University of Zurich[UZH]
ETH zürich
Institute of Informatics – Institute of Neuroinformatics
ROBOTICS & PERCEPTION GROUP

Lecture 05
Point Feature Detection and Matching

Davide Scaramuzza
http://rpg.ifi.uzh.ch

Lab Exercise 3 - Today afternoon

- Room ETH HG E 1.1 from 13:15 to 15:00
- Work description: implement the Harris corner detector and tracker

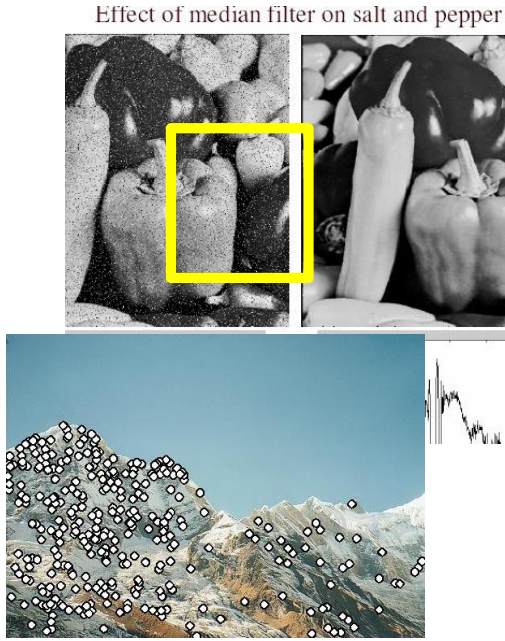
Outline

- Filters for Feature detection
- Point-feature extraction: today and next lecture

Filters for Feature Detection

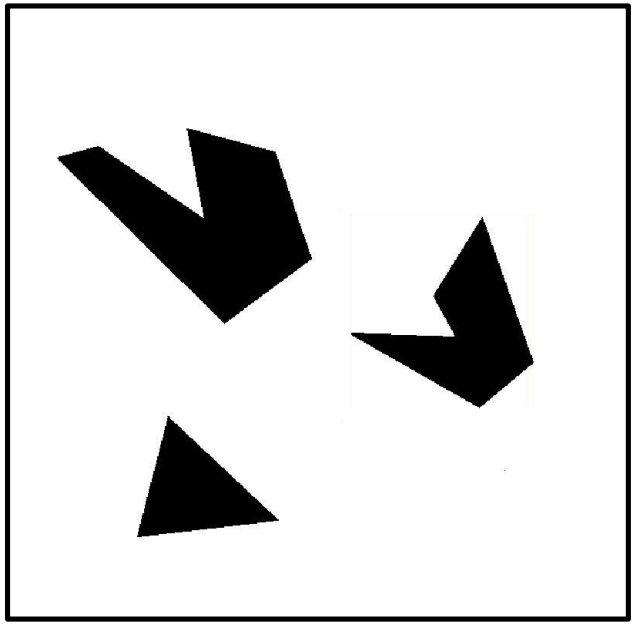
- In the last lecture, we used filters to reduce **noise** or enhance **contours**

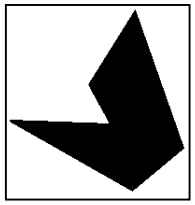
- However, filters can also be used to detect "**features**"
  - Goal: reduce amount of data to process in later stages, discard redundancy to preserve only what is useful (leads to **lower bandwidth and memory storage**)
    - **Edge detection** (we have seen this already; edges can enable line or shape detection)
    - **Template matching**
    - **Keypoint detection**



Filters for Template Matching

- Find locations in an image that are similar to a ***template***
- If we look at filters as **templates**, we can use **correlation** (like convolution but without flipping the filter) to detect these locations

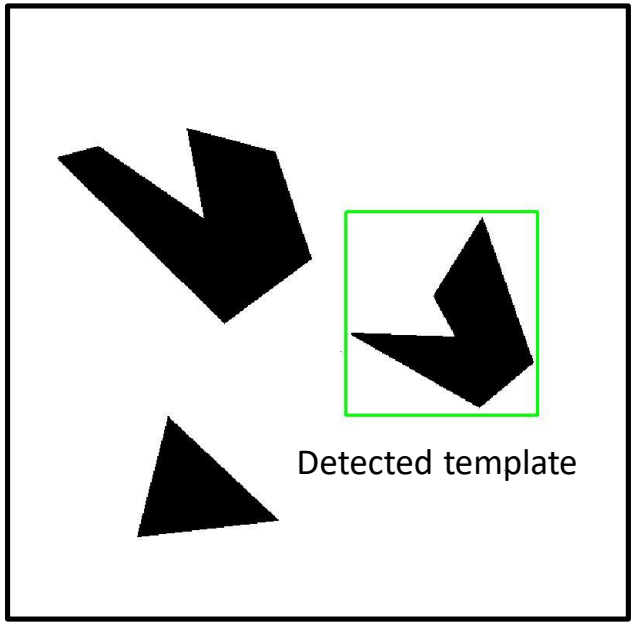


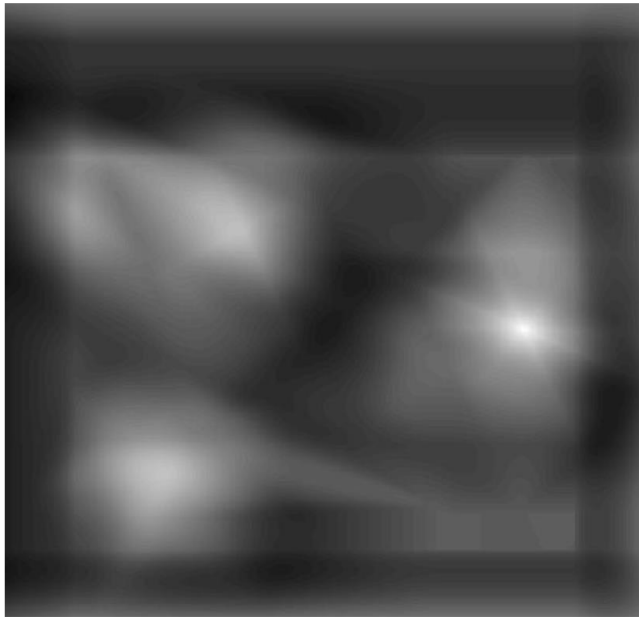


Template

Filters for Template Matching

- Find locations in an image that are similar to a ***template***
- If we look at filters as **templates**, we can use **correlation** (like convolution but without flipping the filter) to detect these locations

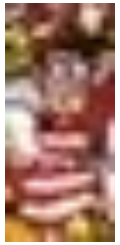




Correlation map

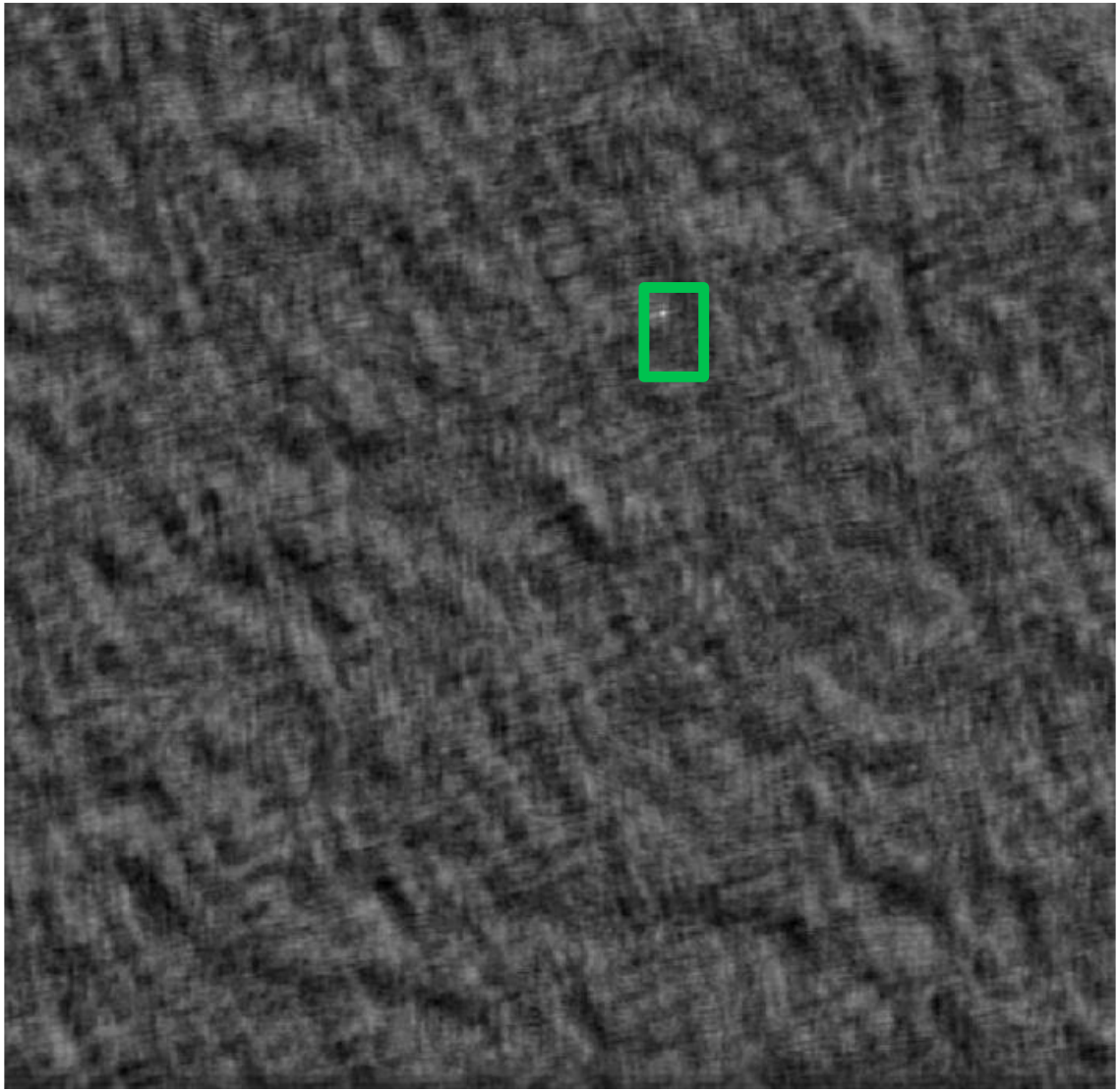
Where’s Waldo?

Scene

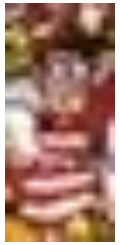


Template

Where’s Waldo?



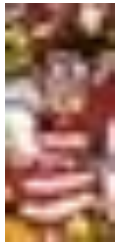
Scene



Template

Where's Waldo?

Scene



Template

Summary of filters

- Smoothing filter:
  - **has positive values**
  - **sums to 1** → preserve brightness of constant regions
  - removes "high-frequency" components: "low-pass" filter
- Derivative filter:
  - **has opposite signs** used to get high response in regions of high contrast
  - **sums to 0** → no response in constant regions
  - **highlights "high-frequency"** components: "high-pass" filter
- Filters as templates
  - Highest response for regions that "*look similar to the filter*"

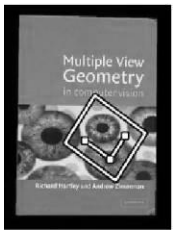
Template Matching

- What if the template is not identical to the object we want to detect?
- Template Matching will only work if **scale**, **orientation**, **illumination**, and, in general, **the appearance of the template and the object to detect are very similar**. What about the pixels in **template background** (*object-background problem*)?

Scene

Template

Scene

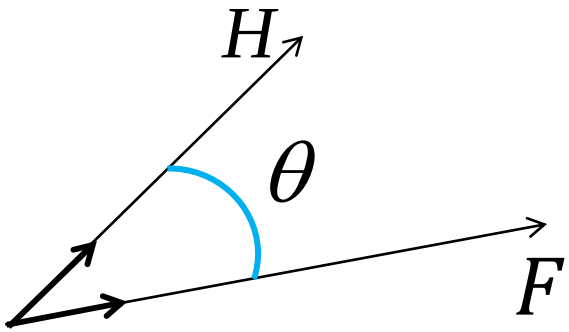


Template

Correlation as Scalar Product

- Consider images H and F as vectors, their correlation is:

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



- In Normalized Cross Correlation (NCC), we consider the unit vectors of H and F, hence we measure their similarity based on the angle θ. If H and F are identical, then NCC = 1.

