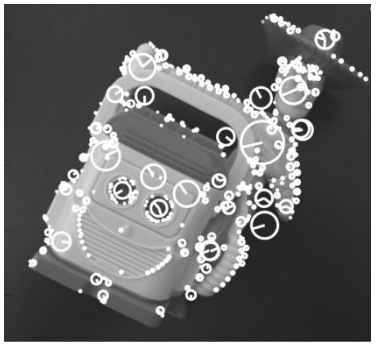
- Consider the image pair below with extracted patches
- Notice how some patches can be localized or matched with higher accuracy than others



Point Features: Corners vs Blobs

- Depending on the type of texture of the image patch, we can have two different types of point features:
  - A **corner** is defined as the intersection of one or more edges
    - A corner has **high localization accuracy**
    - It's **less distinctive than a blob**

  - A **blob** is any other image pattern, **which is not a corner**, that significantly differs from its neighbors in intensity and texture (e.g., a a connected region of pixels with similar color, a circle, etc.
    - **Has less localization accuracy than a corner**
    - It's **more distinctive than a corner**



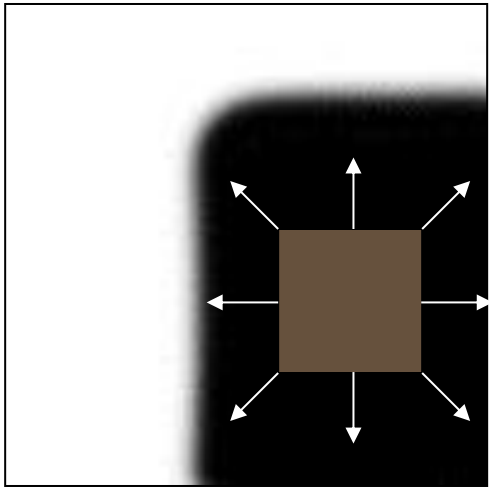
Finding Corners

- Key property: in the region around a corner, image gradient has **two or more** dominant directions
- Corners are **repeatable** and **distinctive**

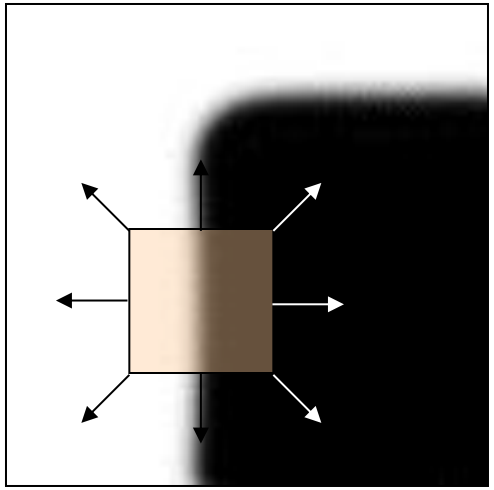
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." , 1988 *Proceedings of the 4th Alvey Vision Conference*: pages 147--151.

Identifying Corners

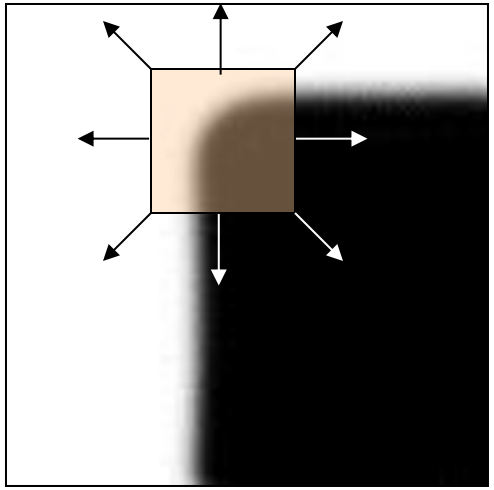
- How do we identify corners?
- We can easily recognize the point by looking through a small window
- Shifting a window in **any direction** should give a **large change** in intensity (e.g., in SSD) in at least 2 directions



"flat" region:
no intensity change
(i.e., $\mathrm{SSD} \approx 0$ in all directions)



"edge":
no change along the edge direction
(i.e., $\mathrm{SSD} \approx 0$ along edge but $\gg 0$ in other directions)



"corner":
significant change in at least 2 directions
(i.e., $\mathrm{SSD} \gg 0$ in all directions)

How do we implement this?

- Consider two image patches of size ***P***. One centered at (x, y) and one centered at $(x+\Delta x, y+\Delta y)$

- The Sum of Squared Differences between them is:

$$SSD(\Delta x, \Delta y) = \sum_{x,y \in P} \left(I(x, y) - I(x+\Delta x, y+\Delta y) \right)^2$$

- Let $I_x = \frac{\partial I(x, y)}{\partial x}$ and $I_y = \frac{\partial I(x, y)}{\partial y}$. Approximating with a 1st order Taylor expansion:

$$I(x+\Delta x, y+\Delta y) \approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y$$

- This produces the approximation

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} \left(I_x(x, y)\Delta x + I_y(x, y)\Delta y \right)^2$$

How do we implement this?

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} \left(I_x(x,y)\Delta x + I_y(x,y)\Delta y \right)^2$$

- This can be written in a matrix form as

$$SSD(\Delta x, \Delta y) \approx \sum \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$\Rightarrow SSD(\Delta x, \Delta y) \approx \sum \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$M = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

How do we implement this?

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} \left(I_x(x,y)\Delta x + I_y(x,y)\Delta y \right)^2$$

- This can be written in a matrix form as

$$SSD(\Delta x, \Delta y) \approx \sum \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

$$\Rightarrow SSD(\Delta x, \Delta y) \approx \sum \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Notice that these are NOT matrix products but pixel-wise products!

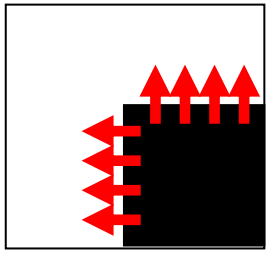
$$M = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

2nd moment matrix Alternative way to write M

What does this matrix reveal?

- First, consider an axis-aligned corner:

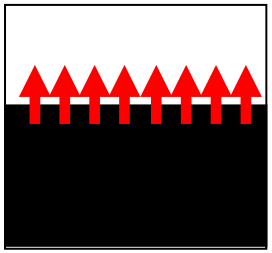
$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



Corner

- This means dominant gradient directions align with x or y axis
- If either λ is close to 0, then this is **not** a corner:

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



Edge

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

Flat region

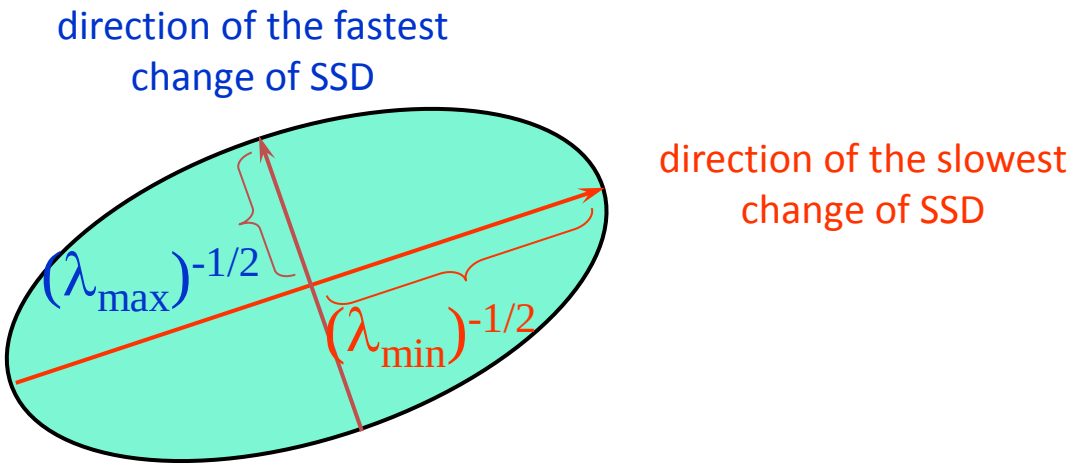
- What if we have a corner that is **not aligned** with the image axes?

General Case

Since M is symmetric, it can always be decomposed into $M = R^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

- We can visualize $[\Delta x \quad \Delta y] M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = const$ as an ellipse with axis lengths determined by the ***eigenvalues*** and the two axes' orientations determined by *R* (i.e., the **eigenvectors** of M)

- The two eigenvectors identify the directions of largest and smallest changes of SSD



How to compute λ_1, λ_2, R from M

Eigenvalue/eigenvector review

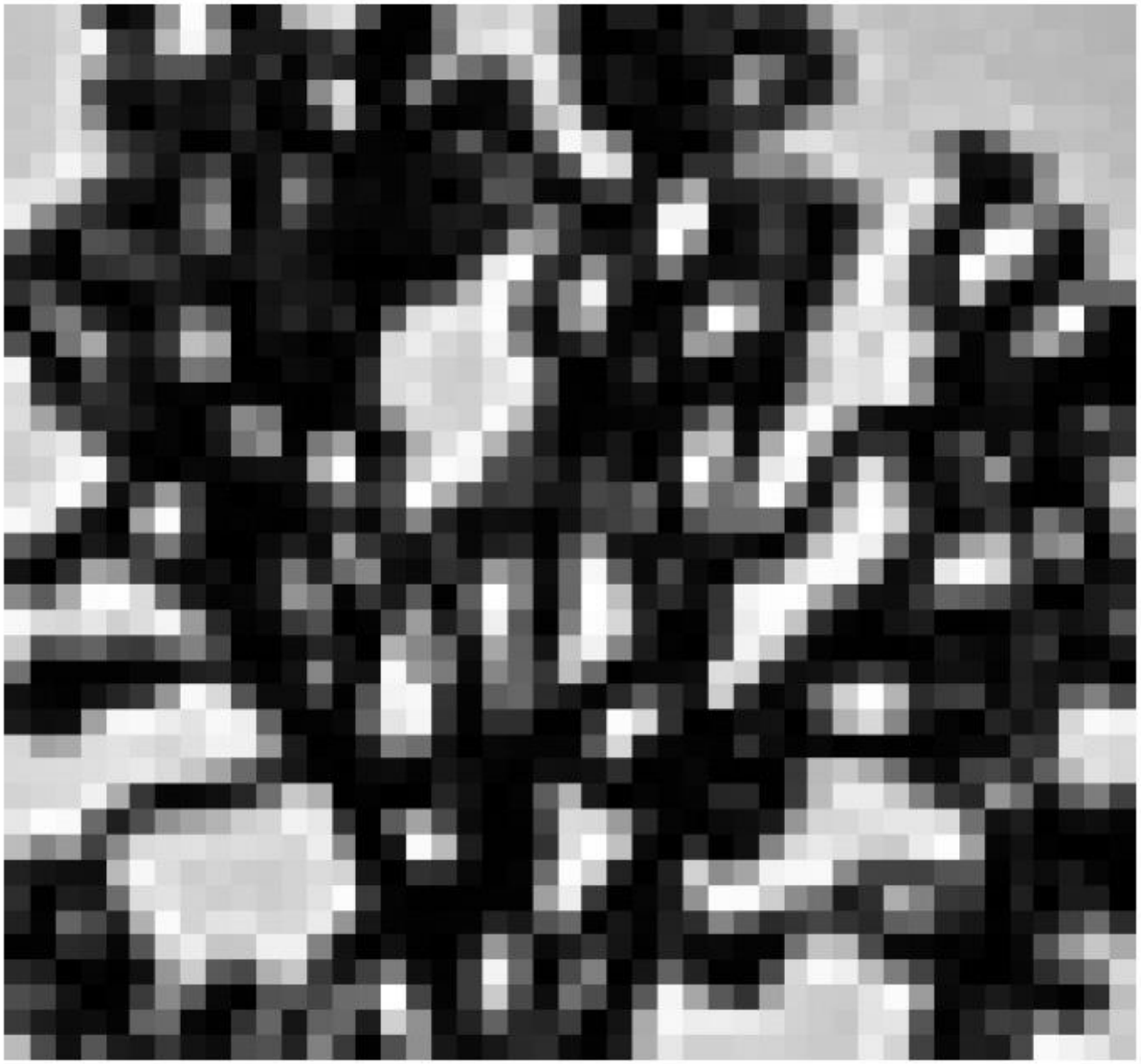
- You can easily proof that λ_1, λ_2 are the **eigenvalues** of M.
- The **eigenvectors** and **eigenvalues** of a matrix **A** are the vectors **x** and scalars λ that satisfy:

$$Ax = \lambda x$$

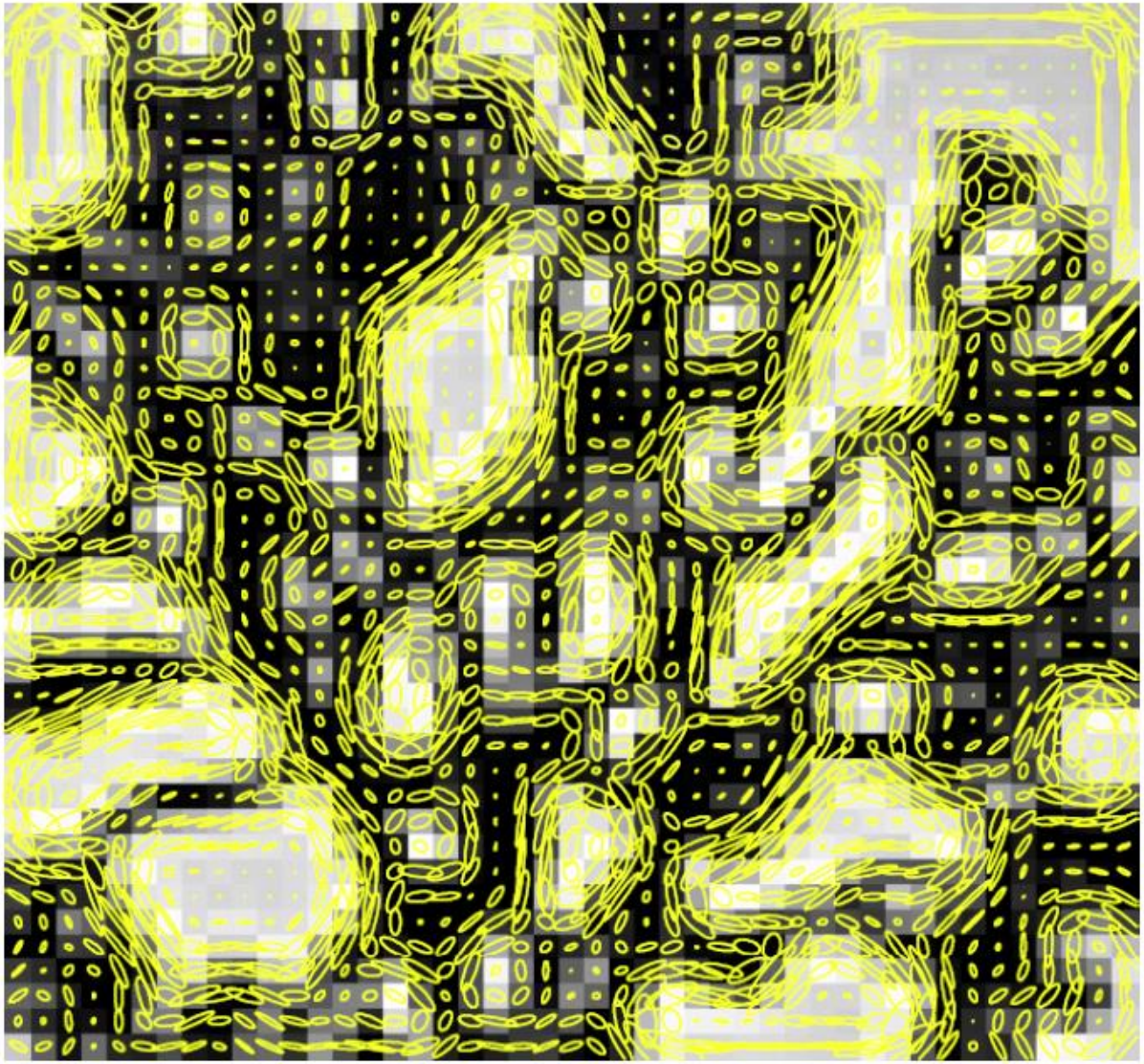
- The scalar λ is the **eigenvalue** corresponding to **x**
  - The eigenvalues are found by solving: $\det(A - \lambda I) = 0$
  - In our case, ***A*** = ***M*** is a 2x2 matrix, so we have $\det\begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix} = 0$
  - The solution is: $\lambda_{1,2} = \frac{1}{2}\left[(m_{11} + m_{22}) \pm \sqrt{4m_{12}m_{21} + (m_{11} - m_{22})^2}\right] = 0$
  - Once you know λ, you find the two eigenvectors **x** (i.e., the two columns of R) by solving:

$$\begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = 0$$

Visualization of 2nd moment matrices



Visualization of 2nd moment matrices



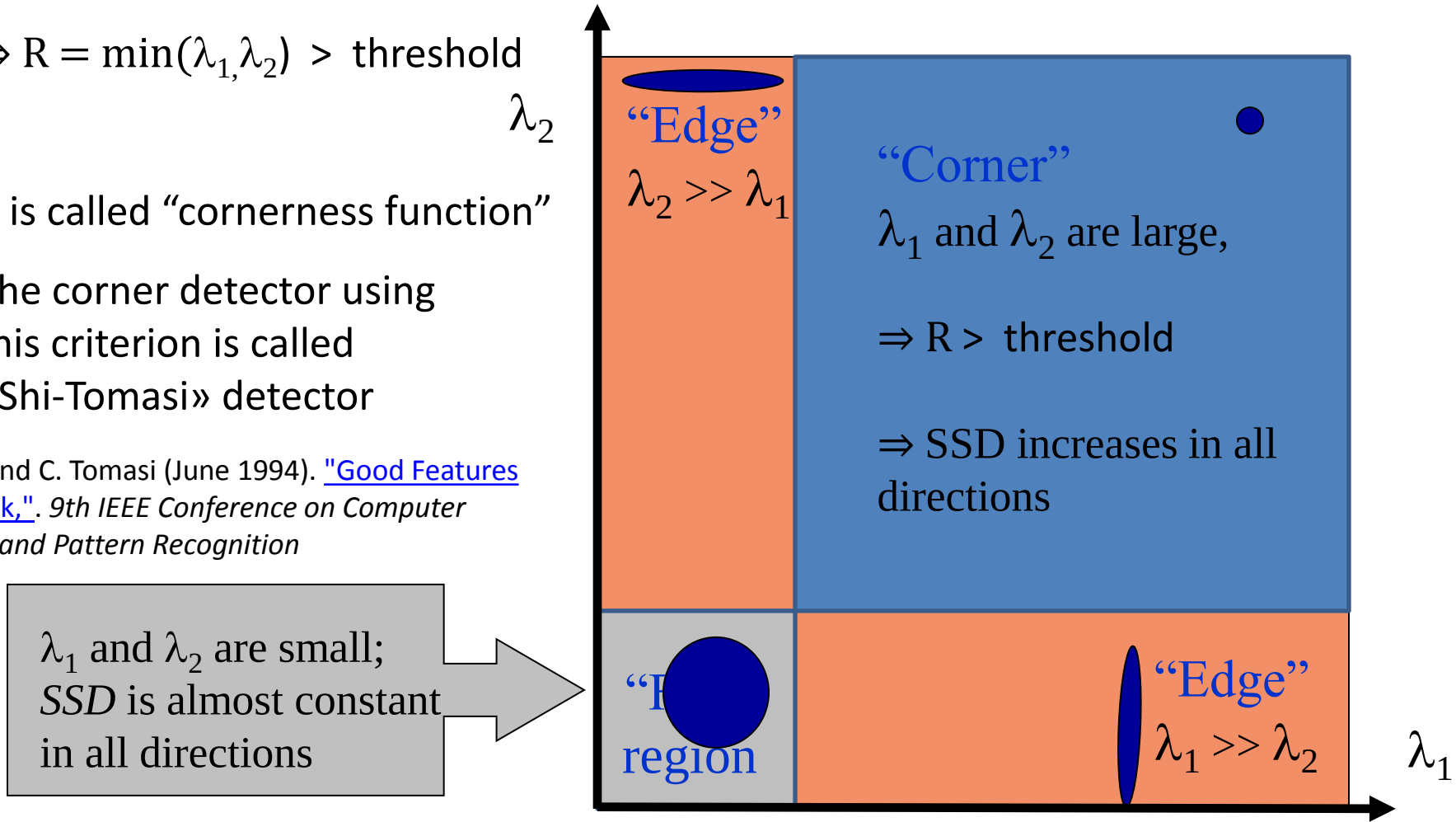
Interpreting the eigenvalues

- Classification of image points using eigenvalues of *M*
- A corner can then be identified by checking whether the minimum of the two eigenvalues of M is larger than a certain user-defined threshold

$\Rightarrow R = \min(\lambda_1, \lambda_2) >$ threshold

- R is called “cornerness function”
- The corner detector using this criterion is called «Shi-Tomasi» detector

J. Shi and C. Tomasi (June 1994). "Good Features to Track,". *9th IEEE Conference on Computer Vision and Pattern Recognition*

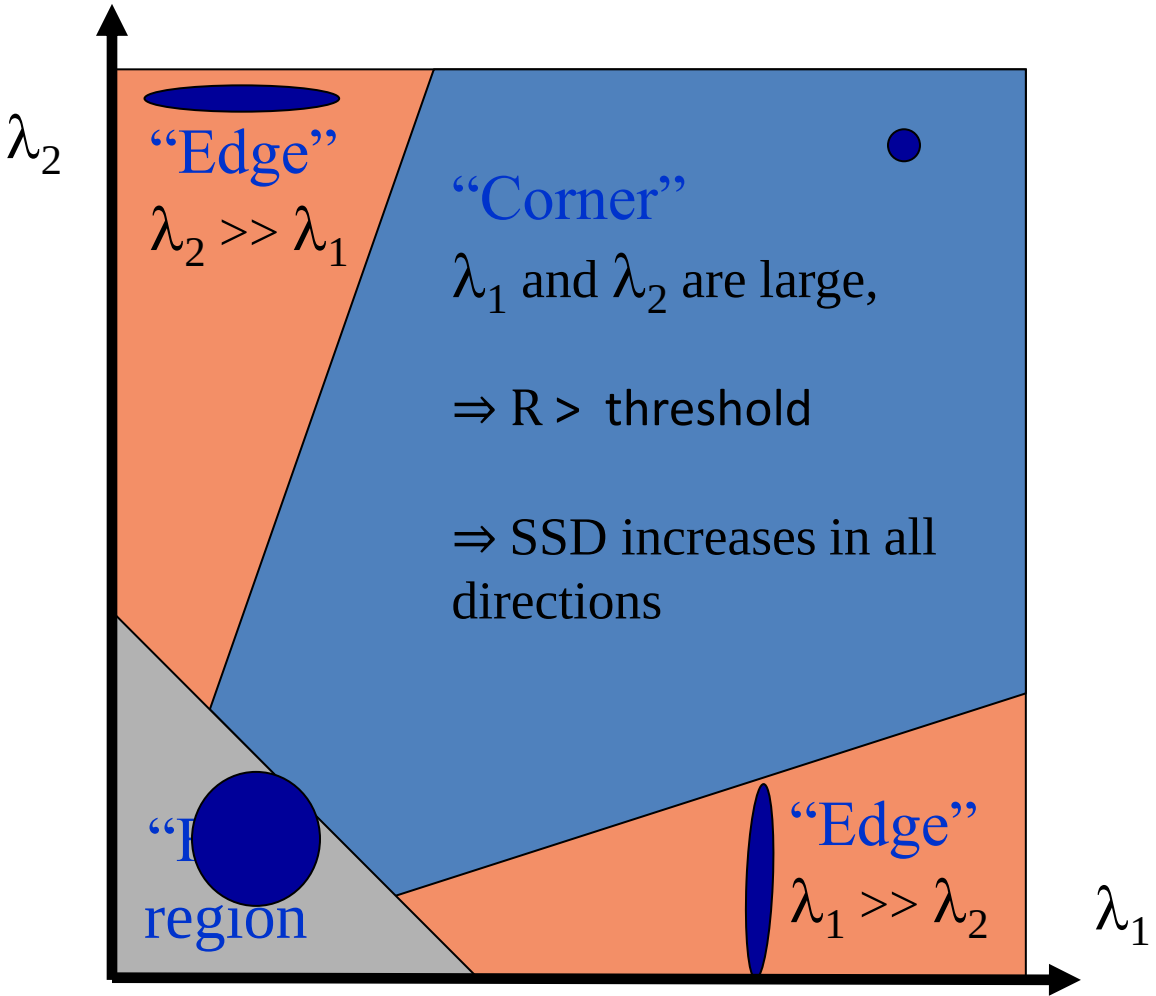


Interpreting the eigenvalues

- Computation of λ_1 and λ_2 is expensive ⇨ Harris & Stephens suggested using a different cornerness function:

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k\,\mathrm{trace}^2(M)$$

- k is a *magic number* in the range (0.04 to 0.15)



Harris Corner Detector

Algorithm:

1. Compute derivatives in x and y directions (I_x, I_y) e.g. with *Sobel filter*
2. Compute ${I_x}^2, {I_y}^2, I_x I_y$
3. Convolve ${I_x}^2, {I_x}^2, I_x I_y$ with a *box filter* to get $\sum {I_x}^2, \sum {I_y}^2, \sum I_x I_y$, which are the entries of the matrix M (optionally use a Gaussian filter instead of a box filter to avoid aliasing and give more "weight" to the central pixels)
4. Compute Harris Corner Measure R (according to Shi-Tomasi or Harris)
5. Find points with large corner response ($R > \text{threshold}$)
6. Take the points of local maxima of *R*

Harris Corner Detector

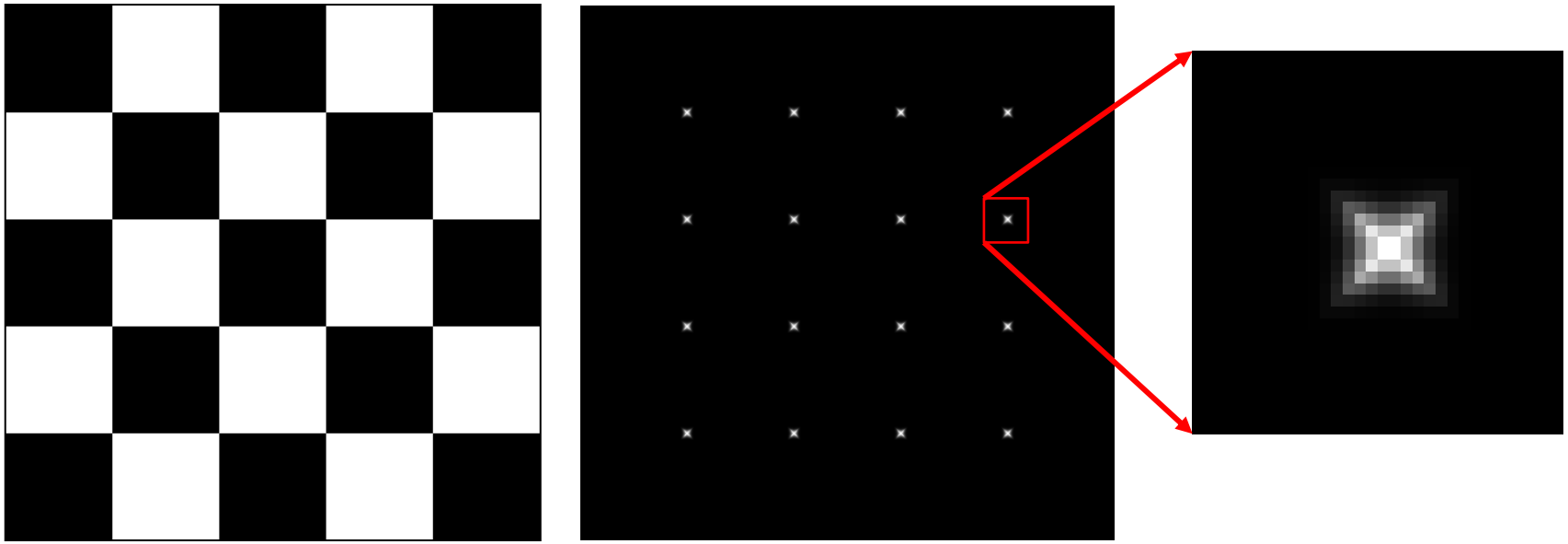
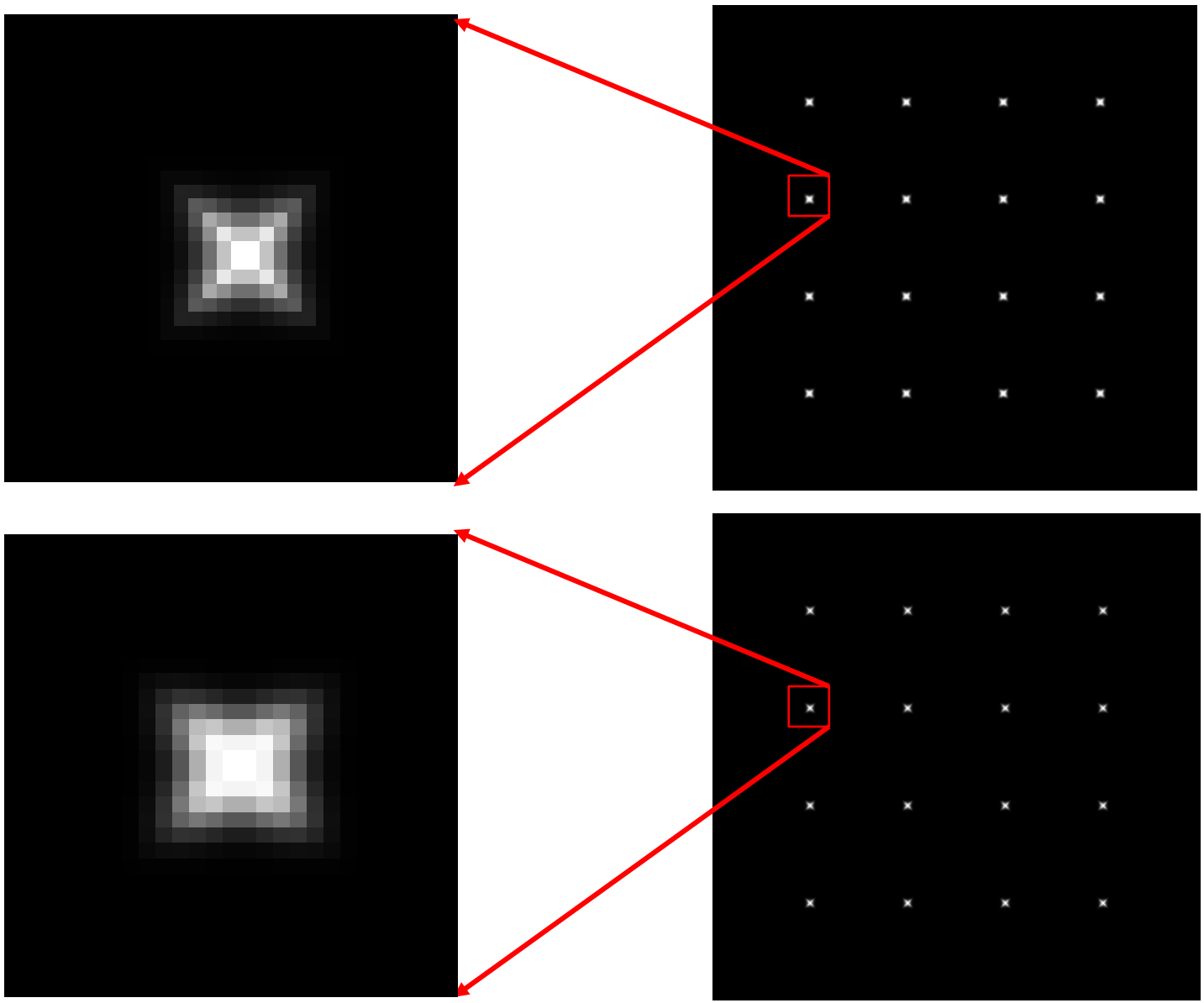