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

Other Similarity measures

- **Normalized Cross Correlation** (**NCC**): takes values between -1 and +1 (+1 = identical)

$$NCC = \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)F(u,v)}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F(u,v)^2}}$$

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

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

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

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

Zero-mean SAD, SSD, NCC

To account for the difference in the average intensity of two images (typically caused by additive illumination changes), we subtract the mean value of each image:

$$\mu_H = \frac{1}{N}\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v) \qquad \mu_F = \frac{1}{N}\sum_{u=-k}^{k}\sum_{v=-k}^{k} F(u,v)$$

N is the number of pixels of H or F

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

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

ZNCC is invariant to affine intensity changes: $I'(x,y) = \alpha I(x,y) + \beta$

- **Zero-mean Sum of Squared Differences (ZSSD)**

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

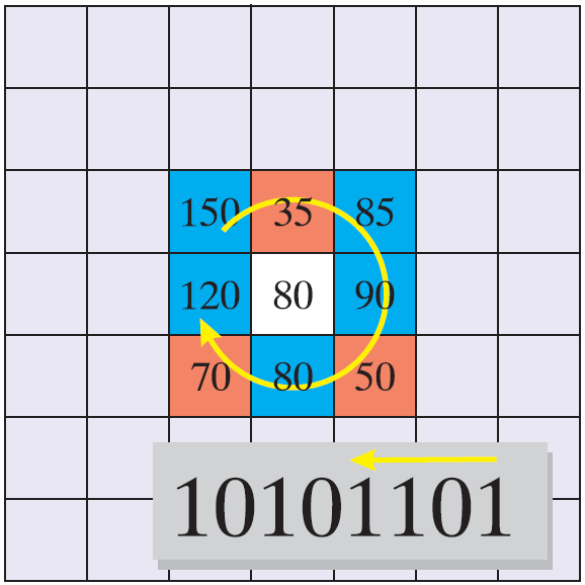
- **Zero-mean Sum of Absolute Differences (ZSAD)** (used in optic mice)

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

Are these invariant to affine illumination changes?

Census Transform

- Maps an image patch to a bit string:
  - **if a pixel is greater than the center pixel** its corresponding bit is set to **1**, else to 0
  - For a $w \times w$ window the string will be $\boldsymbol{w^2 - 1}$ **bits long**
- The two bit strings are **compared using the Hamming distance**, which is the number of bits that are different. This can be computed by counting the number of 1s in the Exclusive-OR (XOR) of the two bit strings



Advantages

- **No square roots or divisions** are required, thus very efficient to implement, especially on FPGA
- Intensities are considered relative to the center pixel of the patch making it **invariant to monotonic intensity** changes

Outline

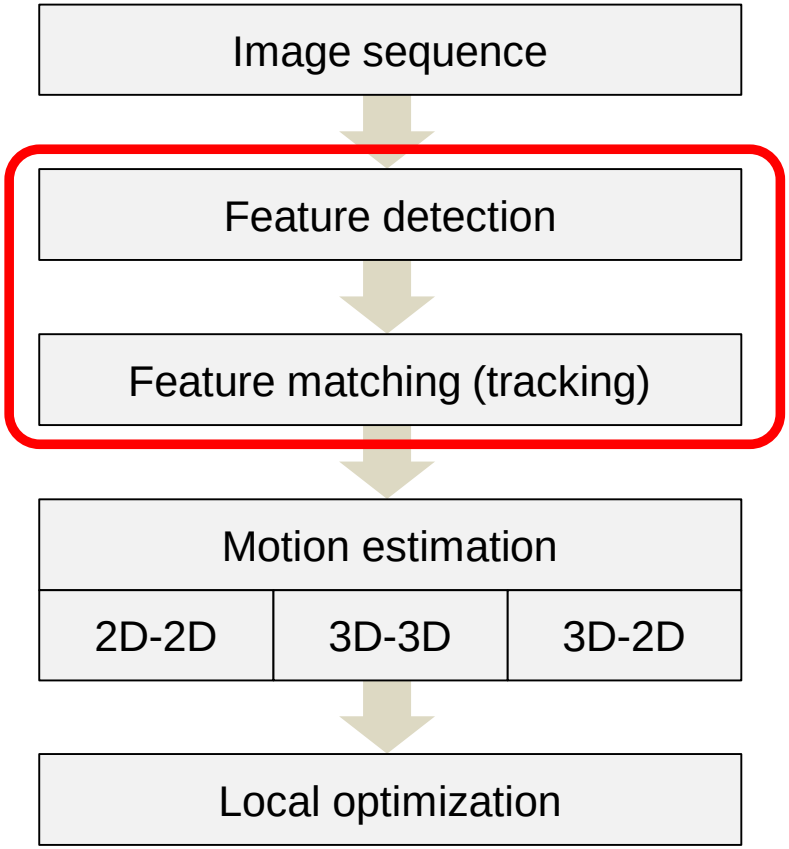
- Filters for feature extraction
- Point-feature (or keypoint) extraction: today and next lecture

Keypoint extraction and matching - Example

Video from “Forster, Pizzoli, Scaramuzza, *SVO: Semi-Direct Visual Odometry*, ICRA’14”
https://www.youtube.com/watch?v=2YnIMfw6bJY

Why do we need keypoints?

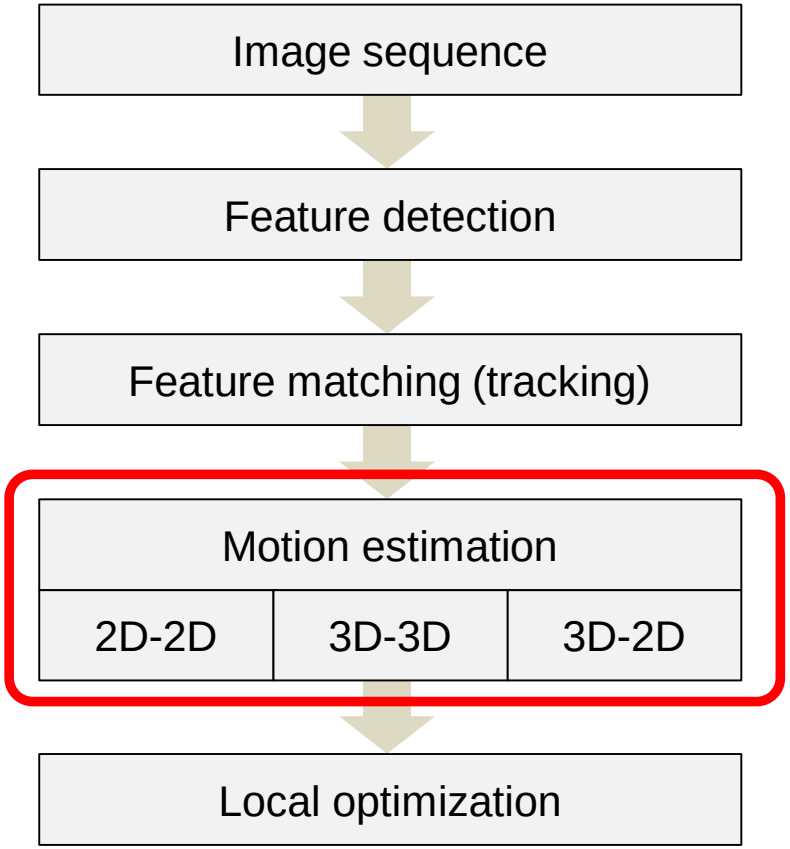
Recall the Visual-Odometry flow chart:

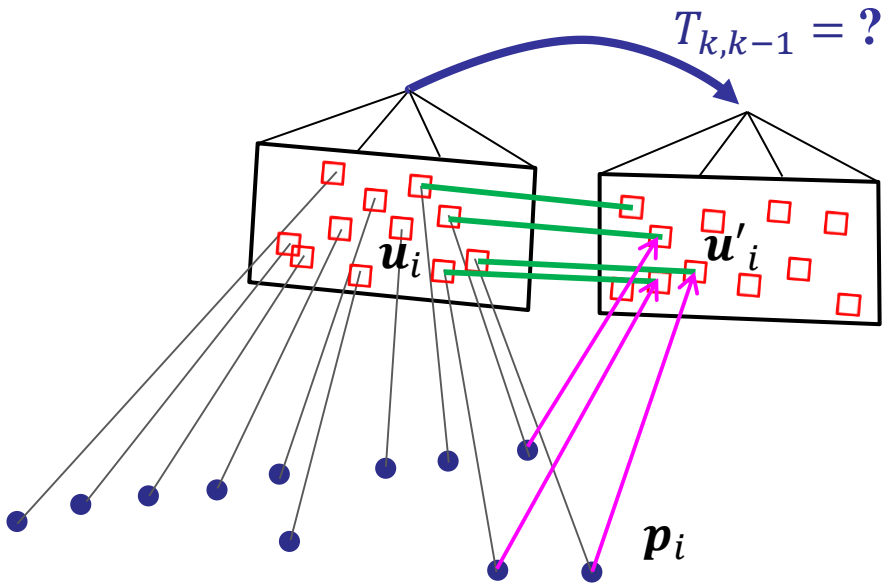


Example of feature tracks

Why do we need keypoints?

Keypoint extraction is the key ingredient of motion estimation!





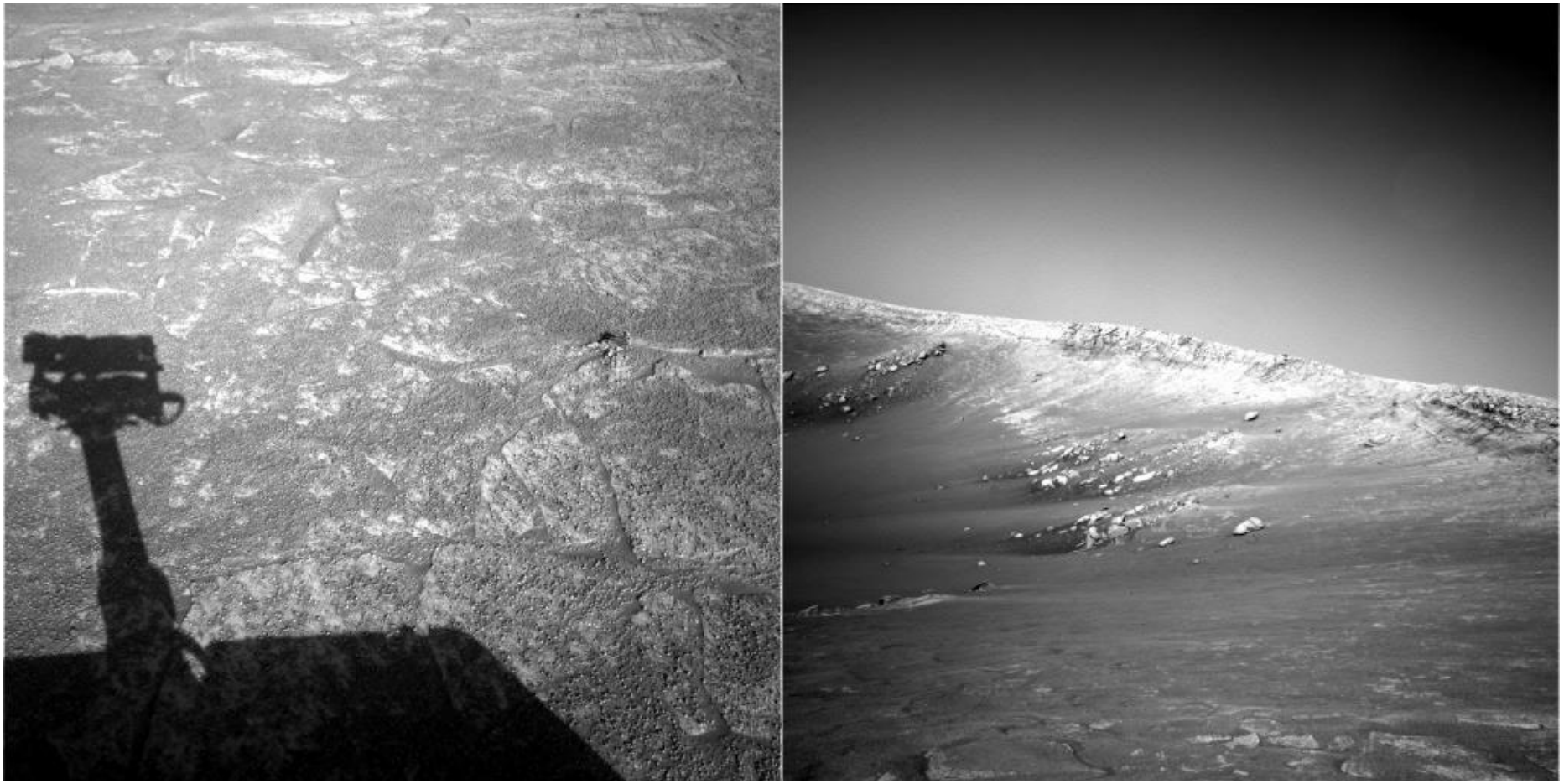
Keypoints are also used for:

- Panorama stitching
- Object recognition
- 3D reconstruction
- Place recognition
- Indexing and database retrieval (e.g., Google Images or http://tineye.com)

These problems go under the name of
Image Matching problem:
finding similar keypoints between two images of the same scene
taken under different conditions

Image matching: why is it challenging?