Image I

Cornerness response R

Harris vs. Shi-Tomasi



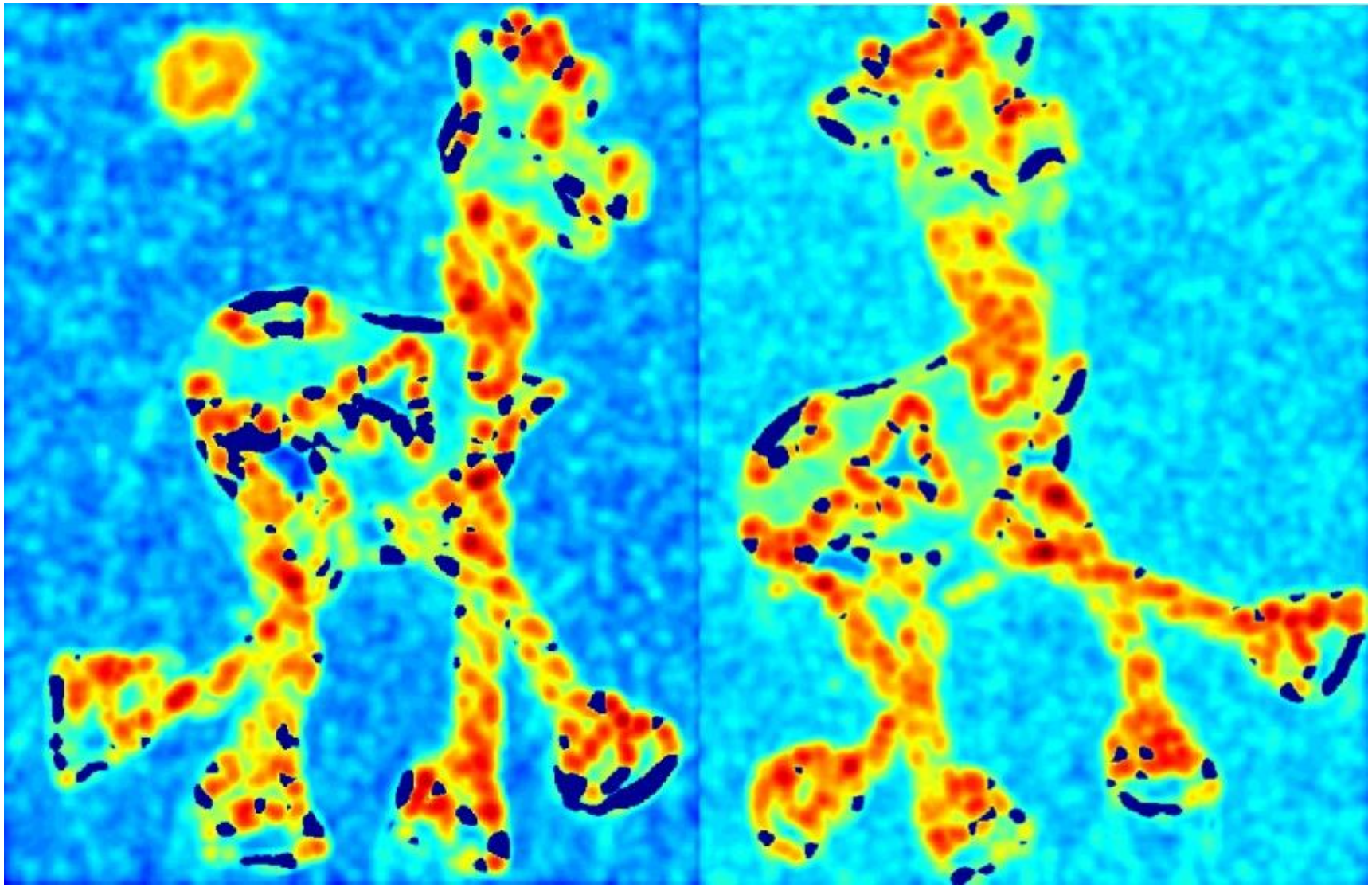
Shi-Tomasi operator

Harris operator

Harris Detector: Workflow

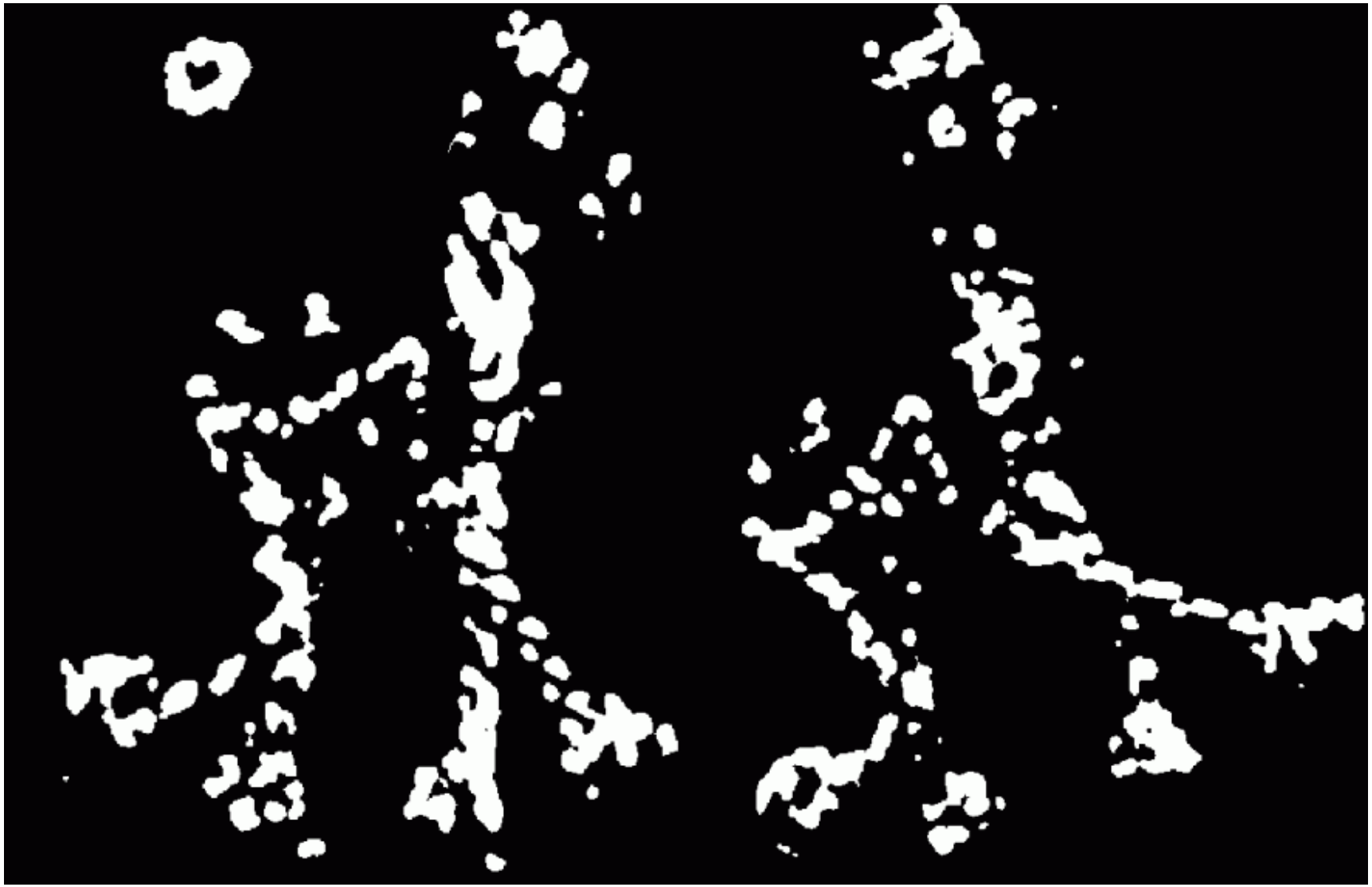
Harris Detector: Workflow

- Compute corner response R



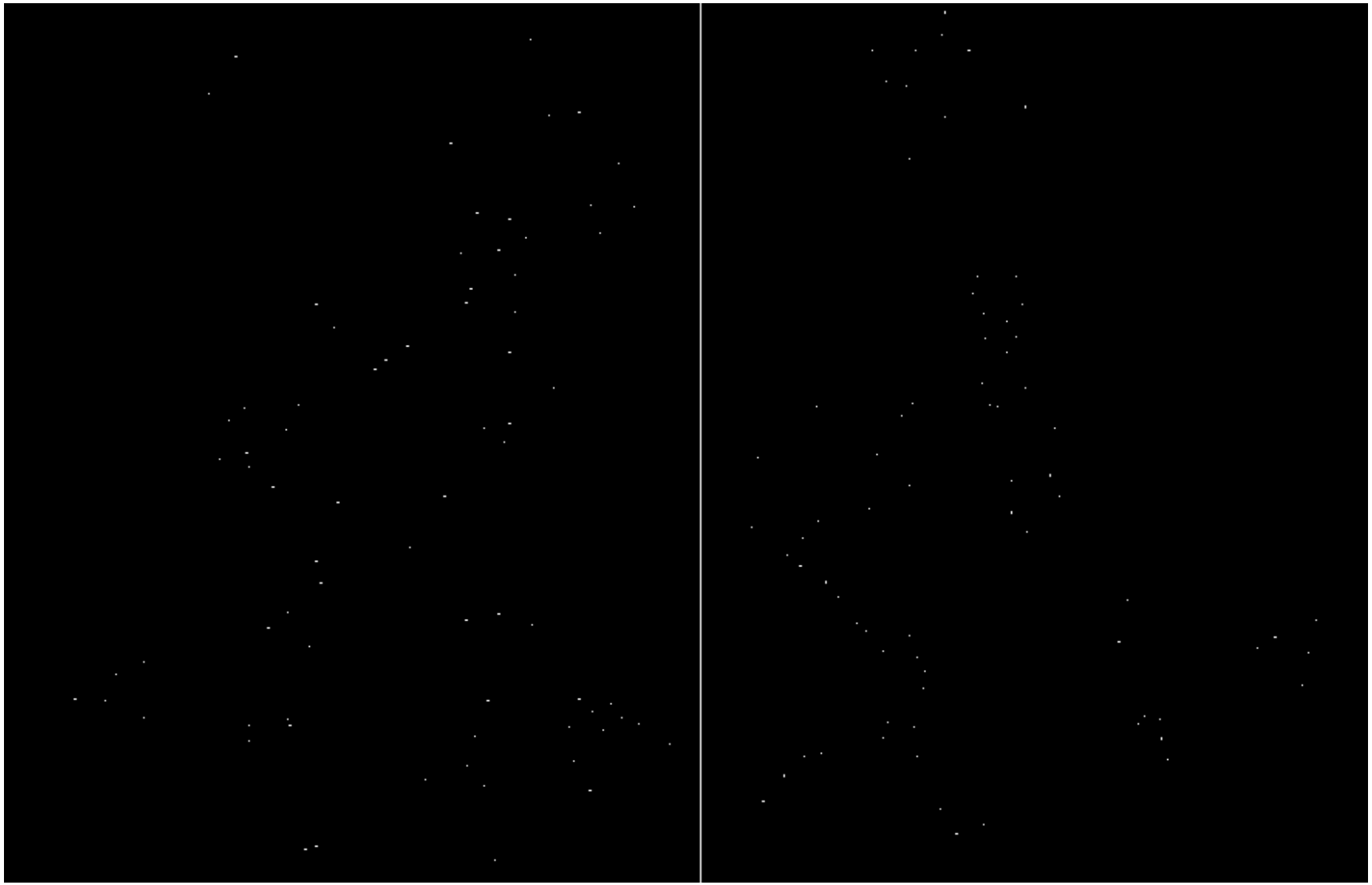
Harris Detector: Workflow

- Find points with large corner response: $R > threshold$



Harris Detector: Workflow

- Take only the points of local maxima of thresholded R



Harris Detector: Workflow

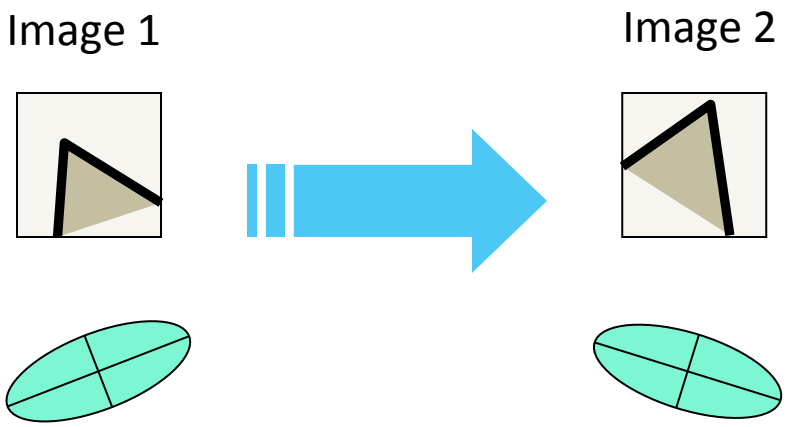
Harris Detector: Some Properties

Repeatability:

- How does the Harris detector behave to common image transformations?
- Can it re-detect the same image patches (Harris corners) when the image exhibits changes in
  - Rotation,
  - View-point,
  - Scale (zoom),
  - Illumination ?
- Solution: Identify properties of detector & adapt accordingly

Harris Detector: Some Properties

- **Rotation invariance**

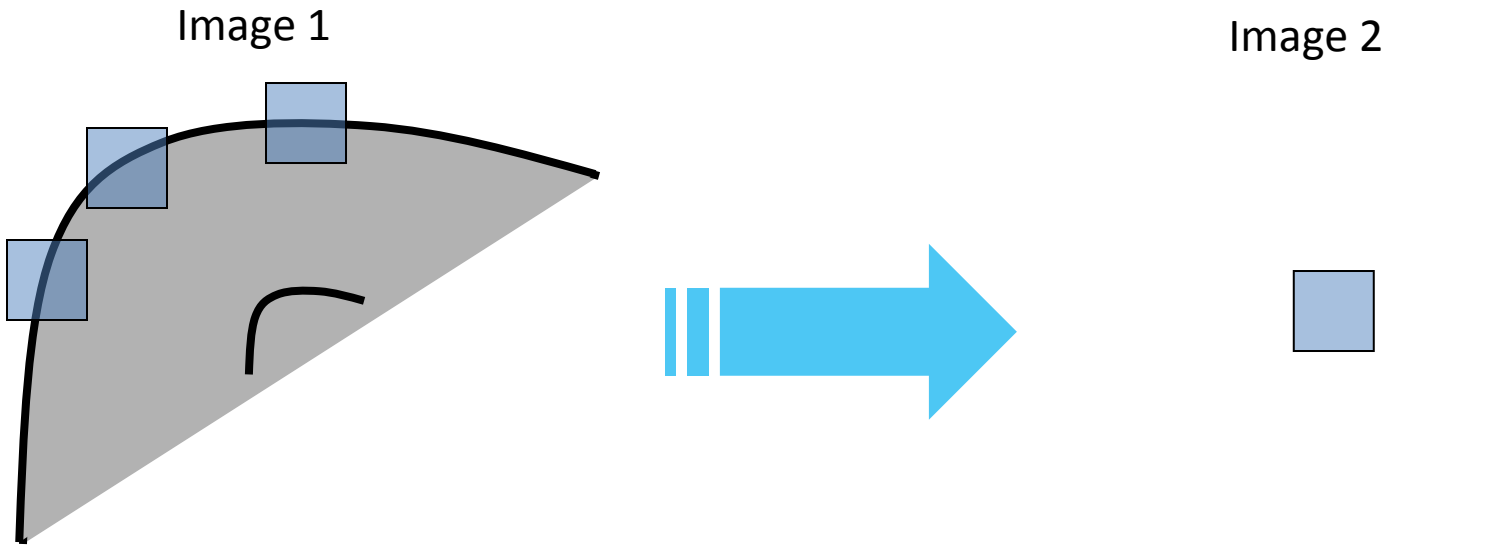


Ellipse rotates but its shape (i.e.`, eigenvalues) remains the same

Corner response R is **invariant to image rotation**

Harris Detector: Some Properties

- But: non-invariant to **image scale**!



All points will be classified as **edges**

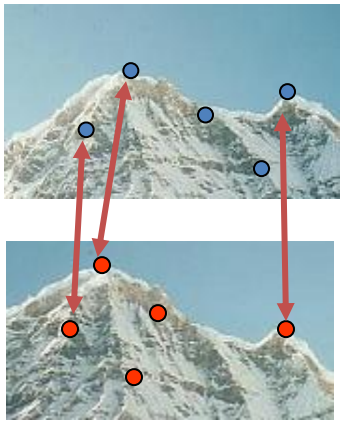
Corner!