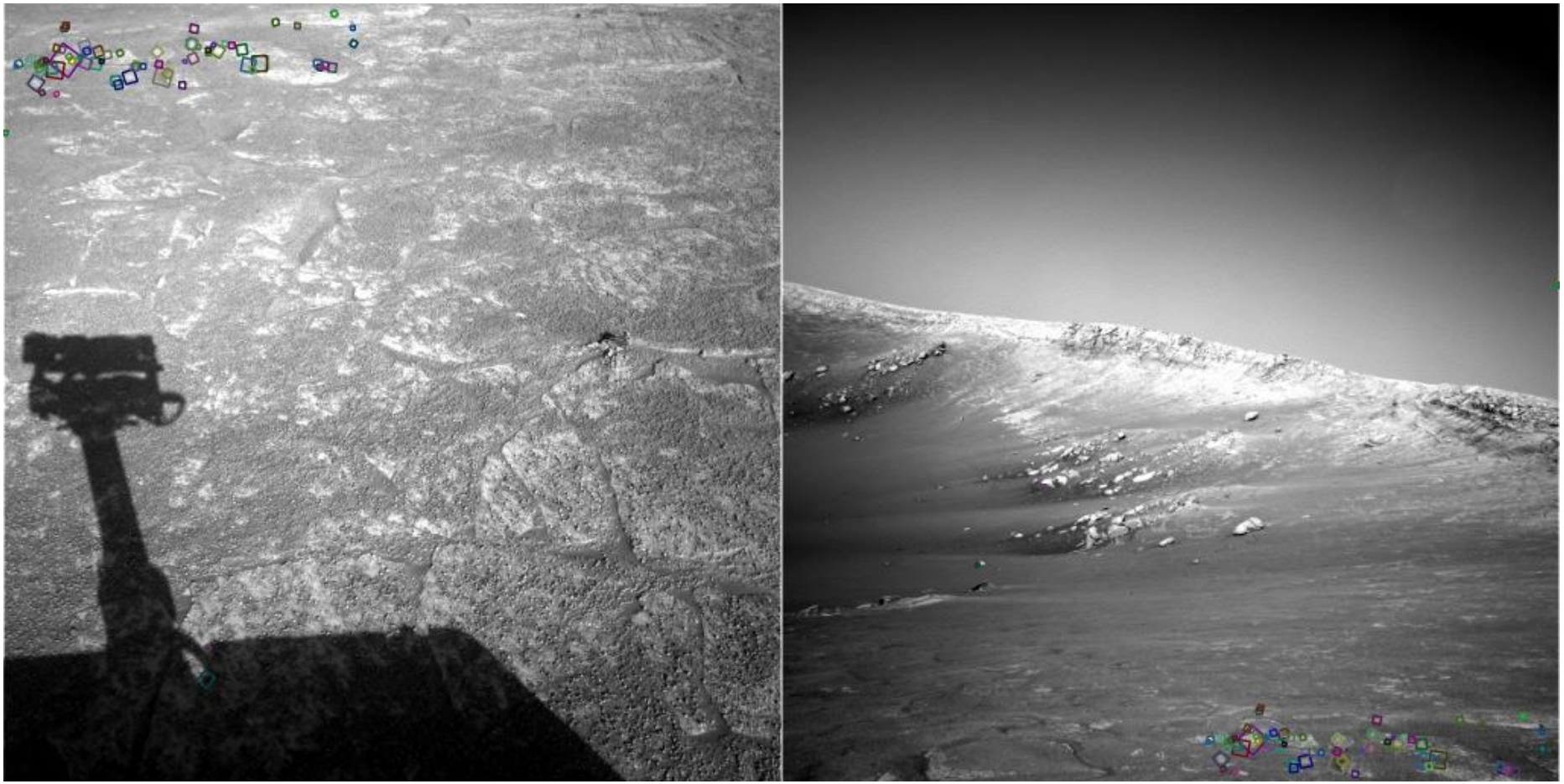
Image matching: why is it challenging?



NASA Mars Rover images

Image matching: why is it challenging?

Answer below



NASA Mars Rover images with SIFT feature matches

Example: panorama stitching

This panorama was generated using AUTOSTITCH:
http://matthewalunbrown.com/autostitch/autostitch.html

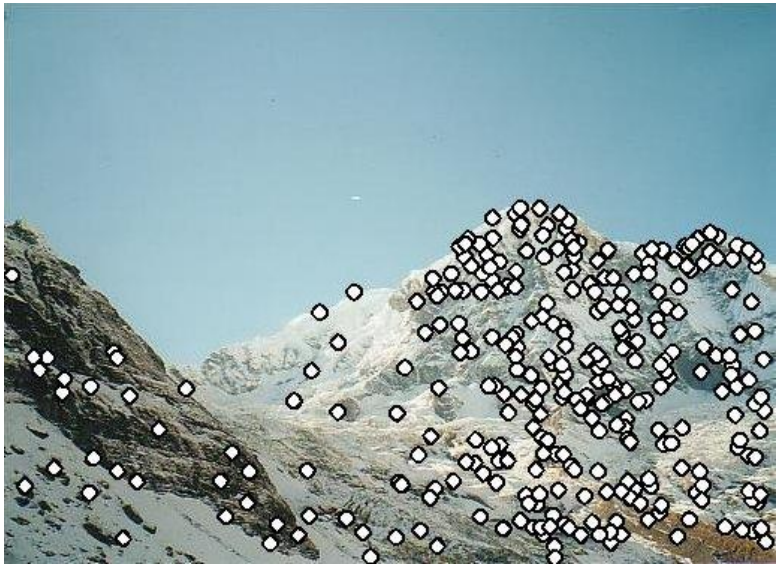
How does it work?

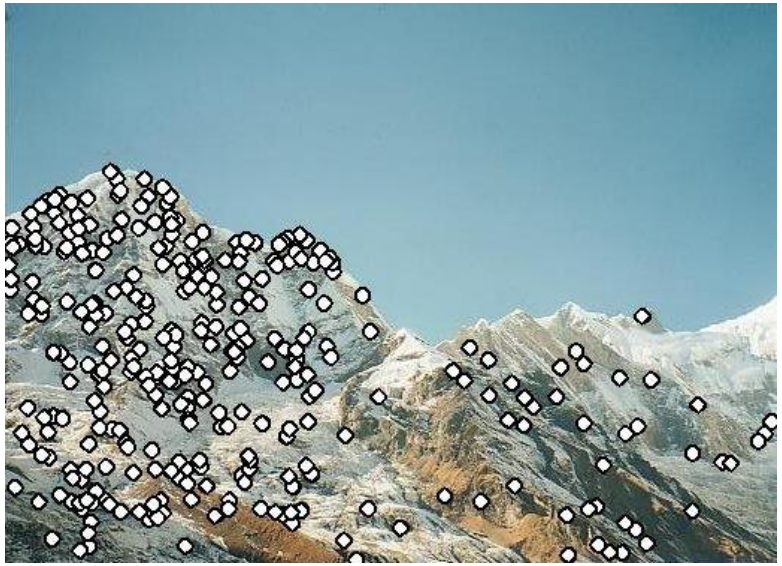
Local features and alignment

- We need to align images
- How would you do it?

Local features and alignment

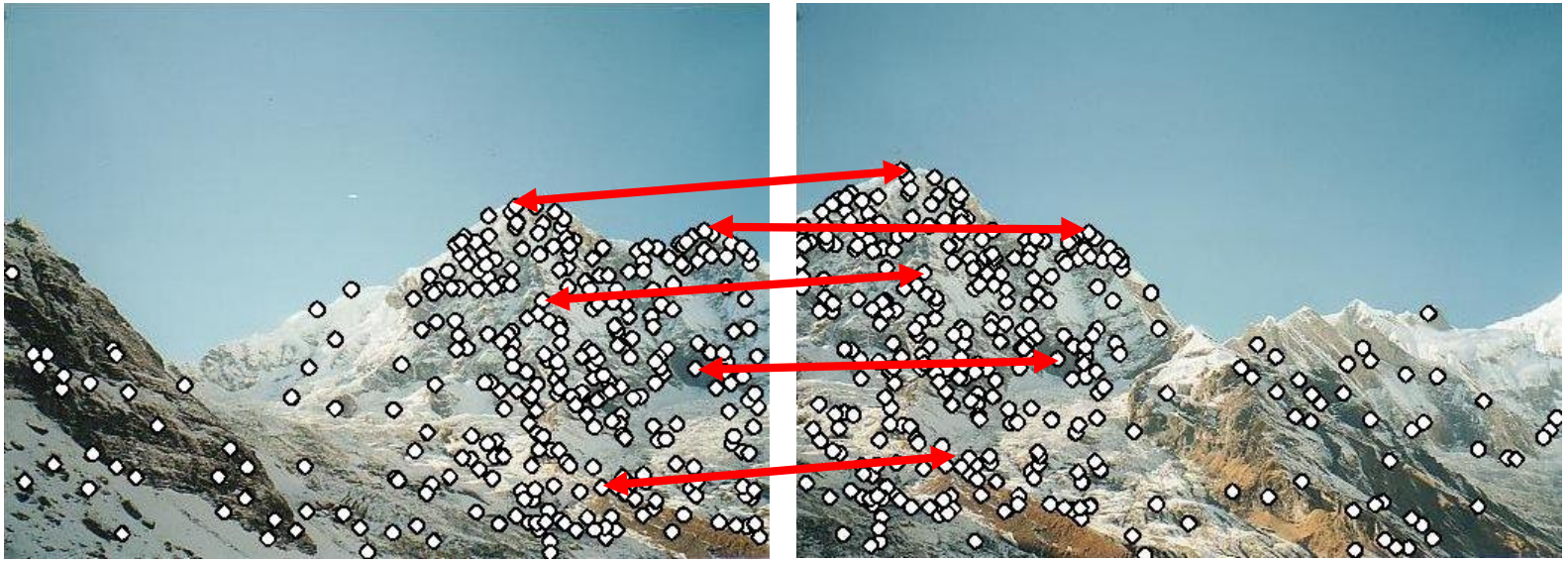
- Detect point features in both images





Local features and alignment

- Detect point features in both images
- Find corresponding pairs

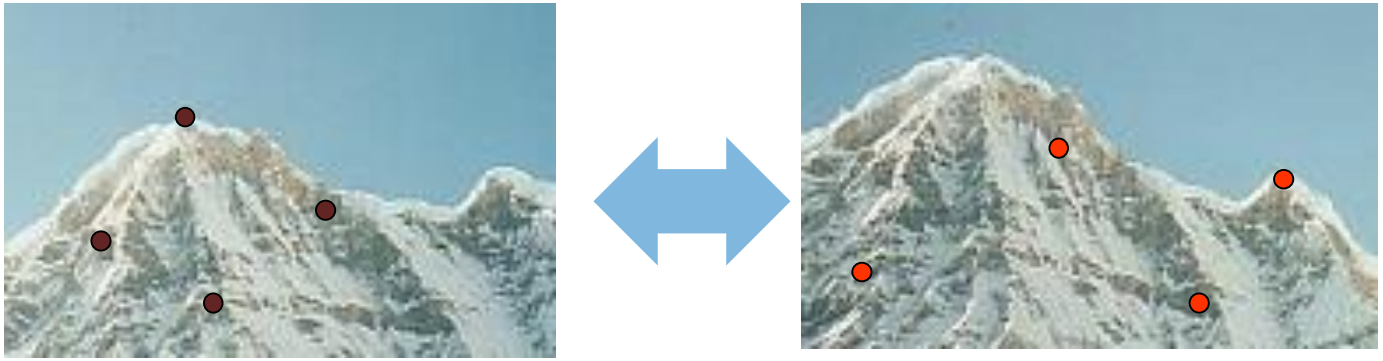


Local features and alignment

- Detect point features in both images
- Find corresponding pairs
- Use these pairs to align the images

Matching with Features

- Problem 1:
  - Detect the **same** points **independently** in both images



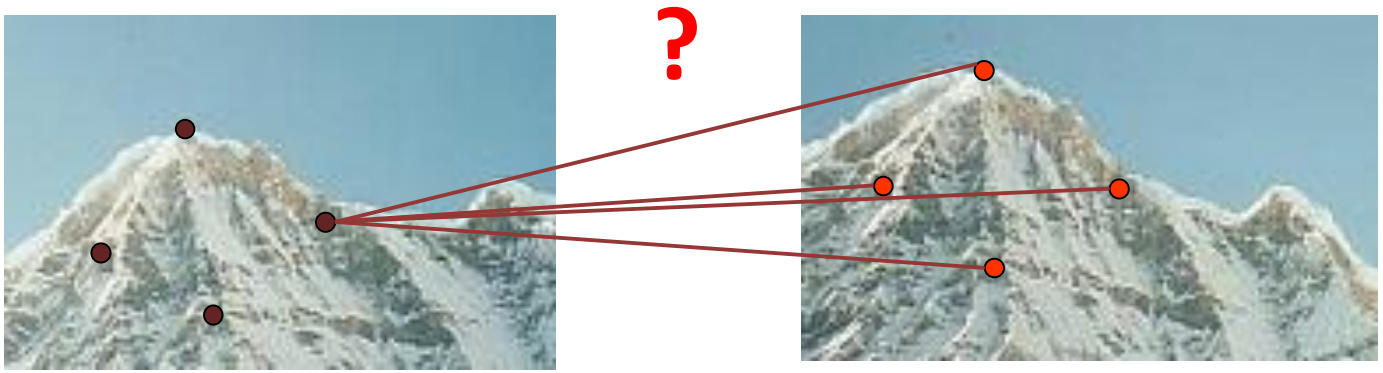
no chance to match!

We need a **repeatable** feature **detector.** Repeatable means that the detector should be able to re-detect the same feature in different images of the same scene.

This property is called **Repeatability** of a feature **detector**.

Matching with Features

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



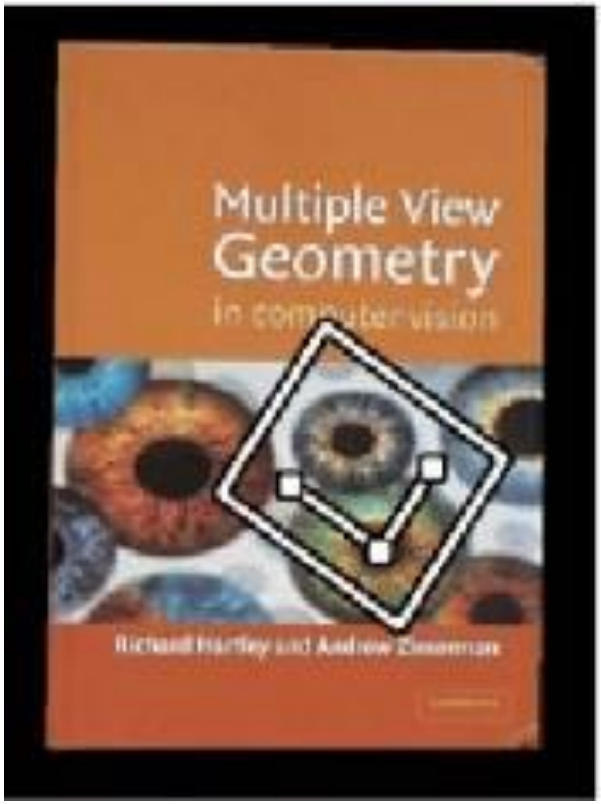
We need a **distinctive** feature descriptor. A descriptor is a "description" of the pixel information around a feature (e.g., patch intensity values, gradient values, etc.). Distinctive means that the descriptor uniquely identifies a feature from its surrounding without ambiguity.

This property is called **Distinctiveness** of a feature **descriptor**.

The descriptor must also be **robust to *geometric* and *illumination*** changes.

Geometric changes

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



Illumination changes

Typically, small illumination changes are modelled with an affine transformation (so called *affine illumination changes)*:

$$I'(x,y) = \alpha I(x,y) + \beta$$

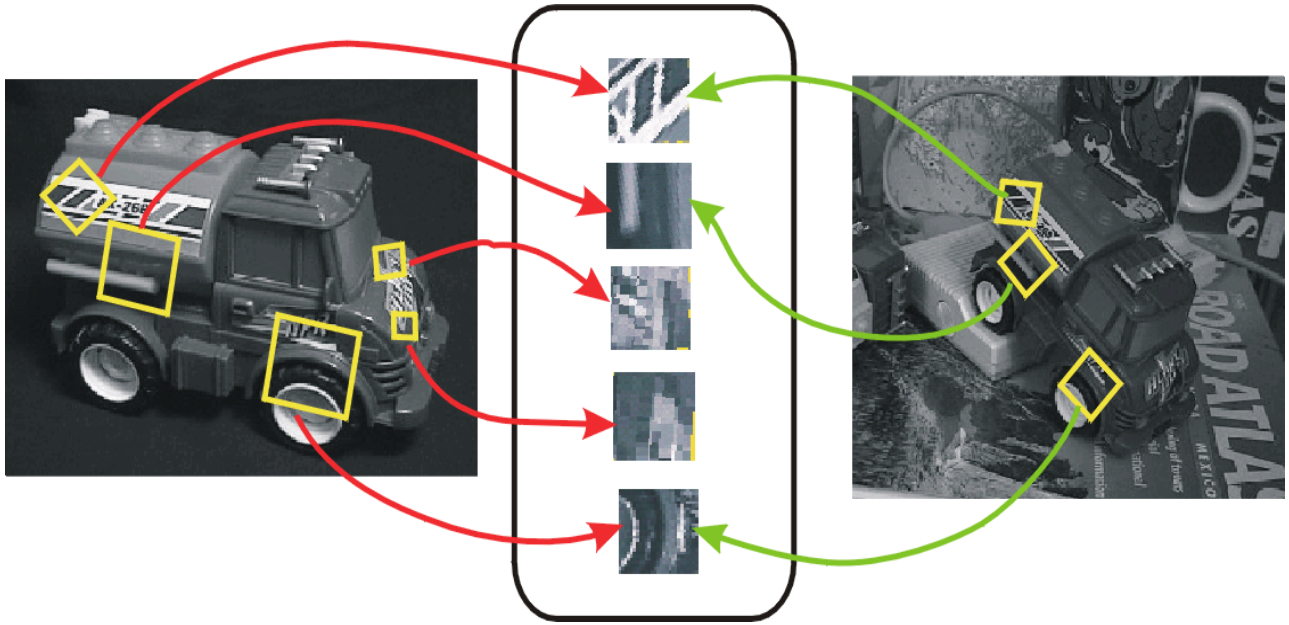
Invariant local features

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

Basic steps:

1) Detect repeatable and distinctive interest points

2) Extract invariant descriptors

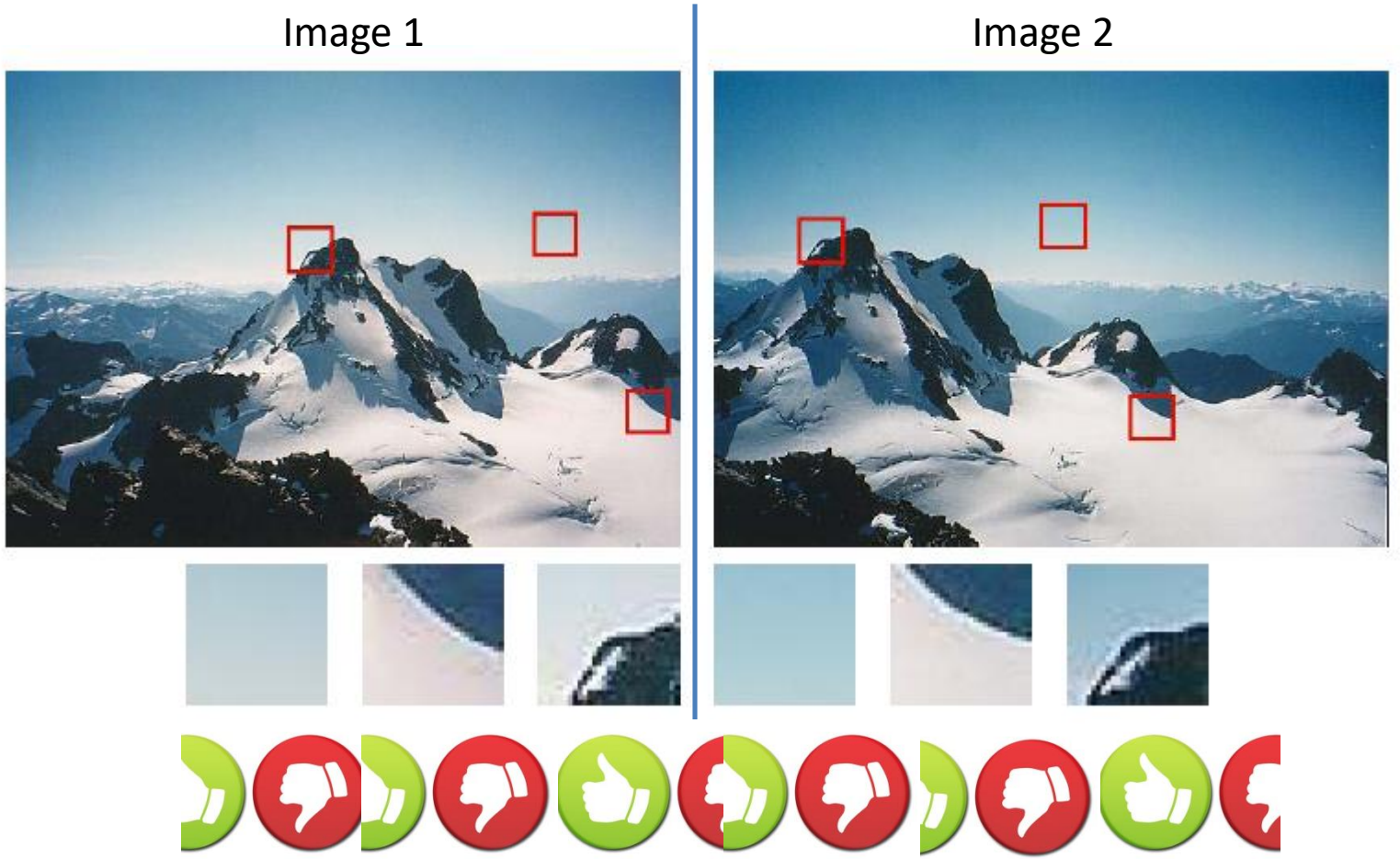


Main questions

- What features are *repeatable* and *distinctive?*
- How to *describe* a feature?
- How to establish *correspondences*, i.e., compute matches?

What is a Repeatable & Distinctive feature?