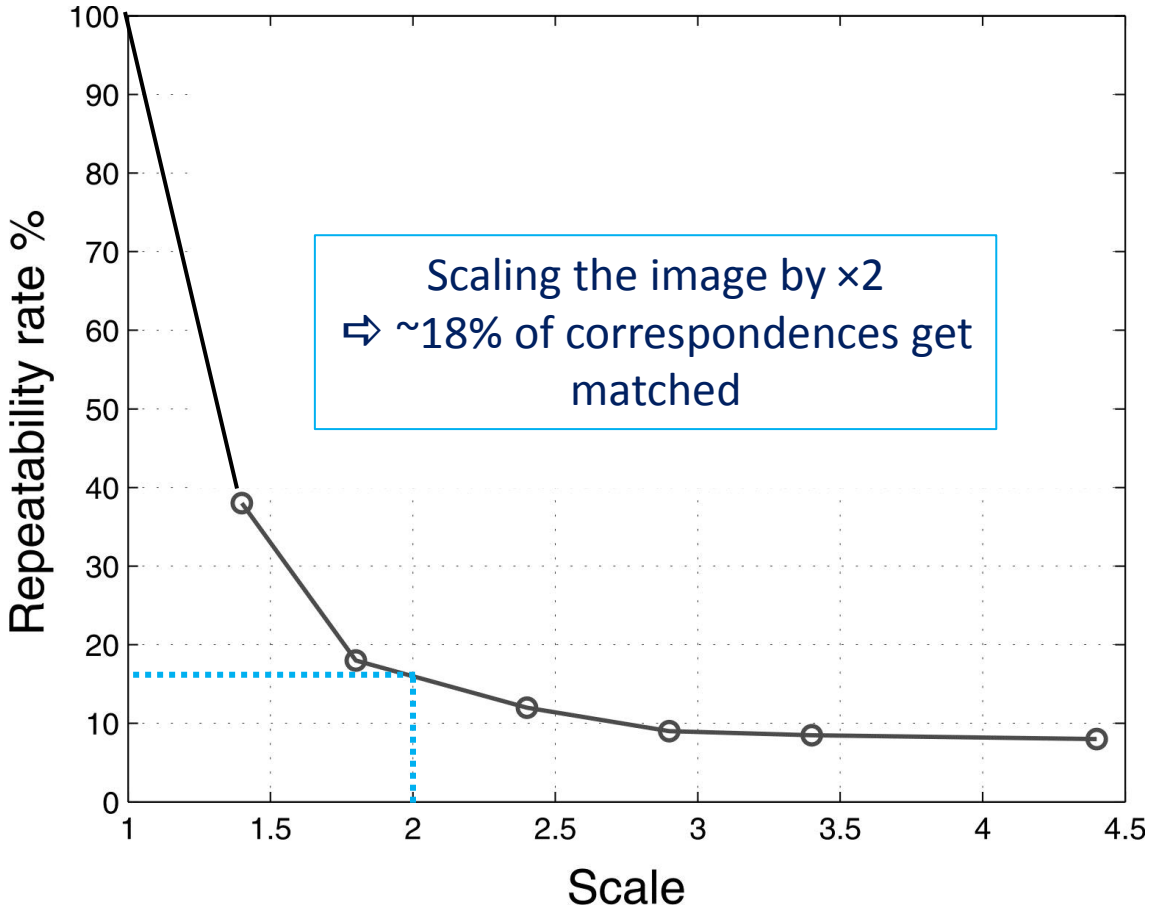
Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability=

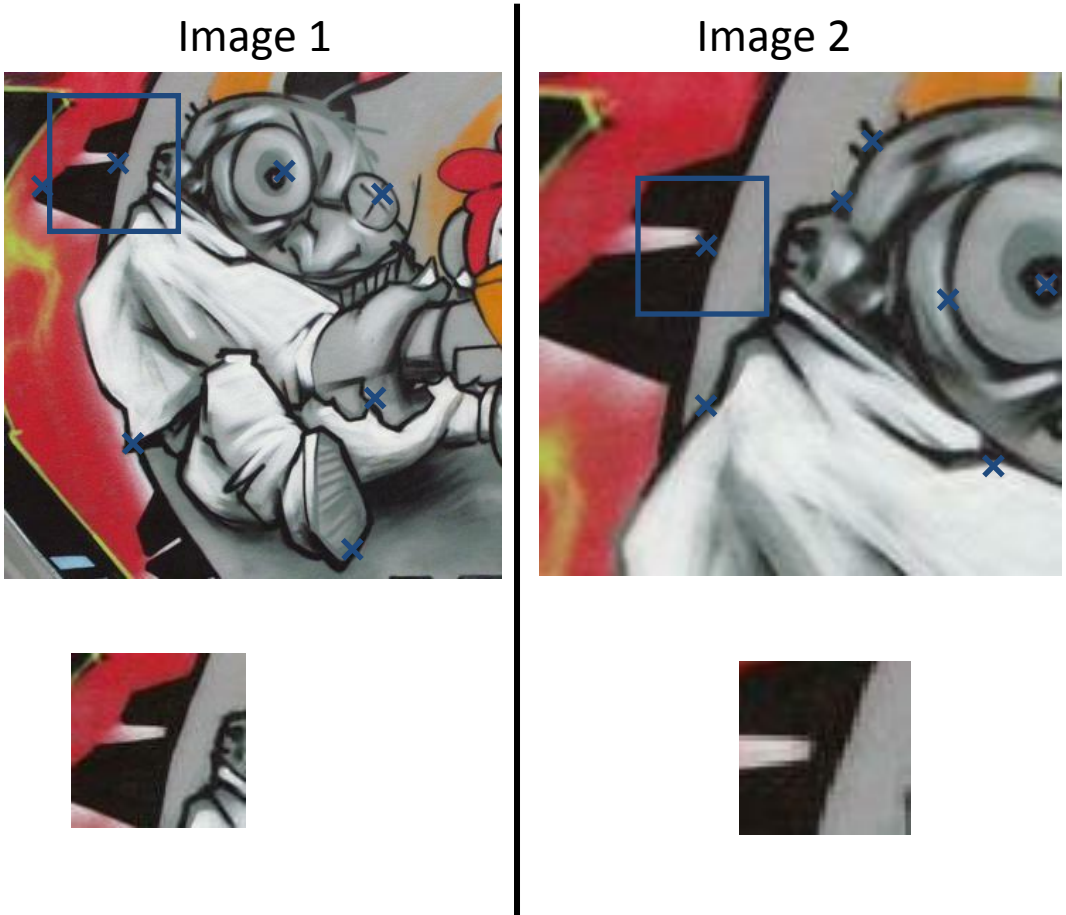
$$\frac{\text{\# correspondences detected}}{\text{\# correspondences present}}$$





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!

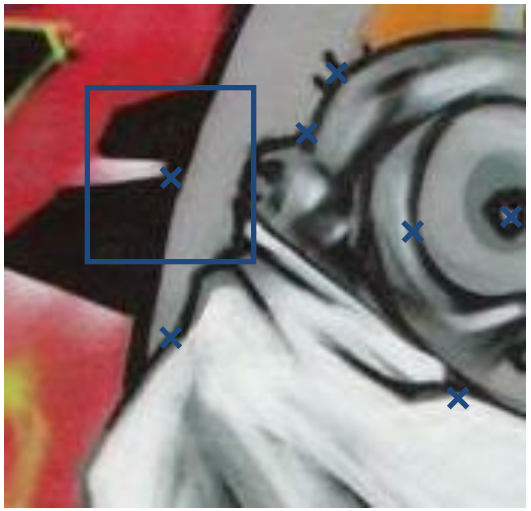


Scale changes

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

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!

Image 1

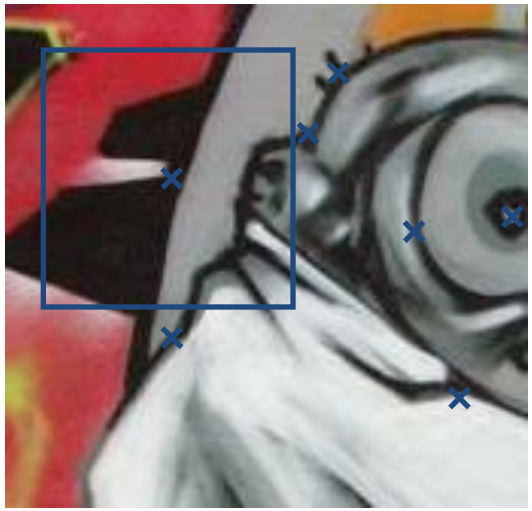
Image 2

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!

Image 1

Image 2



Scale changes

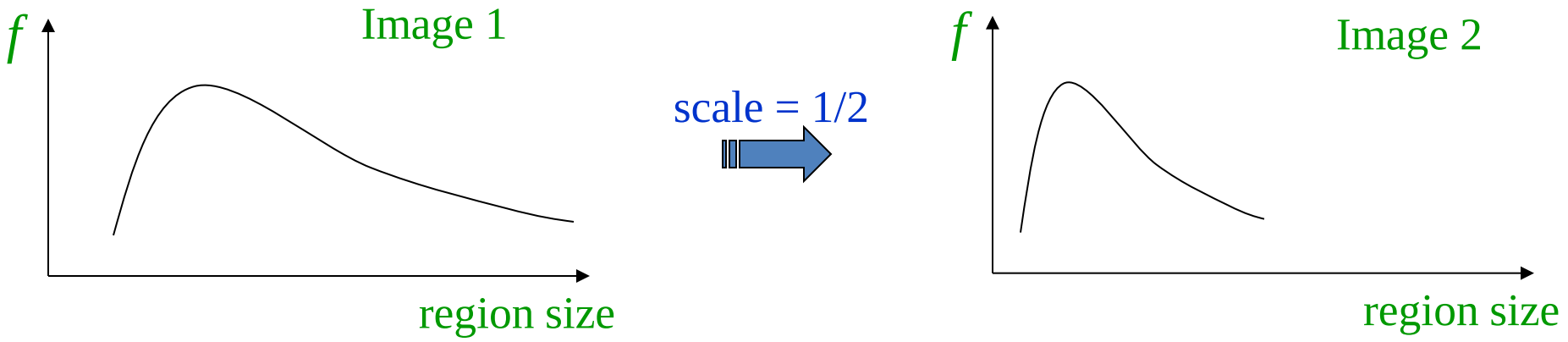
- Scale search is time consuming (needs to be done individually for all patches in one image)
- Possible solution: assign each feature its own “scale” (i.e., size).
  - What’s the optimal scale (i.e., size) of the patch?

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 scales)

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)



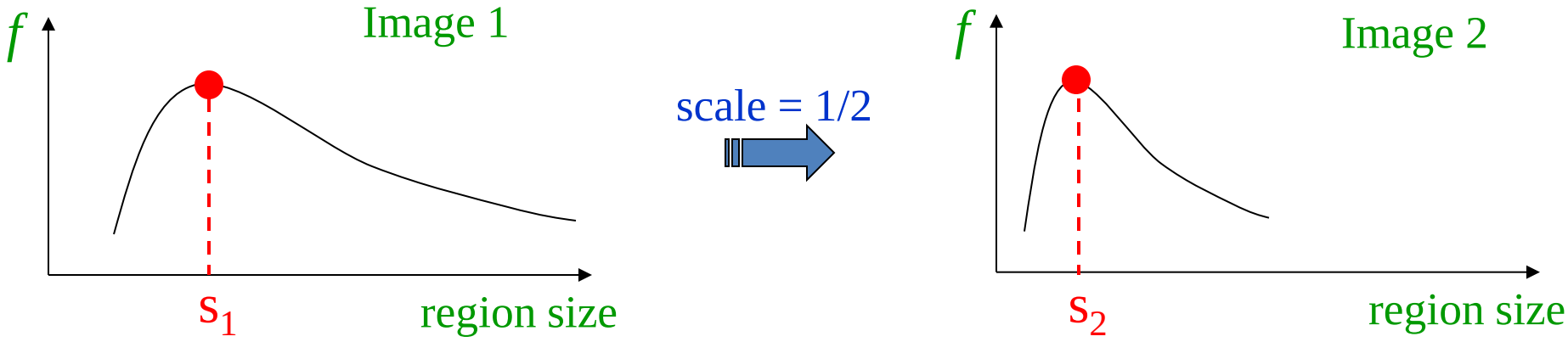
Automatic Scale Selection

- Common approach:

  Take a local maximum of this function

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

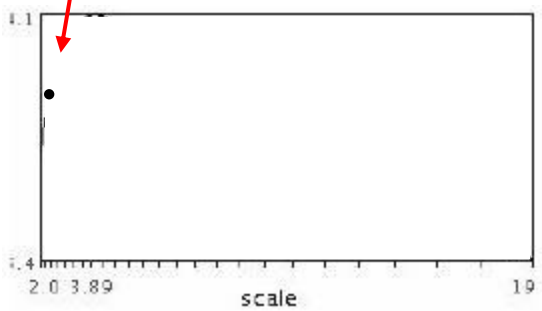
Important: this scale invariant region size is found in each image independently!



Automatic Scale Selection

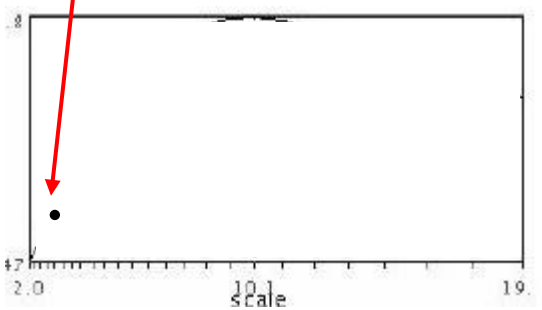
- Function responses for increasing scale (scale signature)

Image 1



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

Image 2

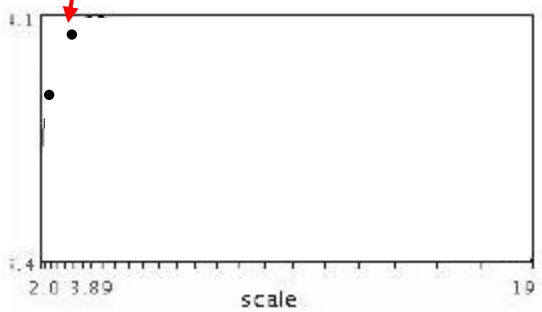


$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

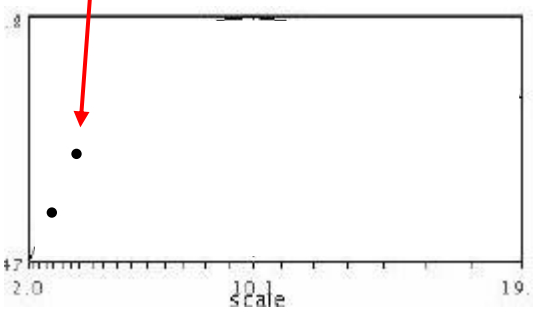
- Function responses for increasing scale (scale signature)

Image 1



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

Image 2

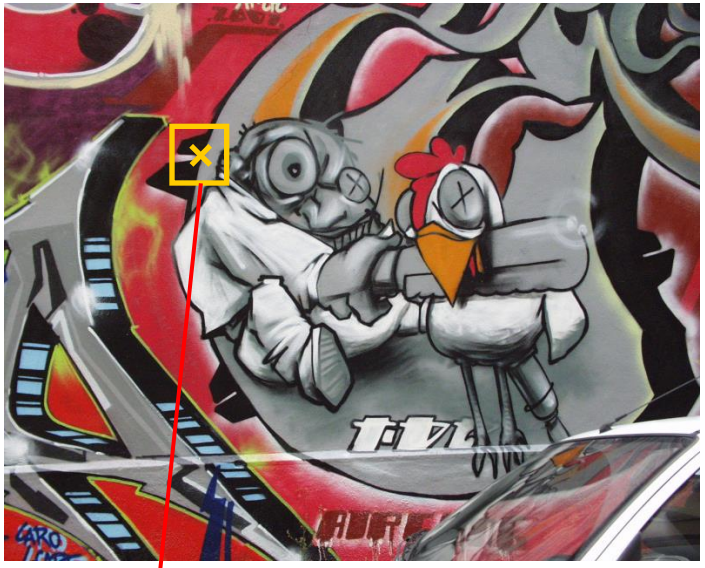


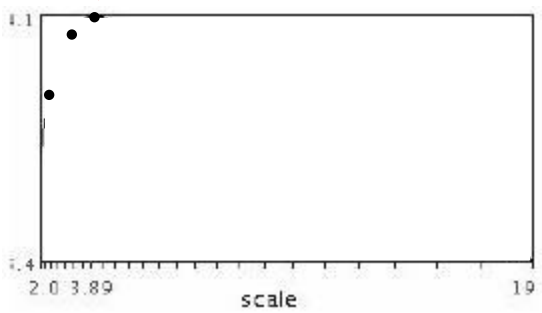
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

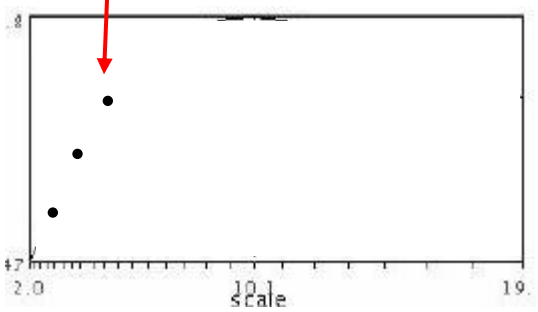
Image 1





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

Image 2

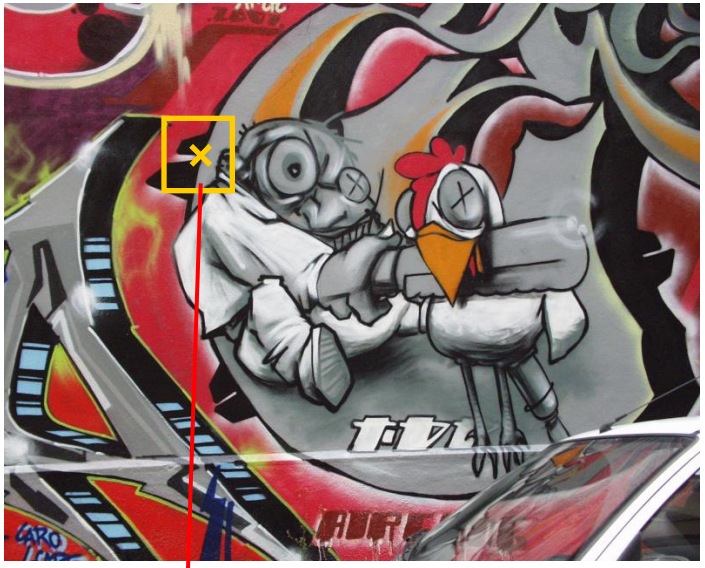


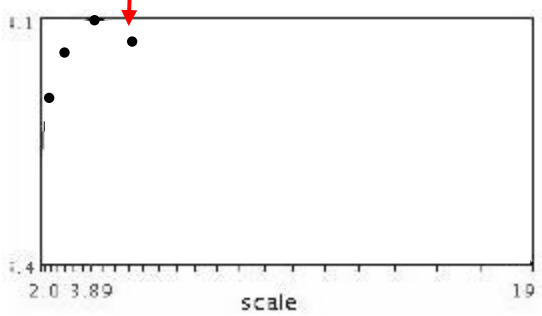
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

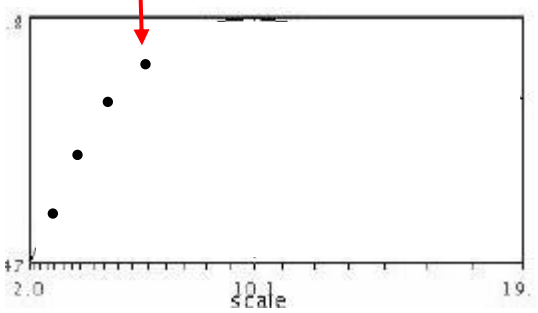
Image 1





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

Image 2

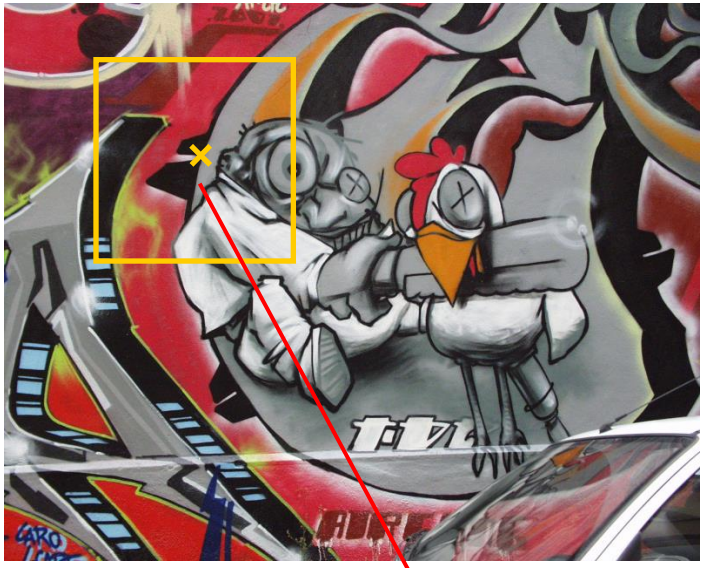


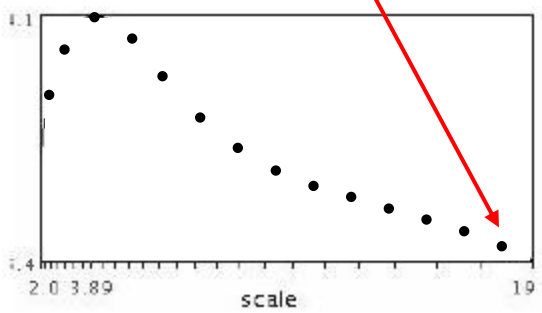
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

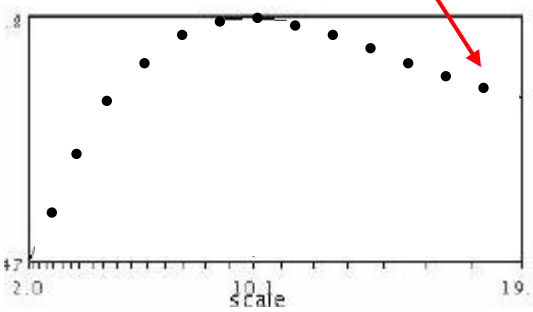
Image 1





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

Image 2

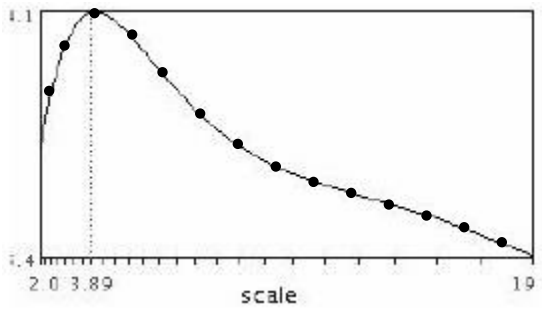


$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

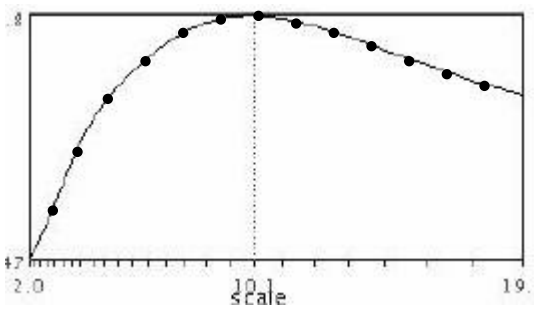
- Function responses for increasing scale (scale signature)

Image 1



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

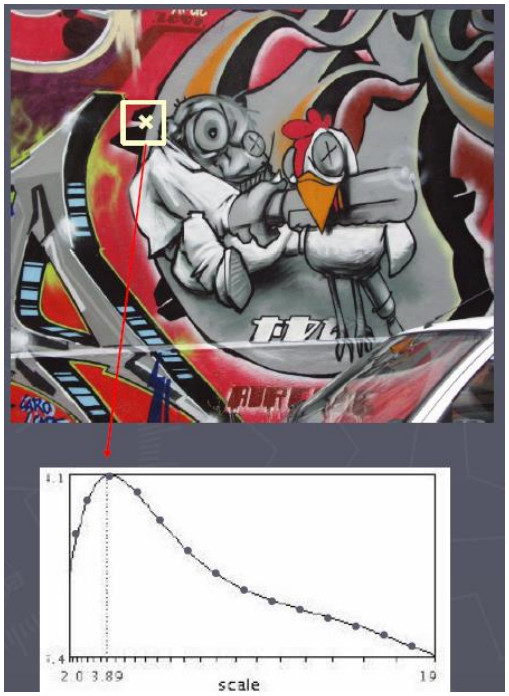
Image 2

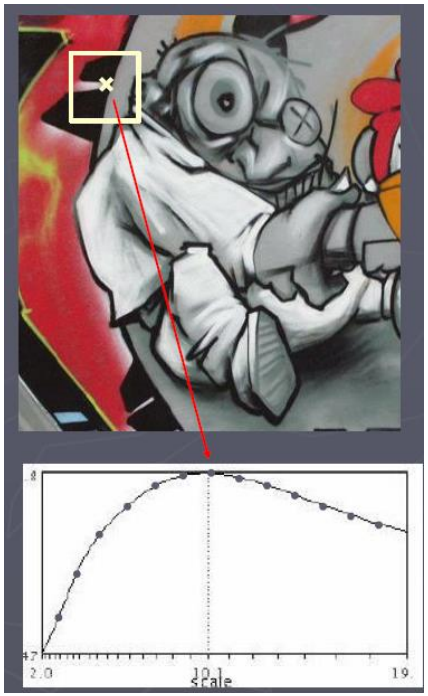


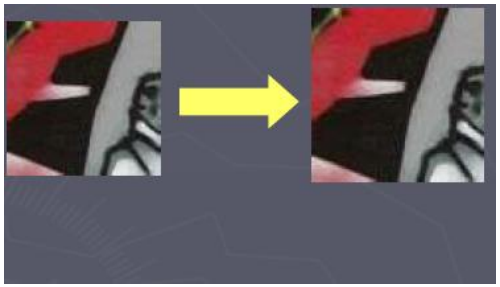
$f(I_{i_1 \ldots i_m}(x',\sigma'))$

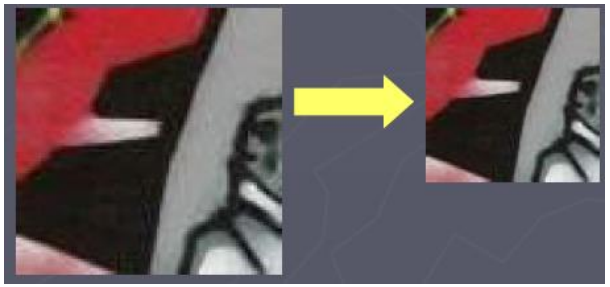
Automatic Scale Selection

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



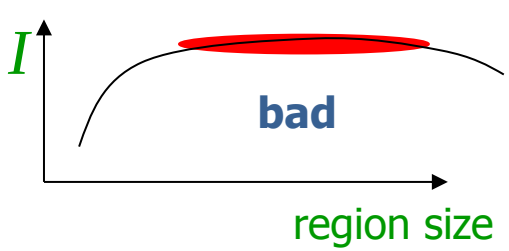




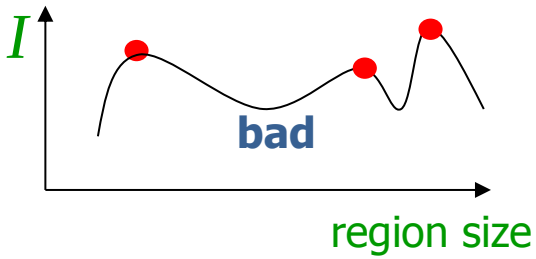


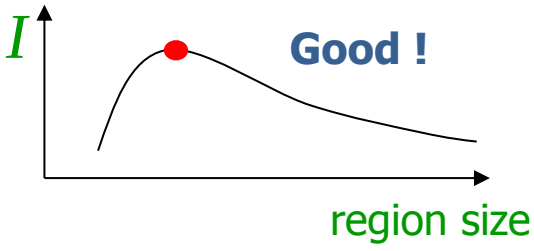
Scale Invariant Detection: Robustness

- 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

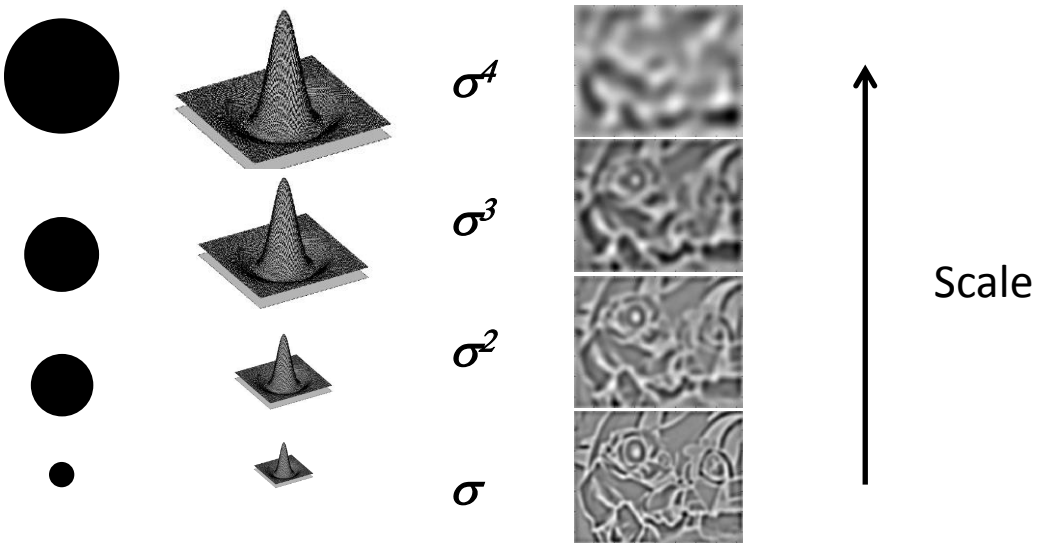
- Functions for determining scale: convolve image with kernel to identify sharp intensity discontinuities

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

- 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

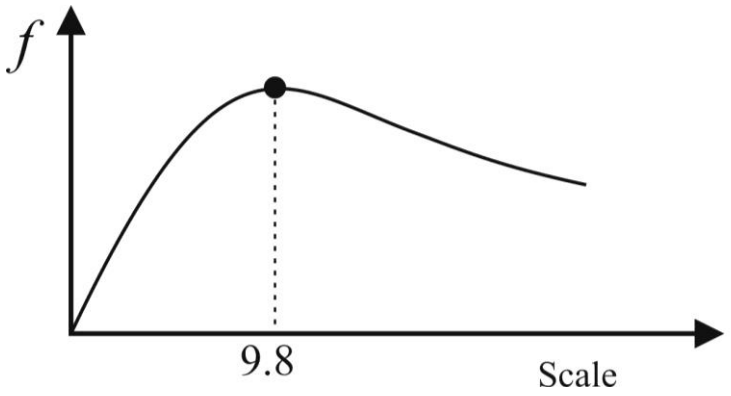
- Functions for determining scale: convolve image with kernel to identify sharp intensity discontinuities