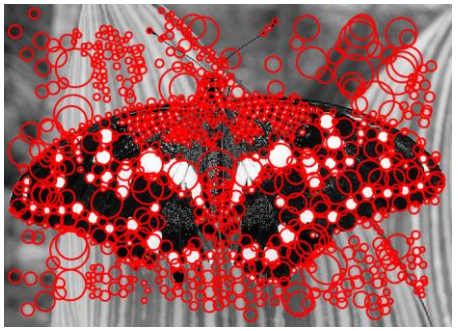
- Consider the image pair below with extracted patches
- Notice how some patches can be localized or matched with higher accuracy than others



Point Features: Corners vs Blob detectors

- A **corner** is defined as the intersection of one or more edges
  - Corner have high localization accuracy
    → Corners are good for VO
  - Corners are **less distinctive than blobs**
  - E.g., *Harris, Shi-Tomasi, SUSAN, FAST*

- A **blob** is any other image pattern **that is not a corner** and differs significantly from its neighbors (e.g., a connected region of pixels with similar color, a circle, etc.)
  - **Blobs have less localization accuracy than corners**
  - Blobs are **more distinctive than a corner**
    → blobs **are better for place recognition**
  - E.g., *MSER, LOG, DOG (SIFT), SURF, CenSurE, BRIEF, BRISK, ORB, FREAK, etc.*

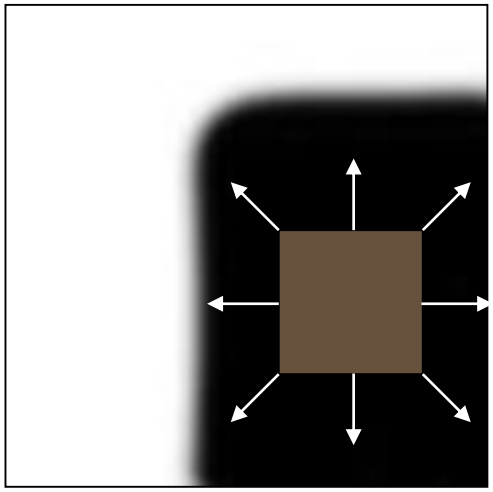


Corner detection

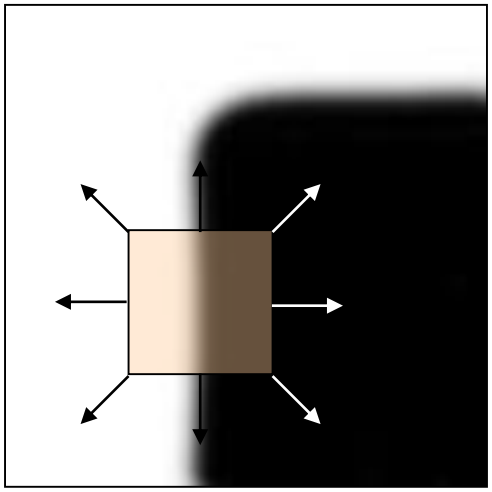
- Key observation: in the region around a corner, image gradient has **two or more** dominant directions
- Corners are **repeatable** and **distinctive**

The Moravec Corner detector (1980)

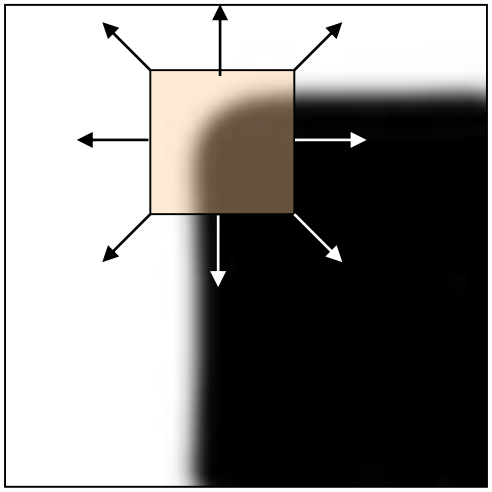
- How do we identify corners?
- Look at a region of pixels through a small window
- Shifting a window in **any direction** should give a **large intensity changes** (e.g., in SSD) in at least 2 directions



"flat" region:
no intensity change
(i.e., SSD ≈ 0 in all directions)



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



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

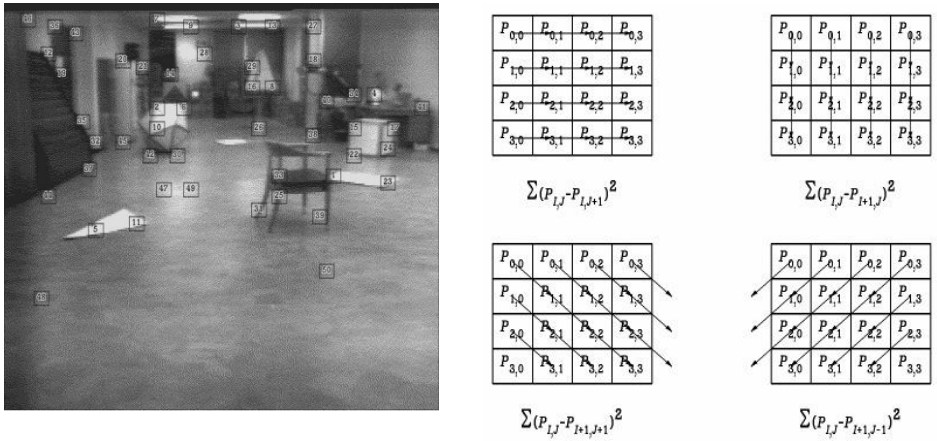
H. Moravec, *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*, PhD thesis, Chapter 5, Stanford University, Computer Science Department, 1980.

The Moravec Corner detector (1980)

Consider the reference patch centered at (x, y) and the shifted window centered at $(x + \Delta x, y + \Delta y)$. The patch has size P. The Sum of Squared Differences between them is:

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

"Sums of squares of differences of pixels adjacent in each of four directions (horizontal, vertical and two diagonals) over each window are calculated, and the window's interest measure is the minimum of these four sums. Features are chosen where the interest measure has local maxima." [Moravec'80, PhD thesis, Chapter 5, link]



H. Moravec, *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*, PhD thesis, Chapter 5, Stanford University, Computer Science Department, 1980.

The Harris Corner detector (1988)

- It implements the Moravec corner detector without having to physically shift the window but rather by just looking at the patch itself, by using differential calculus.

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

How do we implement this?

- Consider the reference patch centered at (x, y) and the shifted window centered at $(x + \Delta x, y + \Delta y)$. The patch has size P.

- The Sum of Squared Differences between them is:

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

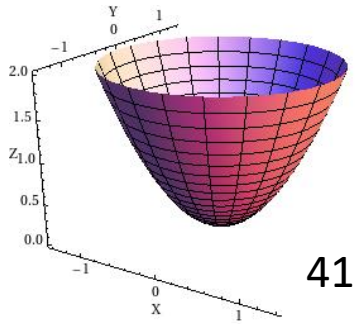
- Let $I_x = \frac{\partial I(x, y)}{\partial x}$ and $I_y = \frac{\partial I(x, y)}{\partial y}$. Approximating with a 1st order Taylor expansion:

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

- This produces the approximation

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

This is a simple quadratic function in two variables $(\Delta x, \Delta y)$



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

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

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

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

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

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

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

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

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

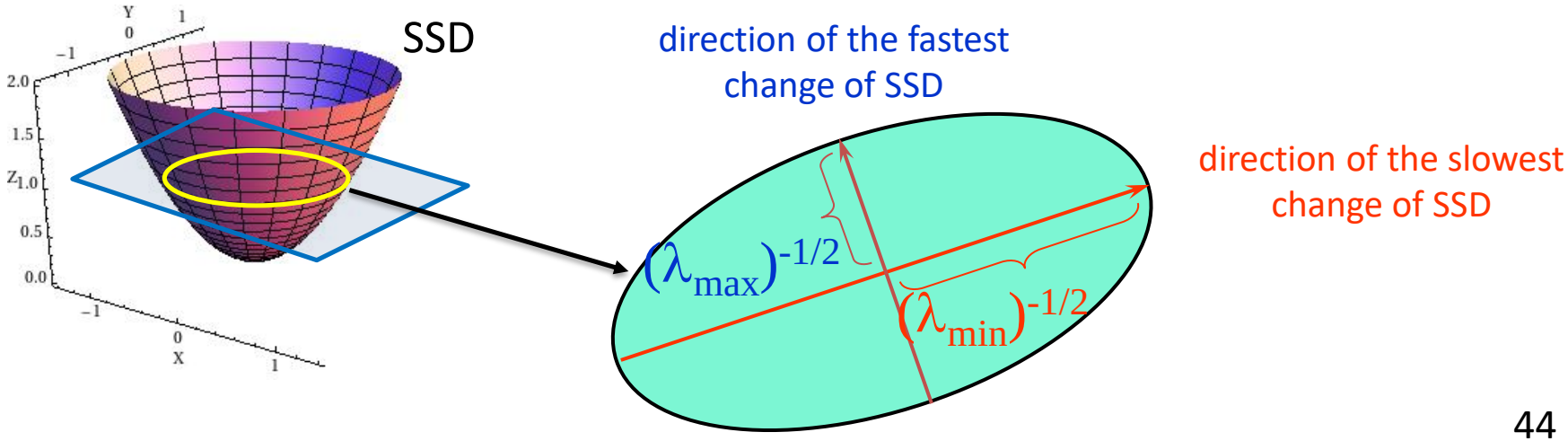
2nd moment matrix Alternative way to write M

What does this matrix reveal?

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

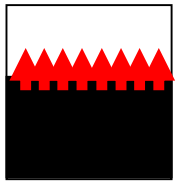
- We can visualize $[\Delta x \quad \Delta y] M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = const$ as an ellipse with axis lengths determined by the ***eigenvalues*** and the two axes' orientations determined by *R* (i.e., the **eigenvectors** of M)

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



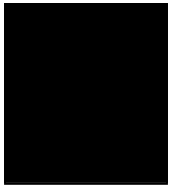
Example

- First, consider an edge or a flat region.



Edge

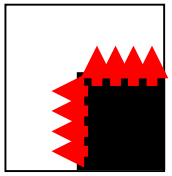
$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



Flat region

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

- We can conclude that if at **least one of the eigenvalues** λ is **close to 0**, then this is **not a corner.**
- Now, let's consider an axis-aligned corner:



Corner

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ -\sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix}$$

- We can observe that the dominant gradient directions are at 45 degrees with x and y axes
- We can also conclude that if **both two eigenvalues are much larger than 0**, then **this is a corner**