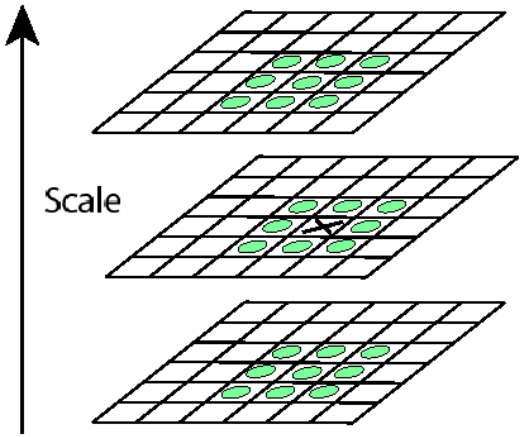
$$f = \text{Kernel} * \text{Image}$$

- 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



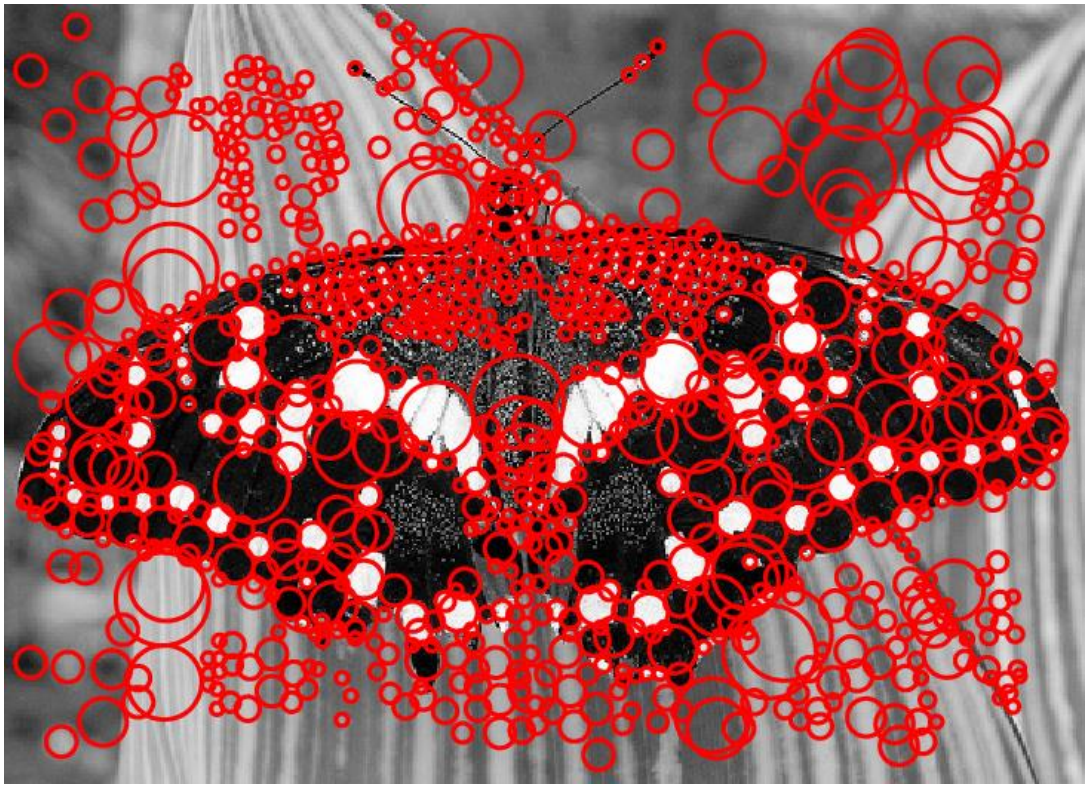


Scale-space detection: Example

Scale-space detection: Example

sigma = 11.9912

Scale-space detection: Example



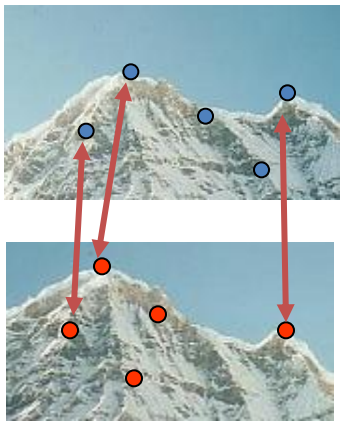
Scale Invariant Detectors

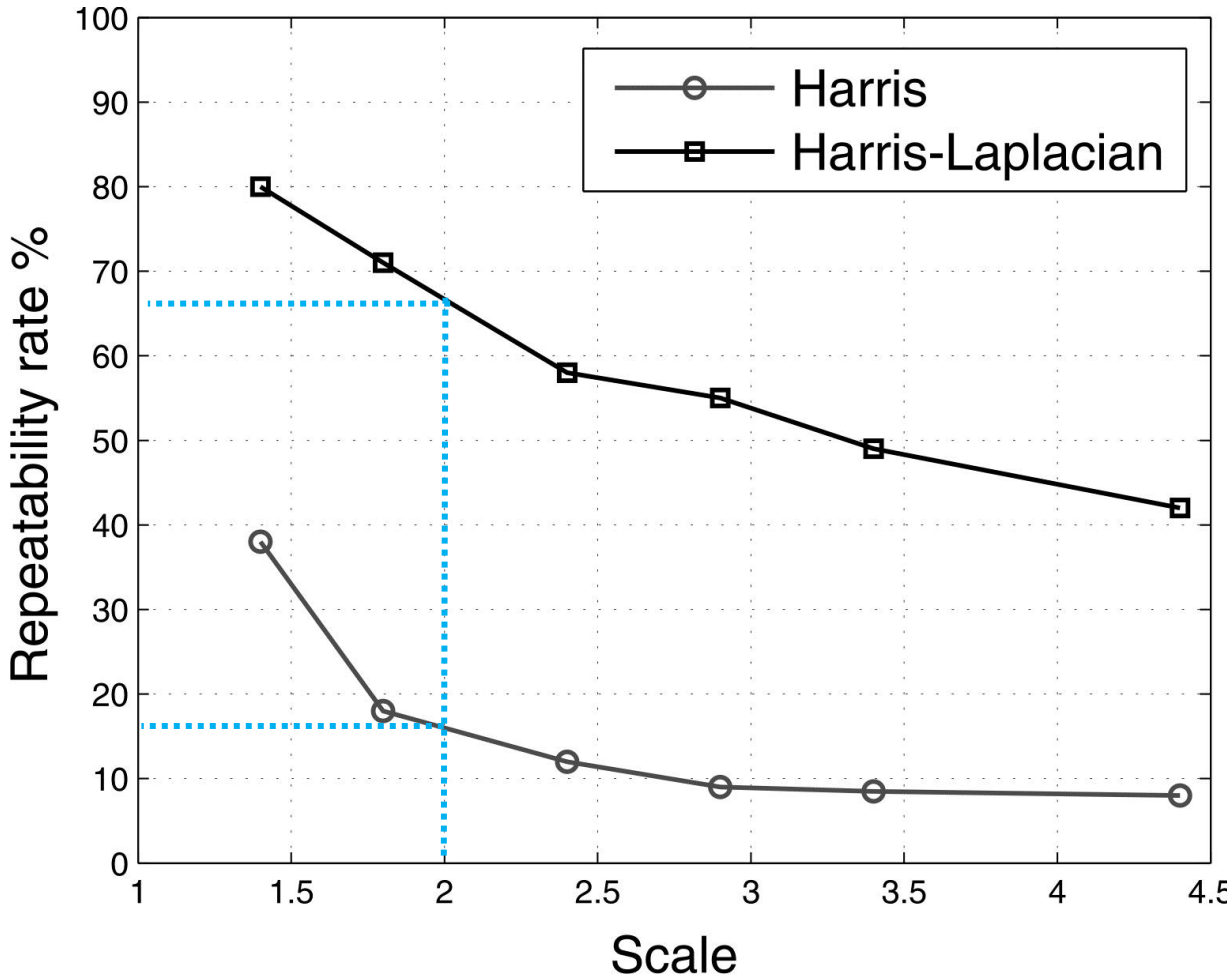
- Experimental evaluation of detectors w.r.t. scale change

Repeatability=

correspondences detected

correspondences present





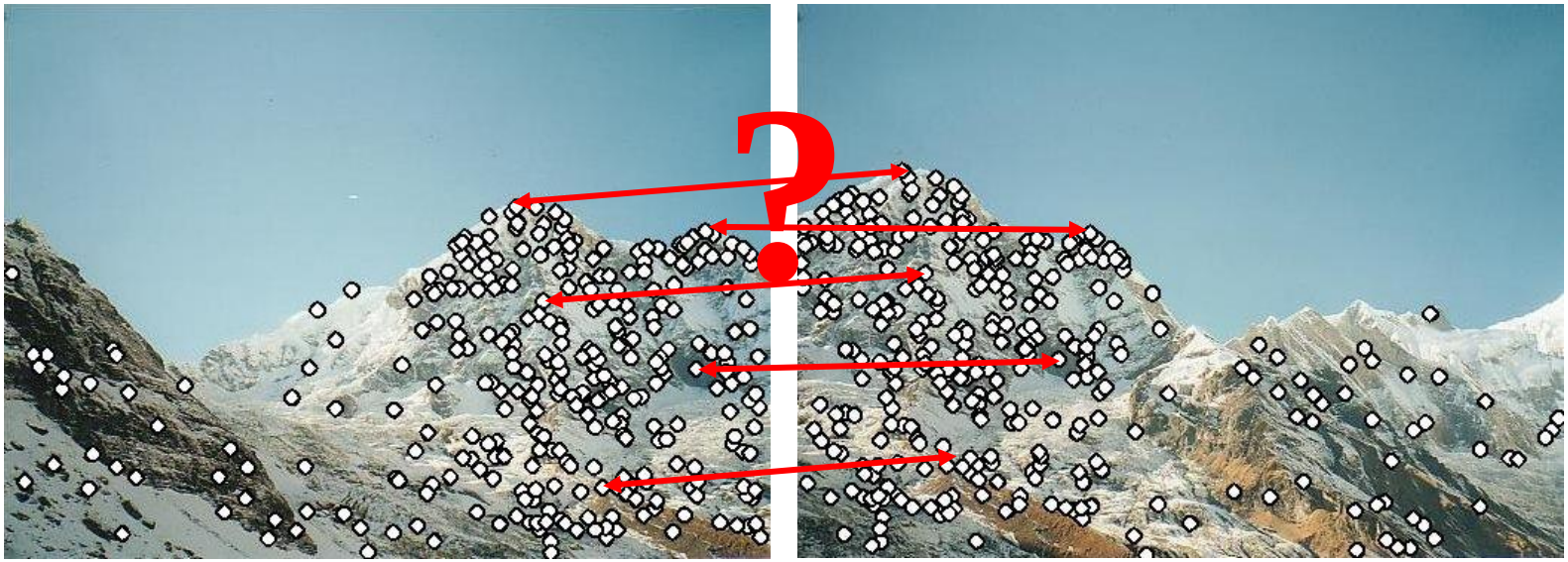
Main questions

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

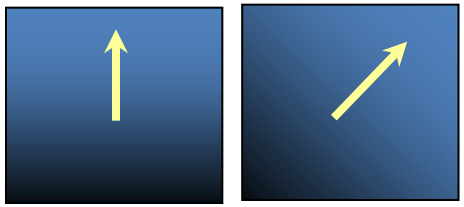
Feature descriptors

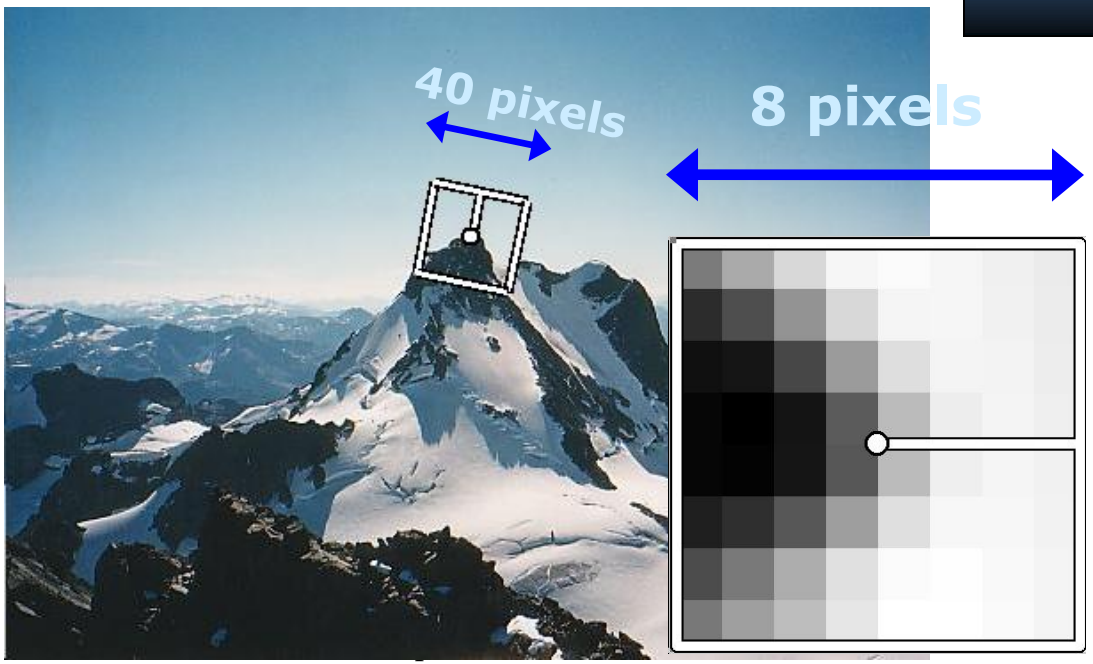
- We'd like to find the same features regardless of the **transformation** (*rotation, scale, view point,* and *illumination)*
  - **Most feature methods** are designed to be invariant to
    - translation,
    - 2D rotation,
    - scale
  - Some of them can also handle
    - **Small view-point invariance** (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

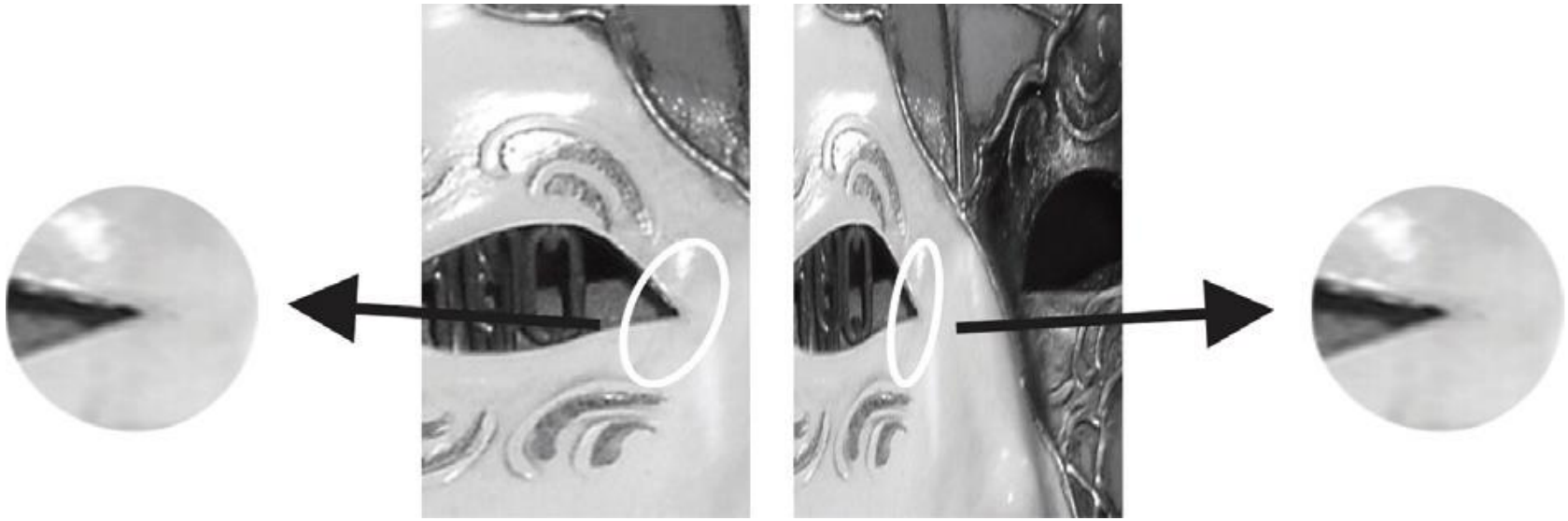




How to achieve invariance

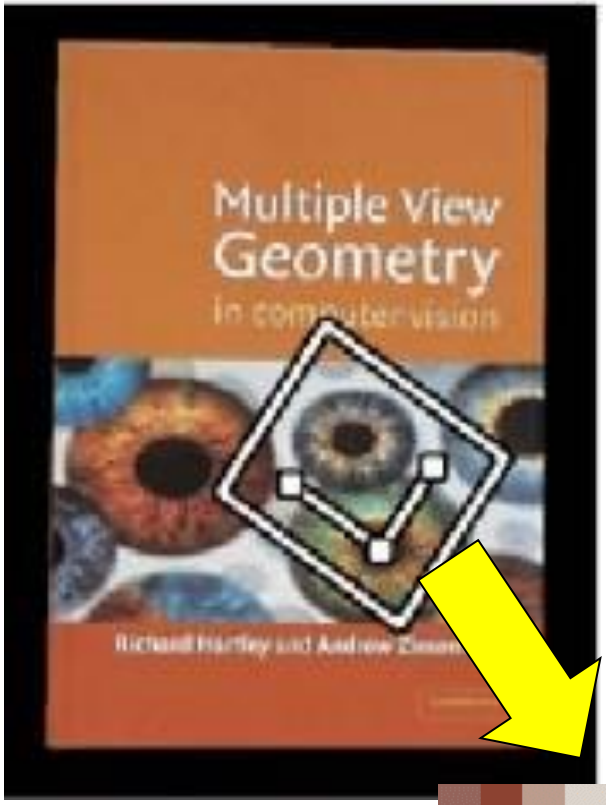
Step 2: Affine Un-warping (to achieve slight view-point invariance)

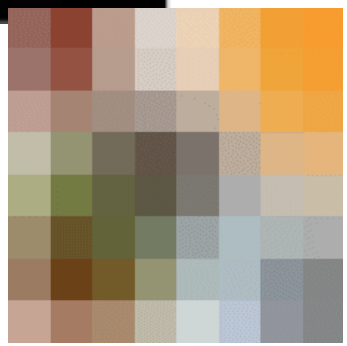
- The second moment matrix *M* can be used to identify the two directions of fastest and slowest change of intensity around the feature.
- Out of these two directions, an elliptic patch is extracted at the scale computed by with the LoG operator.
- The region inside the ellipse is normalized to a circular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

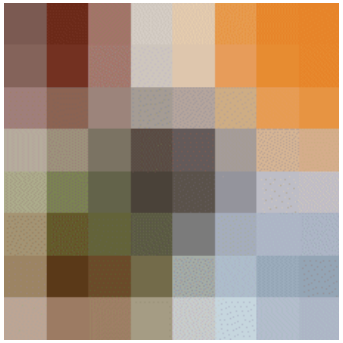


How to achieve invariance

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







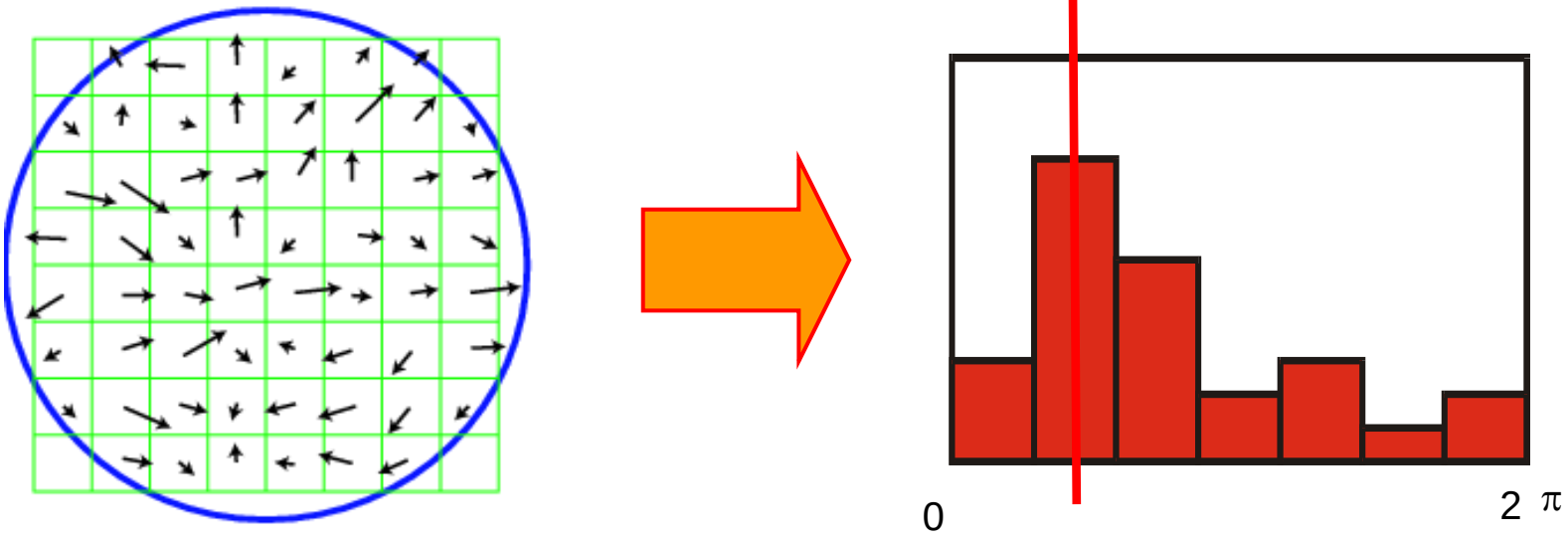
Feature descriptors

- 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)