How to compute $\lambda_1, \lambda_2, \mathrm{R}$ from M

Eigenvalue/eigenvector review

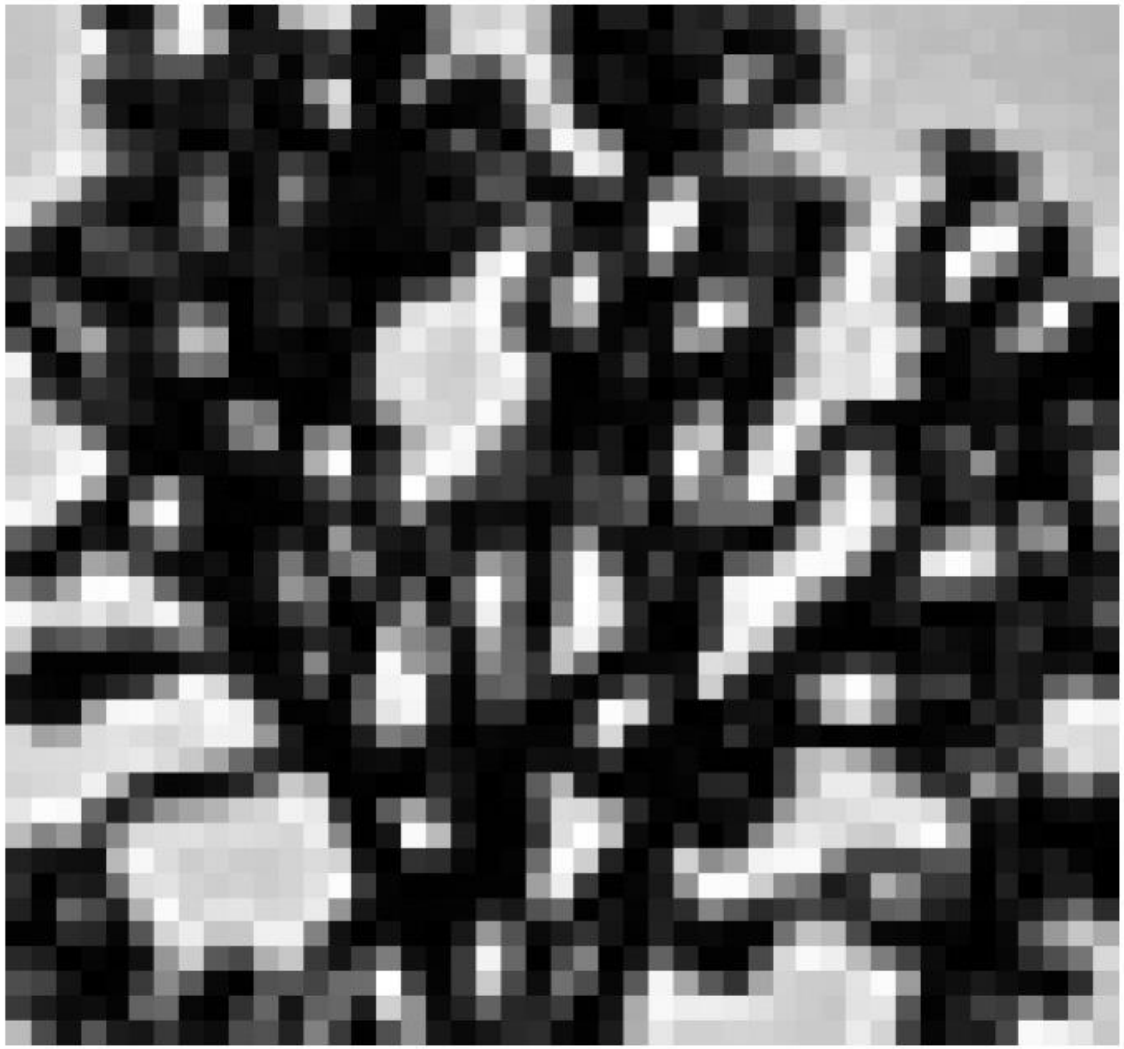
- You can easily prove that λ_1, λ_2 are the **eigenvalues** of M.
- The **eigenvectors** and **eigenvalues** of a square matrix **A** are the vectors **x** and scalars λ that satisfy:

$$Ax = \lambda x$$

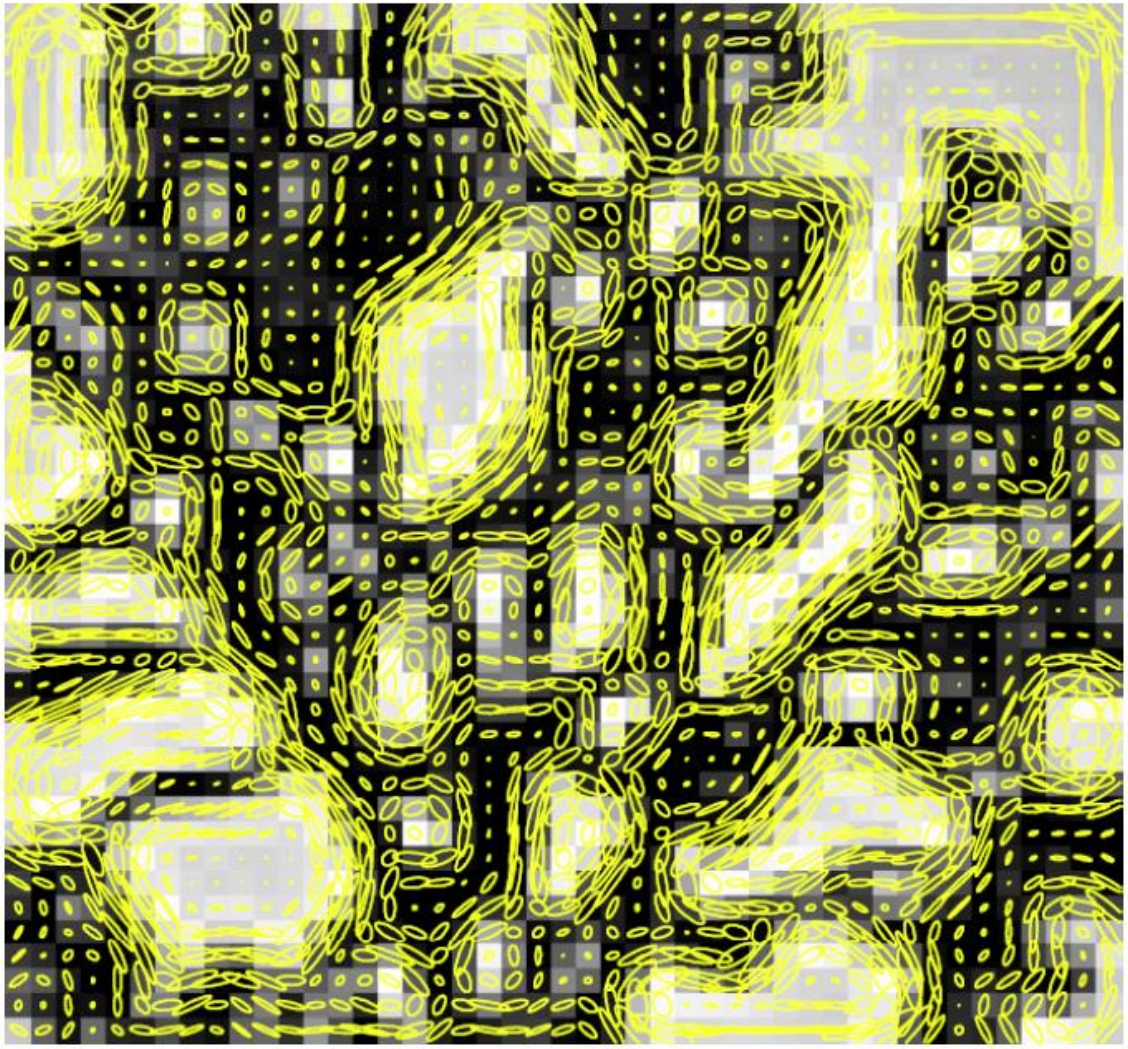
- The scalar λ is the **eigenvalue** corresponding to **x**
  - The eigenvalues are found by solving: $\det(A - \lambda I) = 0$
  - In our case, ***A*** = ***M*** is a 2x2 matrix, so we have $\det\begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix} = 0$
  - The solution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



NB: the ellipses here are plotted proportionally to the eigenvalues and not as iso-SSD ellipses as explained before. So small ellipses here denote a flat region, and big ones, a corner.