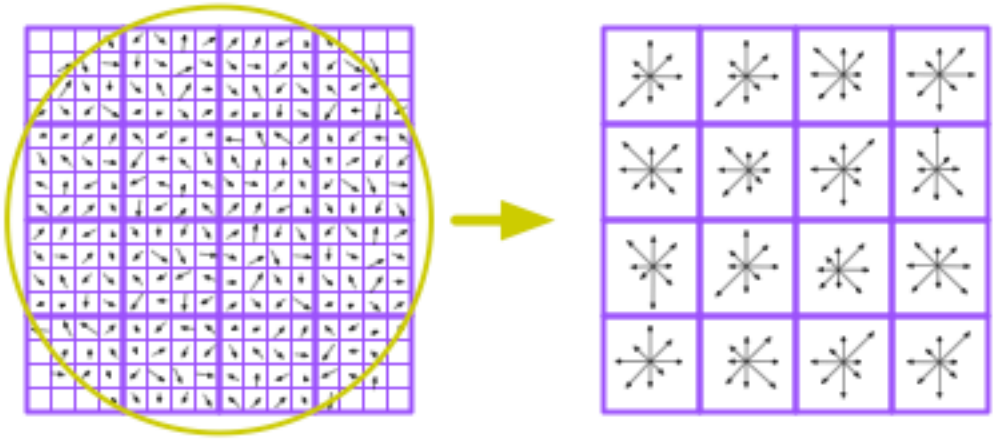
HOG descriptor (Histogram of Oriented Gradients)

- 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



SIFT descriptor

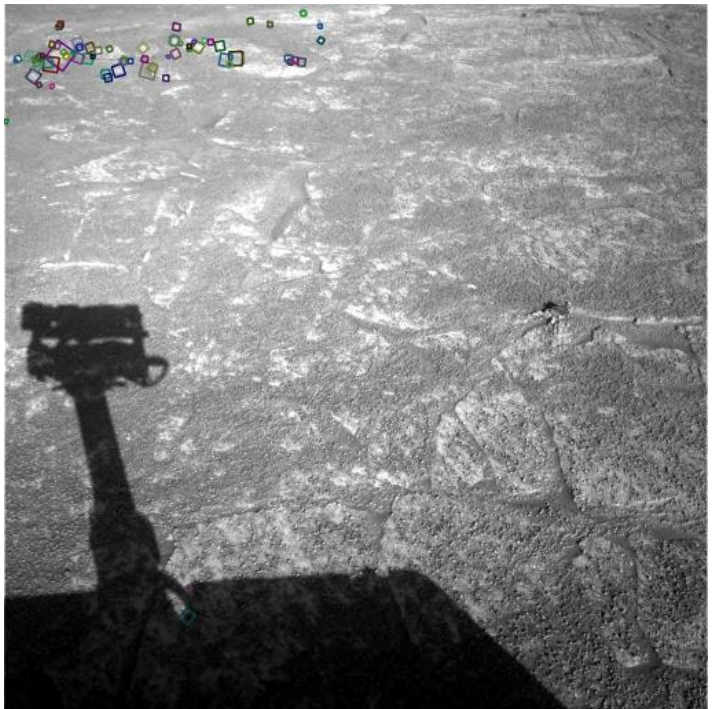
- **S**cale **I**nvariant **F**eature **T**ransform
- Invented by David Lowe [IJCV, 2004]
- Descriptor computation:
  - Divide patch into 4x4 sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
  - Resulting SIFT descriptor: 4x4x8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

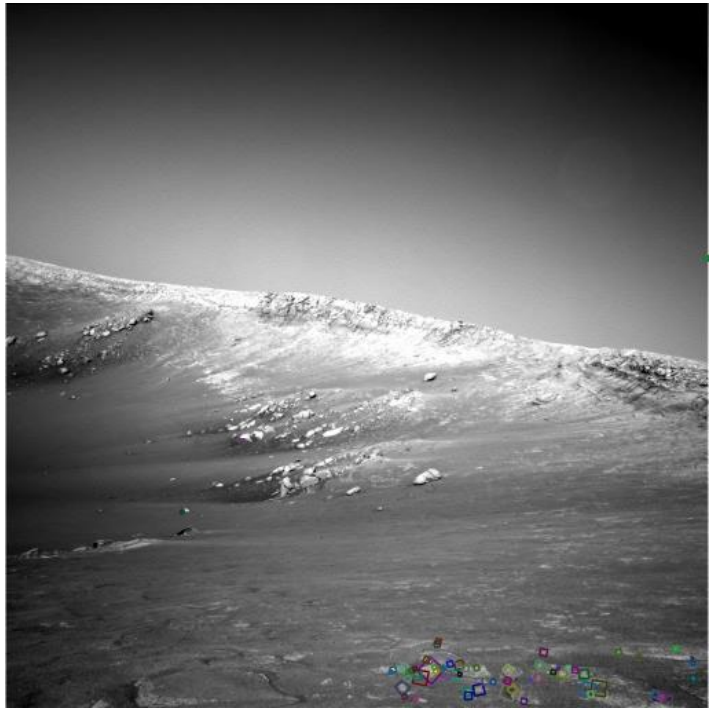


David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* , 2004.

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

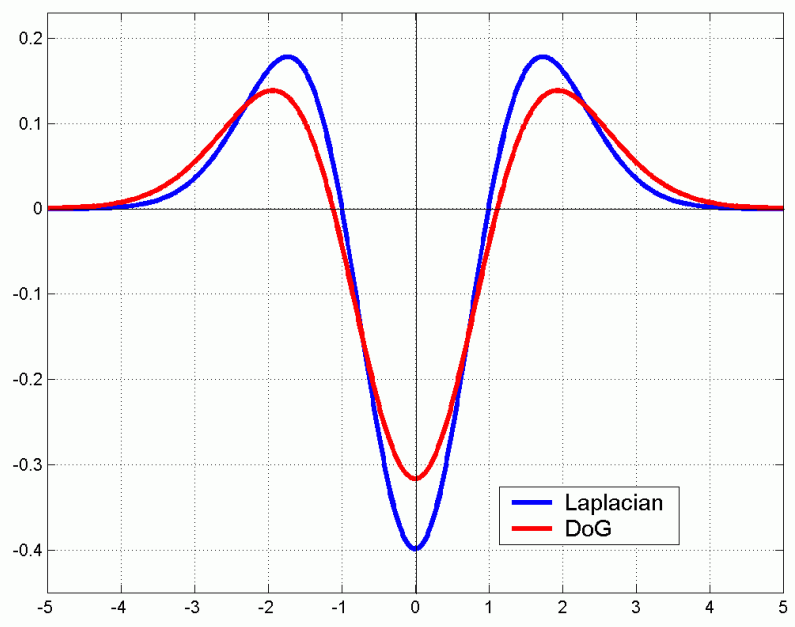




Scale Invariant Detection

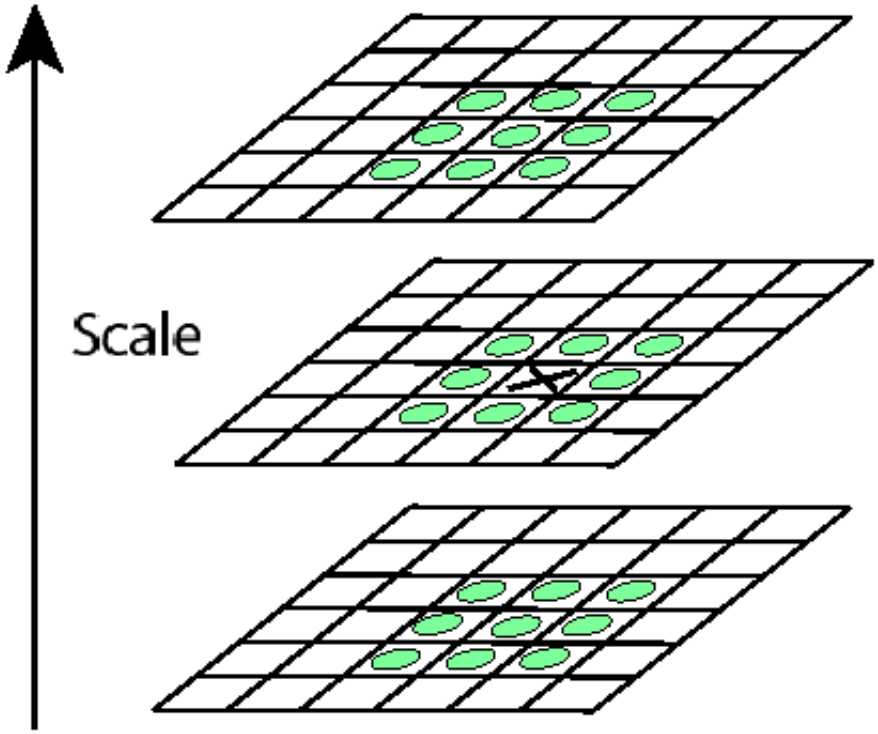
Like to Harris Laplacian but Laplacian of Gaussian kernel is approximated with Difference of Gaussian (DoG) kernel (computationally cheaper):

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



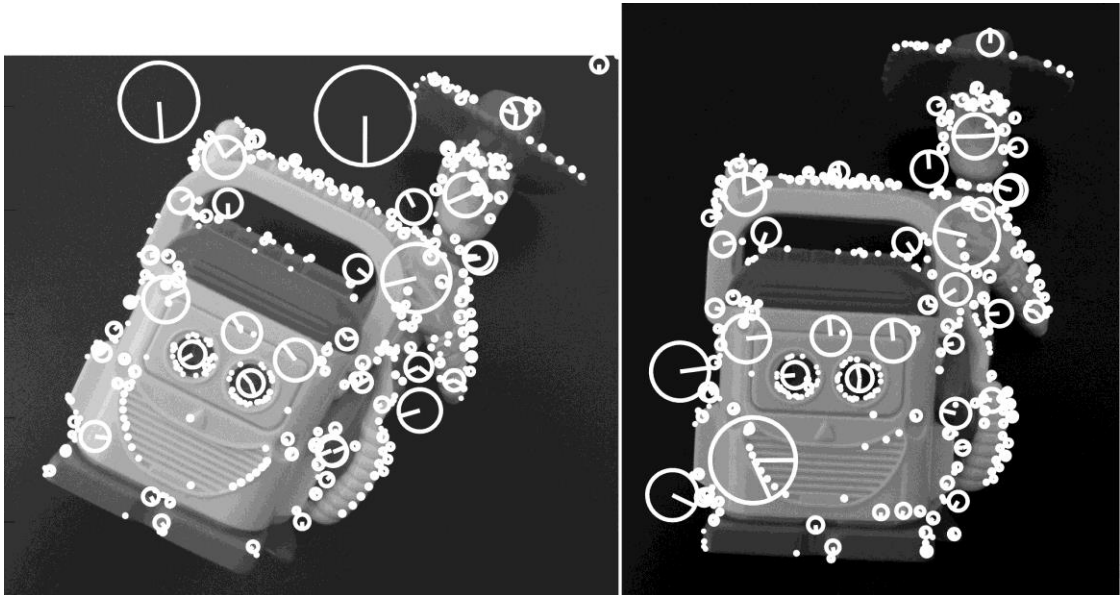
SIFT detector (location + scale)

SIFT keypoints: local extrema in the DoG pyramid



SIFT Features: Summary

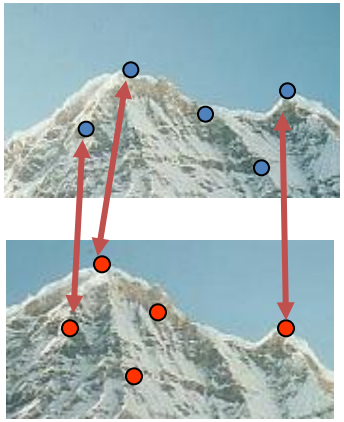
- **SIFT: Scale Invariant Feature Transform** [Lowe, IJCV 2004]
- An approach to **detect and describe** regions of interest in an image.
- SIFT features are **reasonably invariant** to changes in **rotation, scaling**, and **small changes in viewpoint and illumination**
- **Computationally a bit costly (10 Hz)**
  - Expensive steps are the scale detection and descriptor extraction

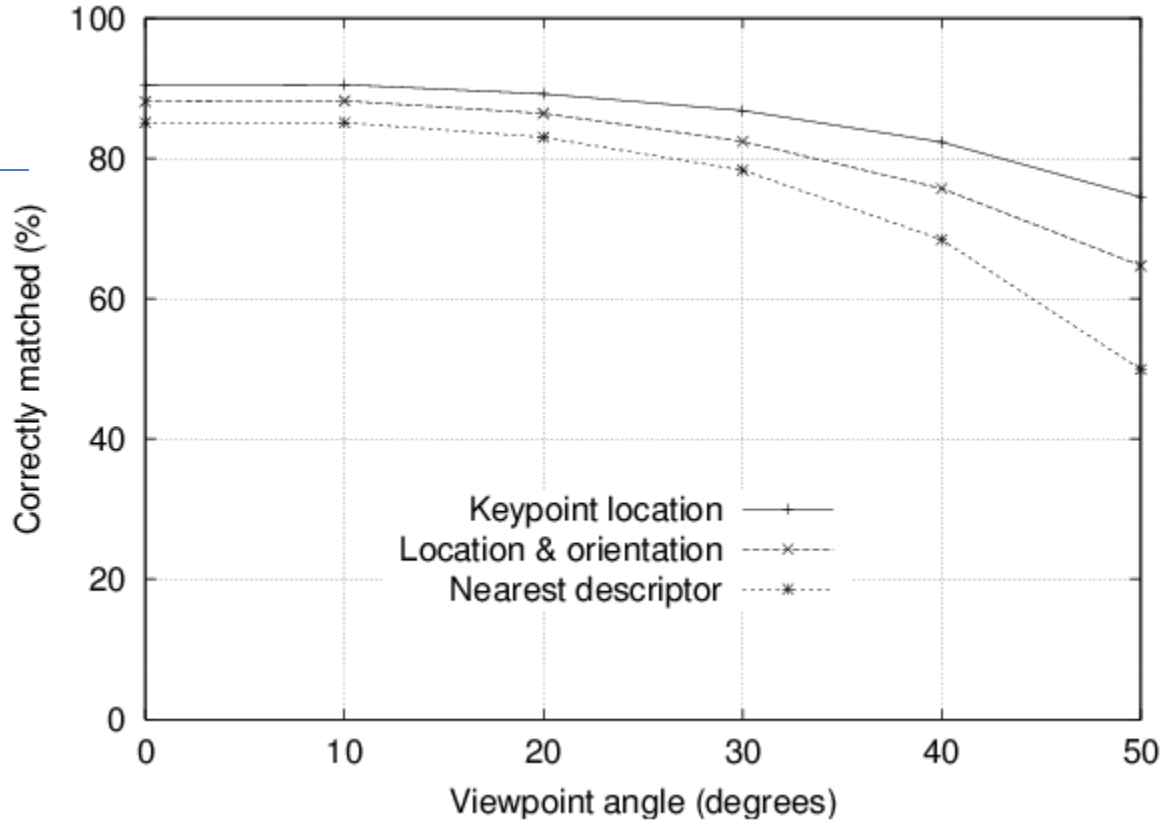


SIFT repeatability vs. viewpoint angle

Repeatability=

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

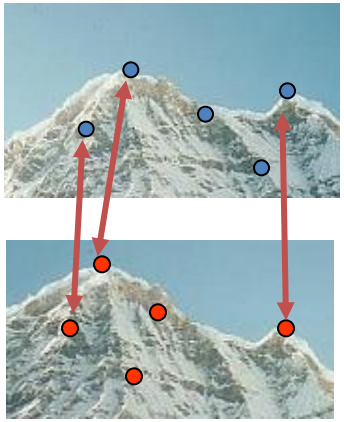


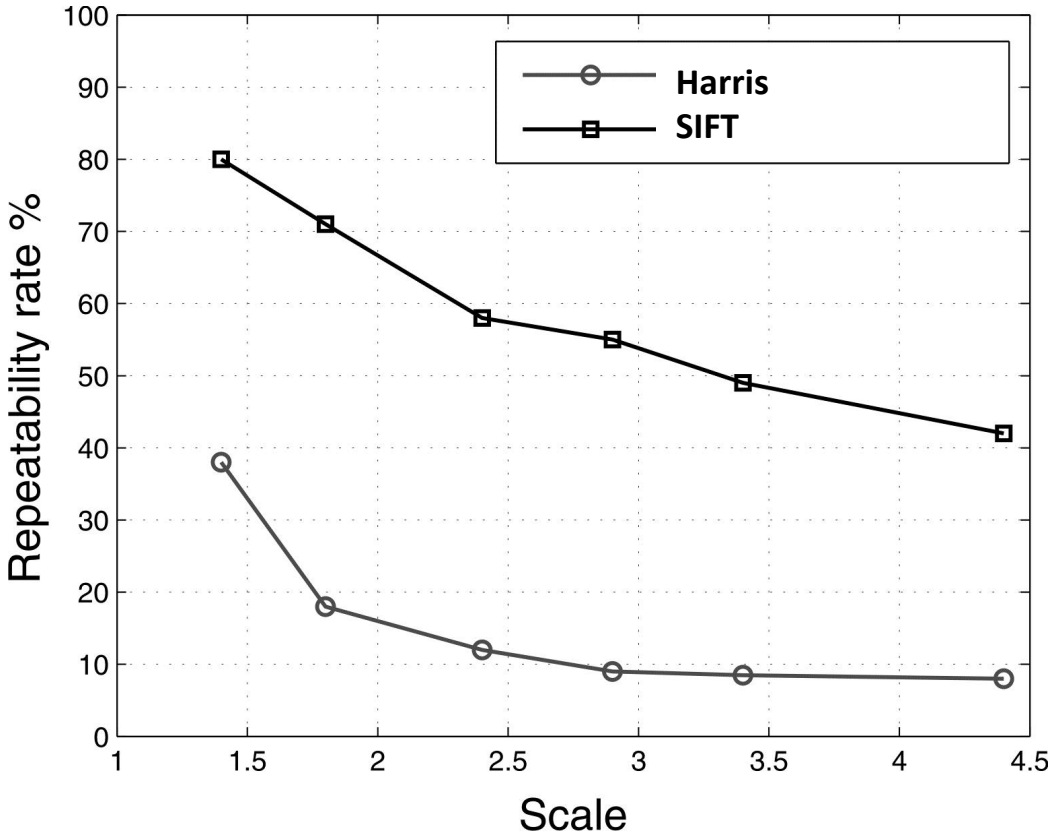


SIFT repeatability vs. Scale

Repeatability=

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



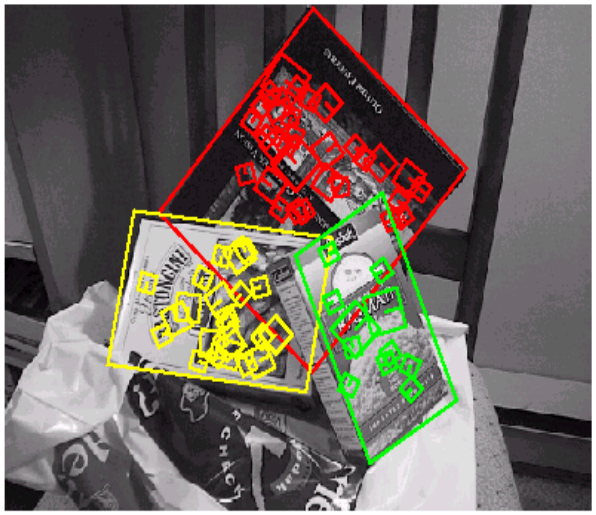


How many parameters are used to define a SIFT feature?

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

SIFT for Planar recognition

- Planar surfaces can be reliably recognized at a rotation of 60° away from the camera
- Only 3 points are needed for recognition
- But objects need to possess enough texture
- Recognition under occlusion



SIFT for Panorama Stitching

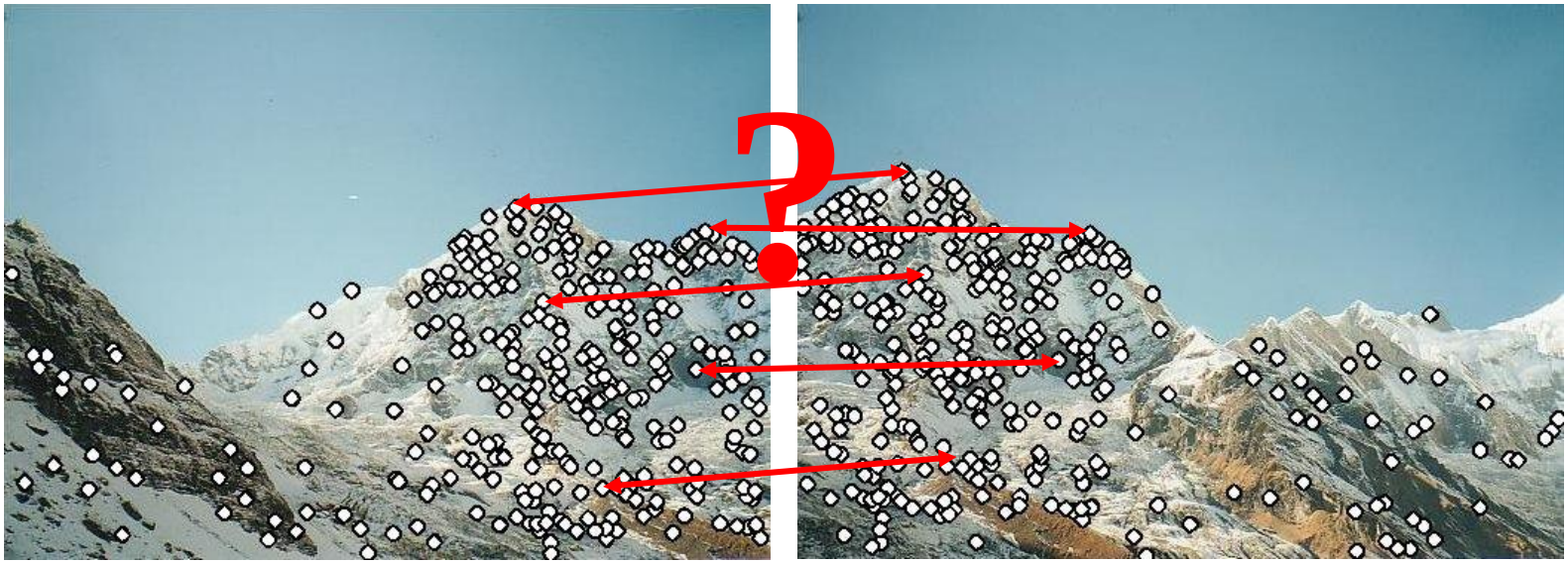
AutoStitch: http://matthewalunbrown.com/autostitch/autostitch.html

[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 (also called L2 norm)
     - SAD
     - ZNCC
  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 to f_1 in I_2
  - f_2' is 2nd best match to f_1 in I_2
  - gives small values for ambiguous matches

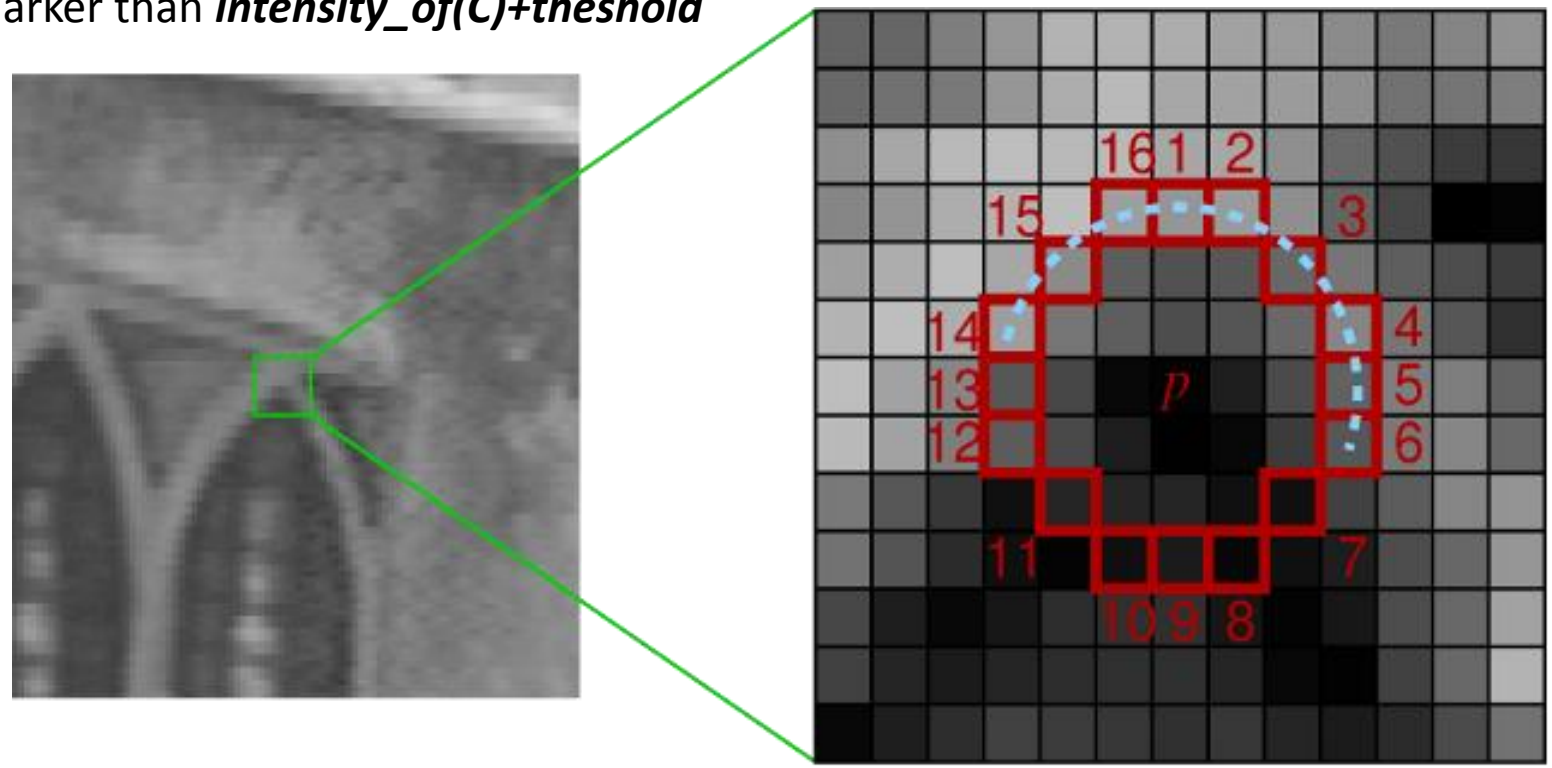
SURF [Bay et al., ECCV 2006]

- ***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

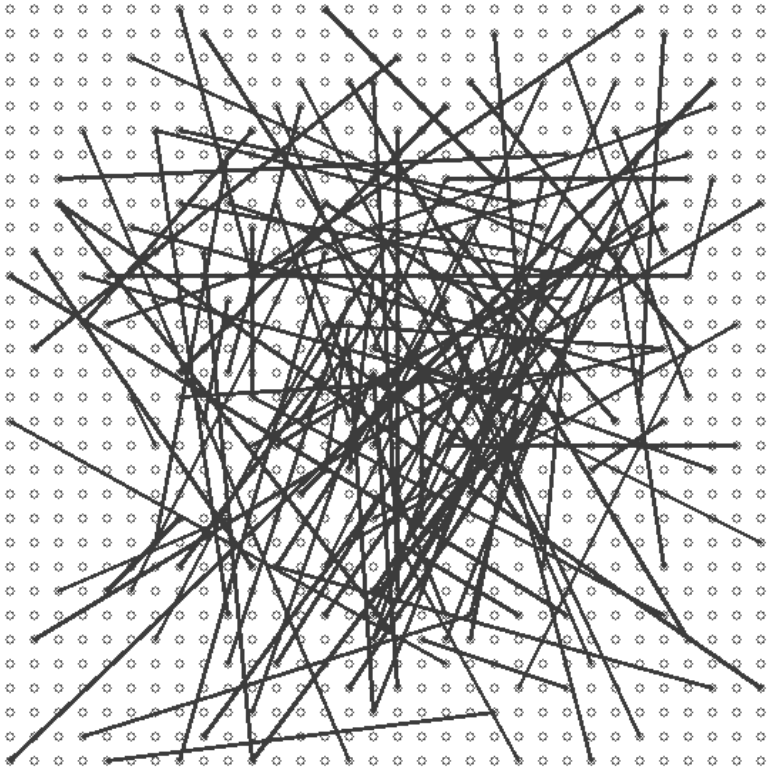
BRIEF descriptor [Calonder et. al, ECCV 2010]

- **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** $p_1 < p_2$ **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

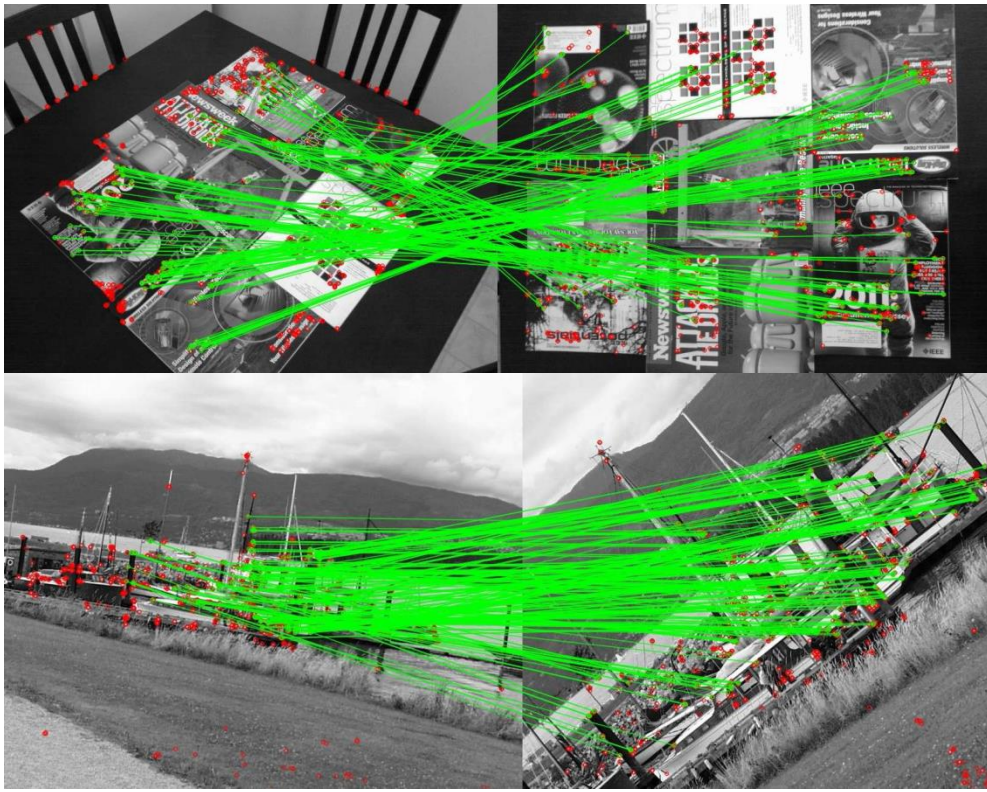


Pattern for intensity pair samples – generated randomly

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

ORB descriptor [Rublee et al., ICCV 2011]

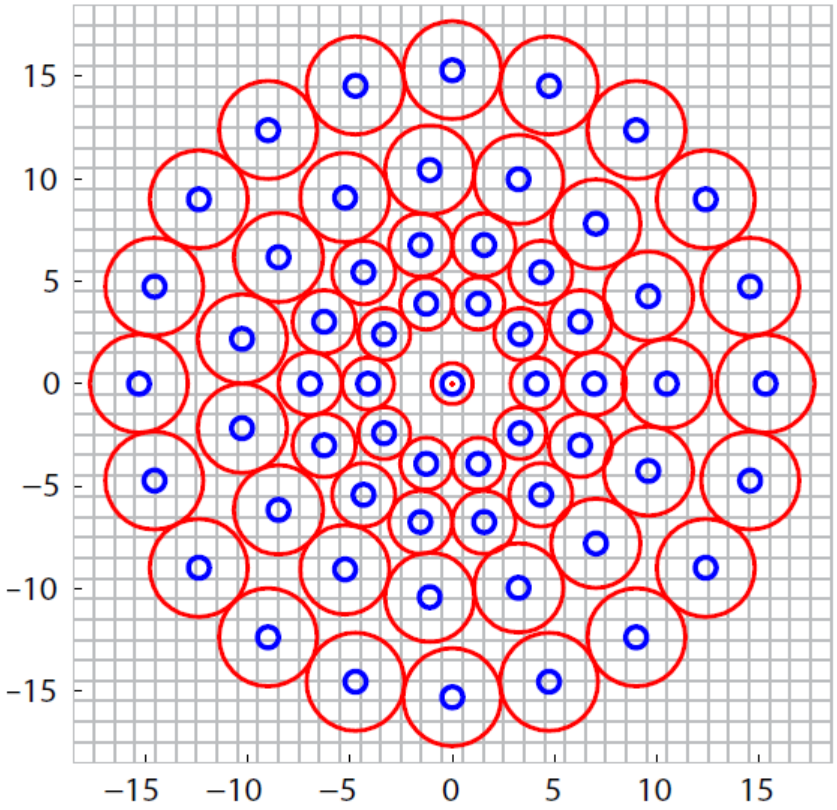
- ***O***riented FAST and ***R***otated ***B***RIEF
- Alterative to SIFT or SURF, designed for fast computation
- Keypoint detector based on **FAST**
- **BRIEF** descriptors are *steered* according to keypoint orientation (to provide rotation invariance)
- Good Binary features are learned by minimizing 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



Summary (things to remember)

- Point feature detection
  - Properties and invariance to transformations
    - Challenges: rotation, scale, view-point, and illumination changes
  - Extraction
    - Harris and Shi-Tomasi
      - Rotation invariance
      - Scale invariance: Harris Laplacian
  - Descriptor
    - Intensity patches
      - How to make them invariant to transformations: rotation, scale, illumination, and view-point (affine)
    - Better solution: Histogram of oriented gradients: SIFT descriptor
  - Matching
    - SSD, SAD, ZNCC, ratio 1^{st} /2^{nd} closest descriptor
  - Depending on the task, you may want to trade repeatability and robustness for speed: approximated solutions, combinations of efficient detectors and descriptors.
    - Fast corner detector: FAST;
    - Keypoint descriptors faster than SIFT: SURF, BRIEF, ORB, BRISK
- **Autonomous Mobile Robot book chapter 4.5**
- **Szeliski book chapters 4.3.2 and 4.1**