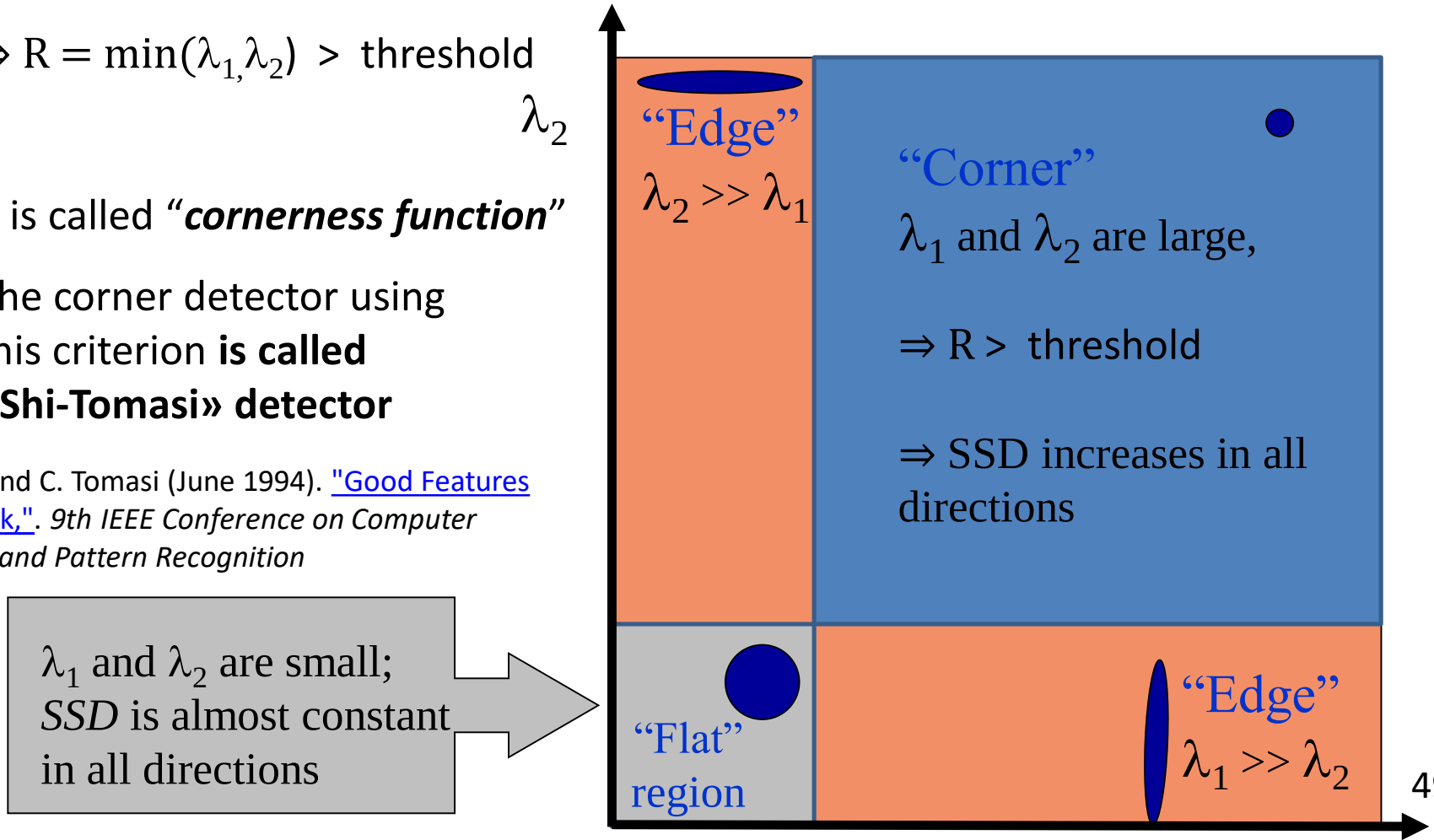
Interpreting the eigenvalues

- 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

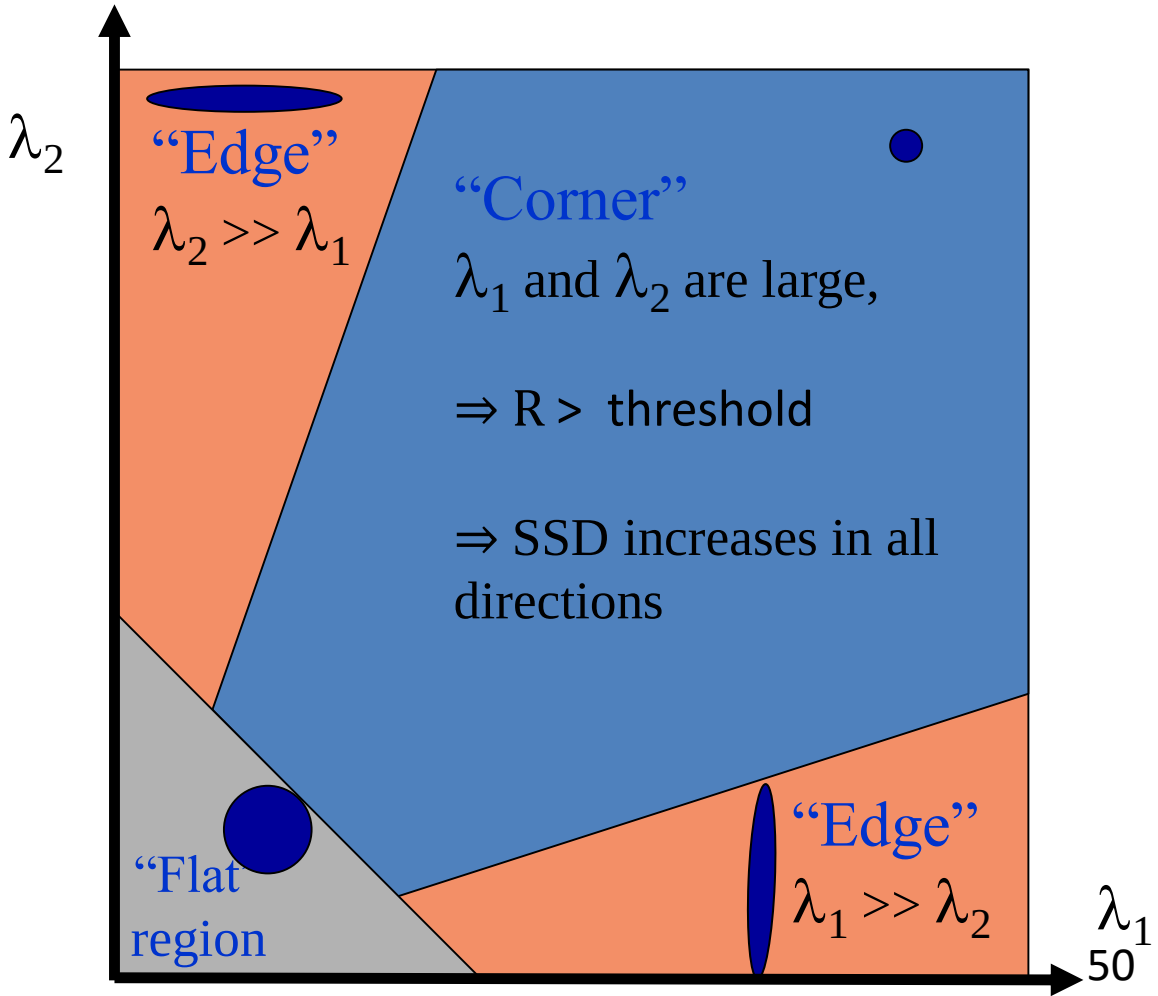
- Computation of λ_1 and λ_2 is expensive ⇨ Harris & Stephens suggested using a **different cornerness function**:

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

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

- The corner detector using this criterion **is called «Harris» detector**

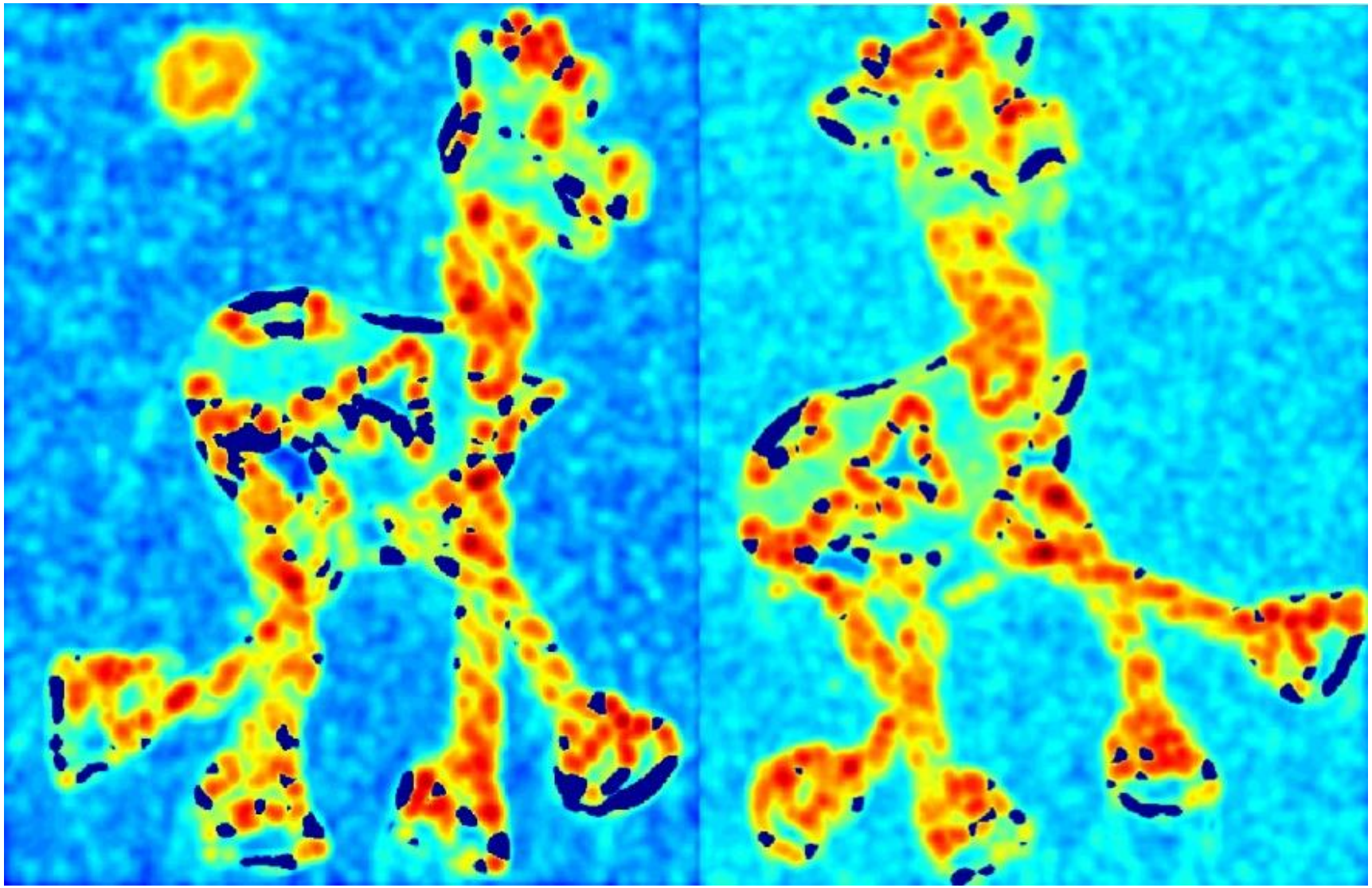
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." , 1988 *Proceedings of the 4th Alvey Vision Conference*: pages 147-151.



Harris Detector: Workflow

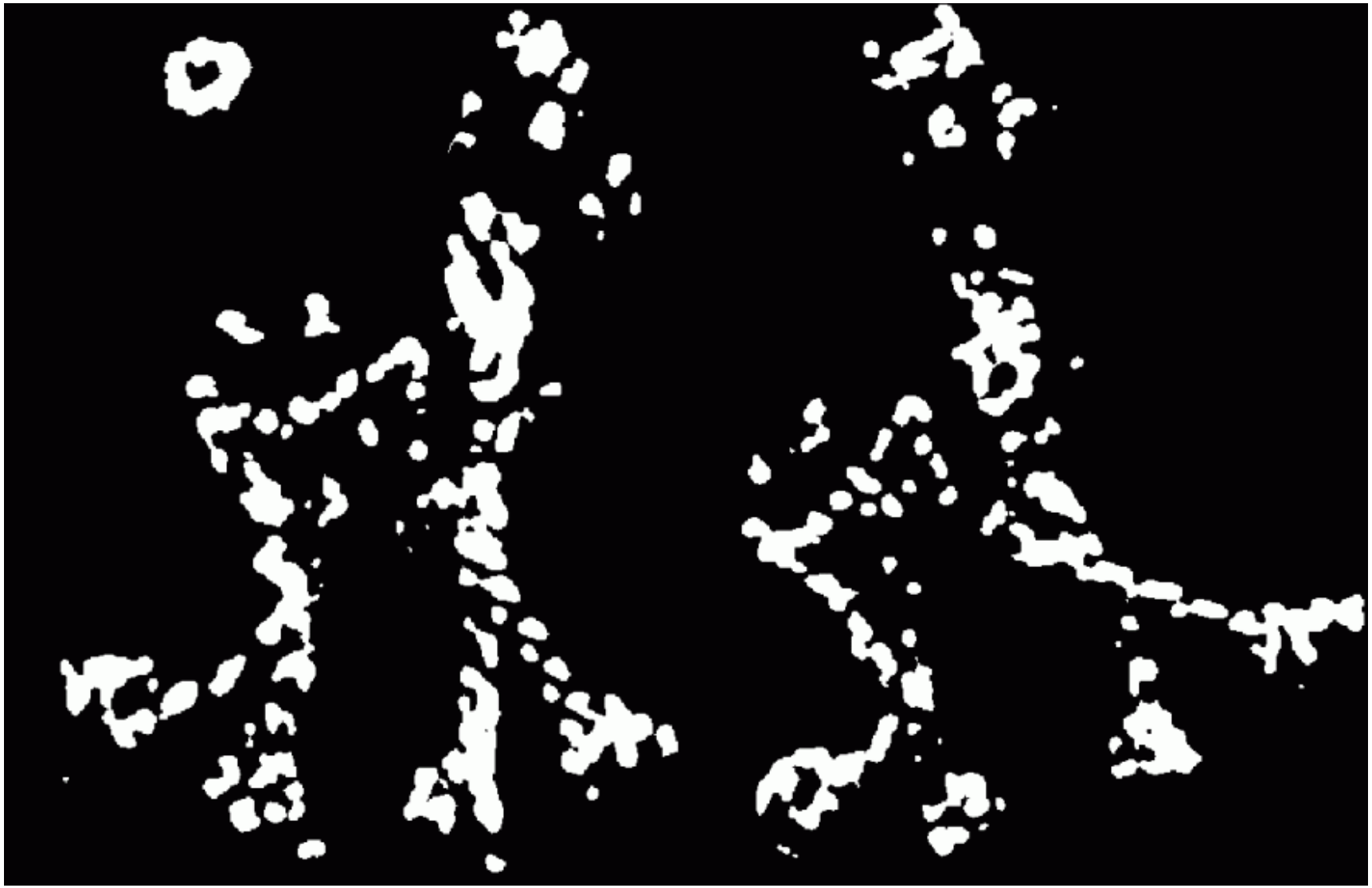
Harris Detector: Workflow

- Compute corner response R



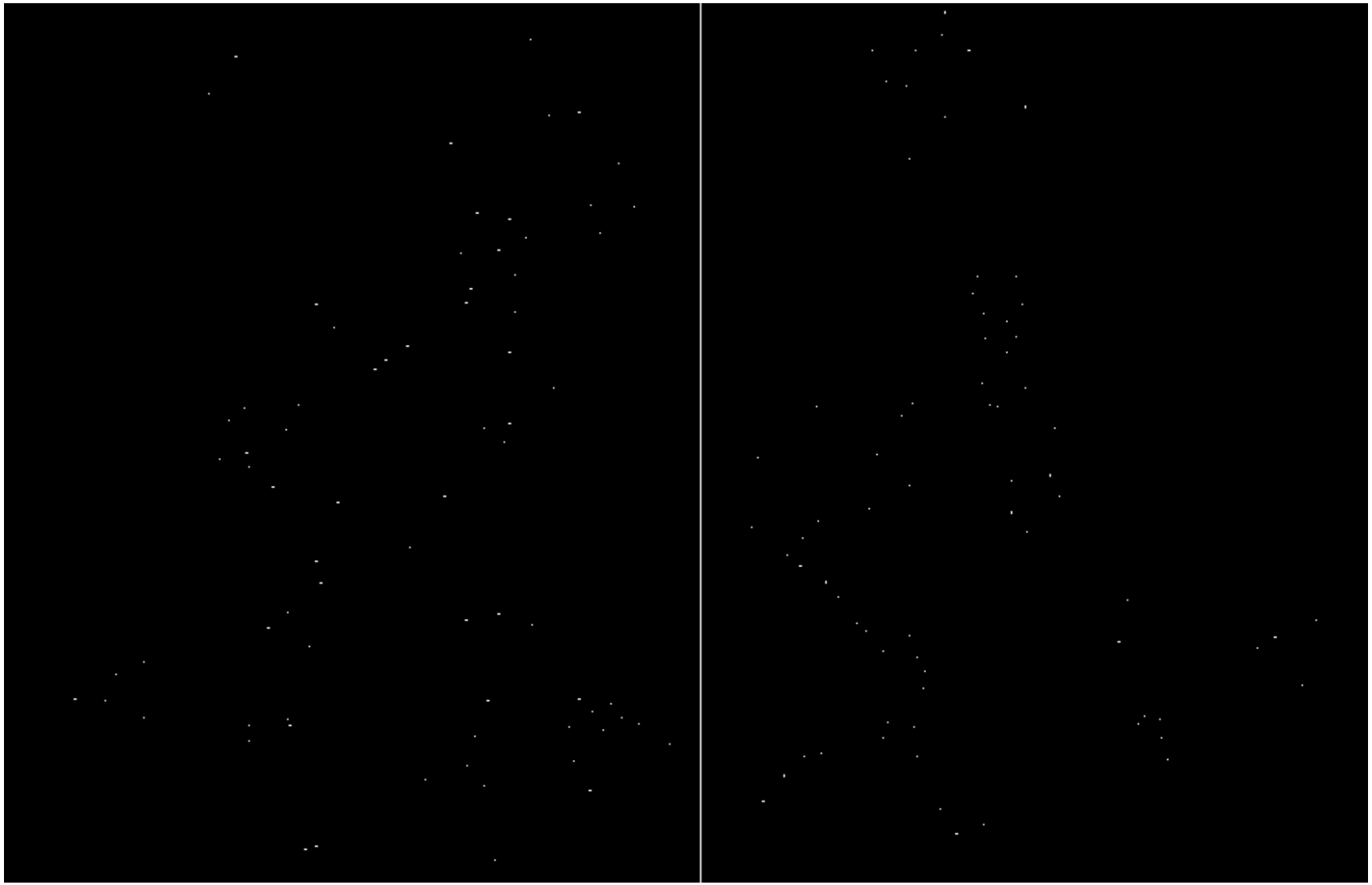
Harris Detector: Workflow

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



Harris Detector: Workflow

- Take only the points of local maxima of thresholded R (non-maxima suppression)



Harris Detector: Workflow

Harris (or Shi-Tomasi) Corner Detector Algorithm

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*

From now on, whenever you hear Harris corner detector we will be referring to either the original Harris detector (1988) or to its modification by Shi-Tomasi (1994). The Shi-Tomasi, despite being a bit more expensive, yet has a small advantage… see next slides

Harris vs. Shi-Tomasi

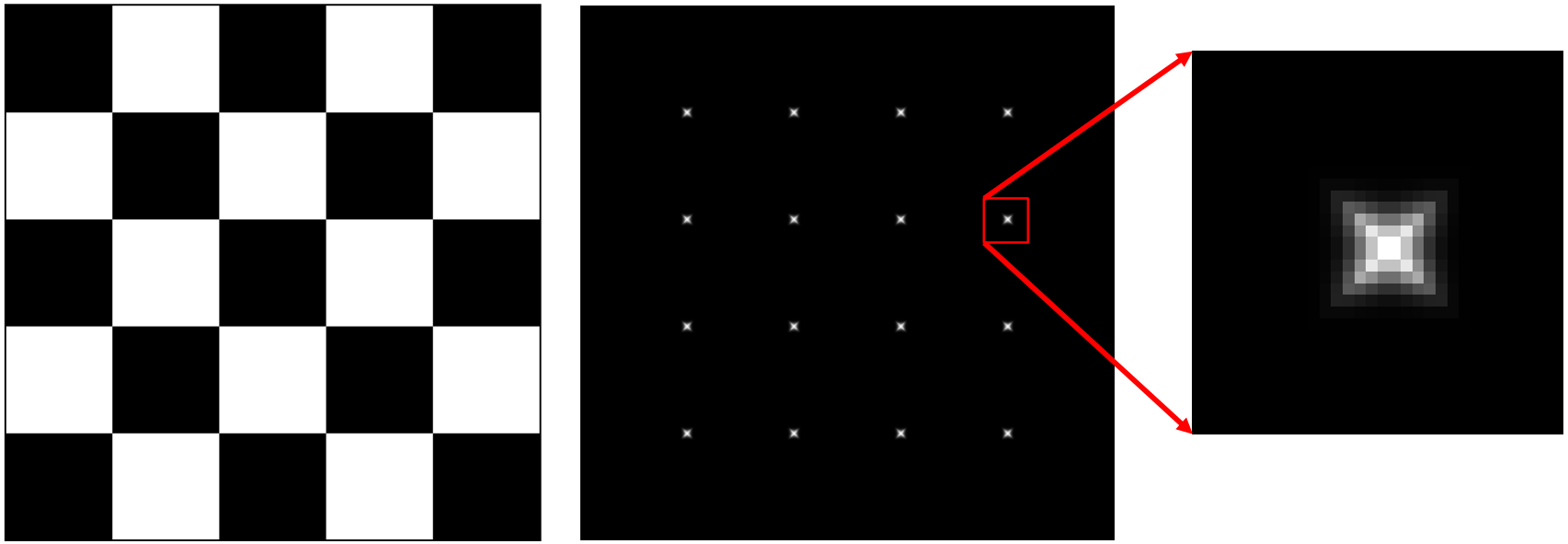
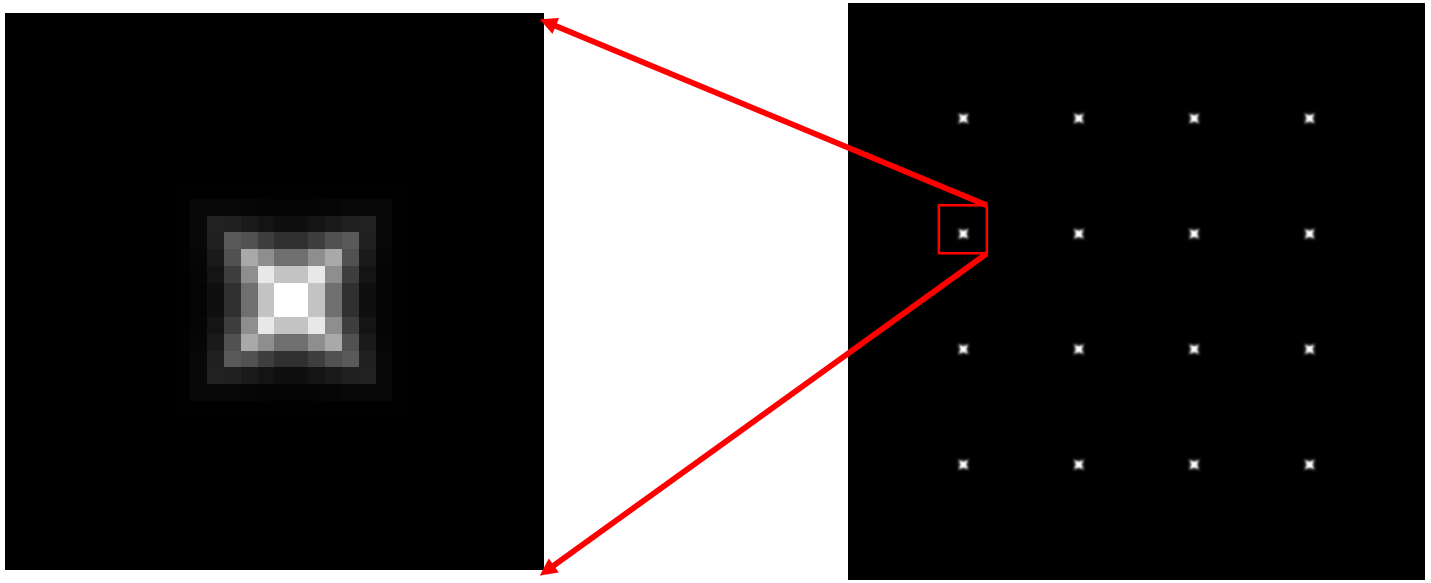


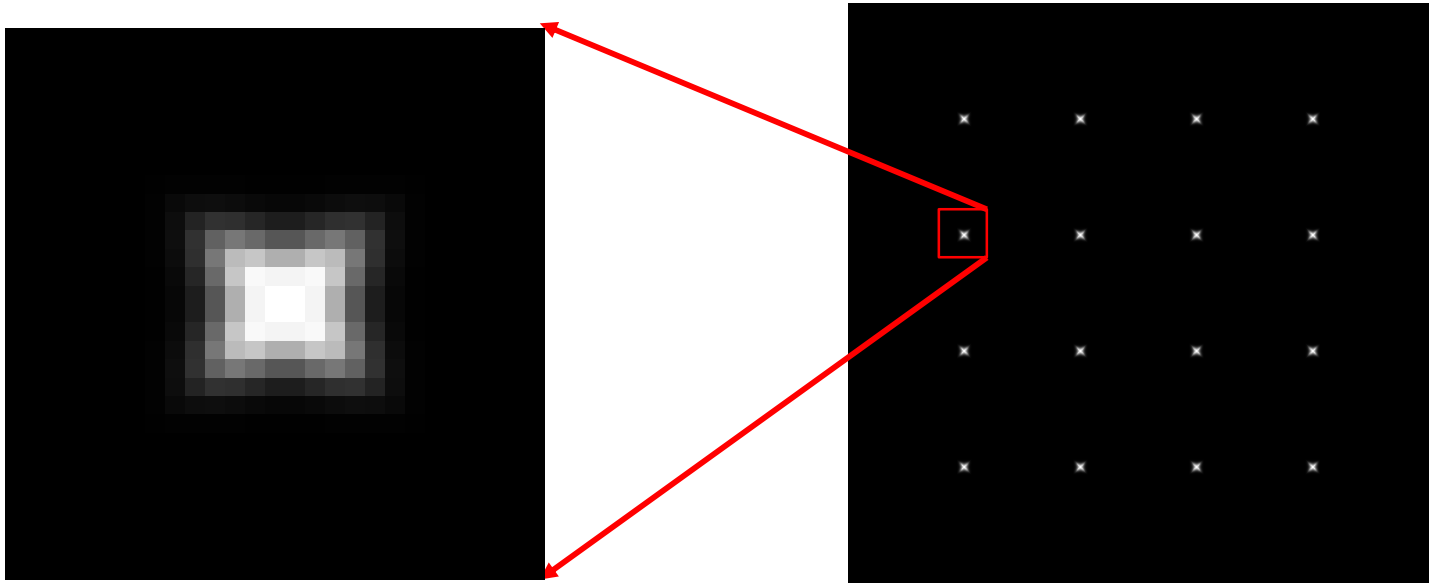
Image I

Cornerness response R

Harris vs. Shi-Tomasi



Shi-Tomasi operator



Harris operator

Harris Detector: Some Properties

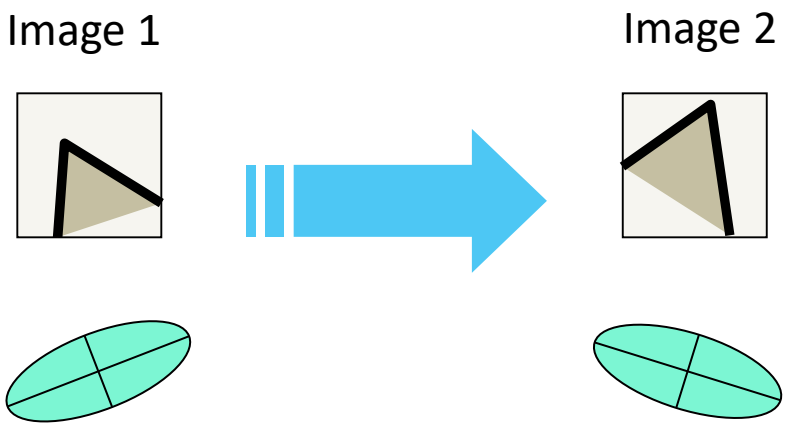
How does the size of the Harris detector affect the performance?

Repeatability:

- How does the Harris detector behave with **geometric changes**? Which means, can it re-detect the same corners when the image exhibits changes in
  - Rotation,
  - Scale (zoom),
  - View-point,
  - Illumination

Harris Detector: Some Properties

- **Rotation invariance**

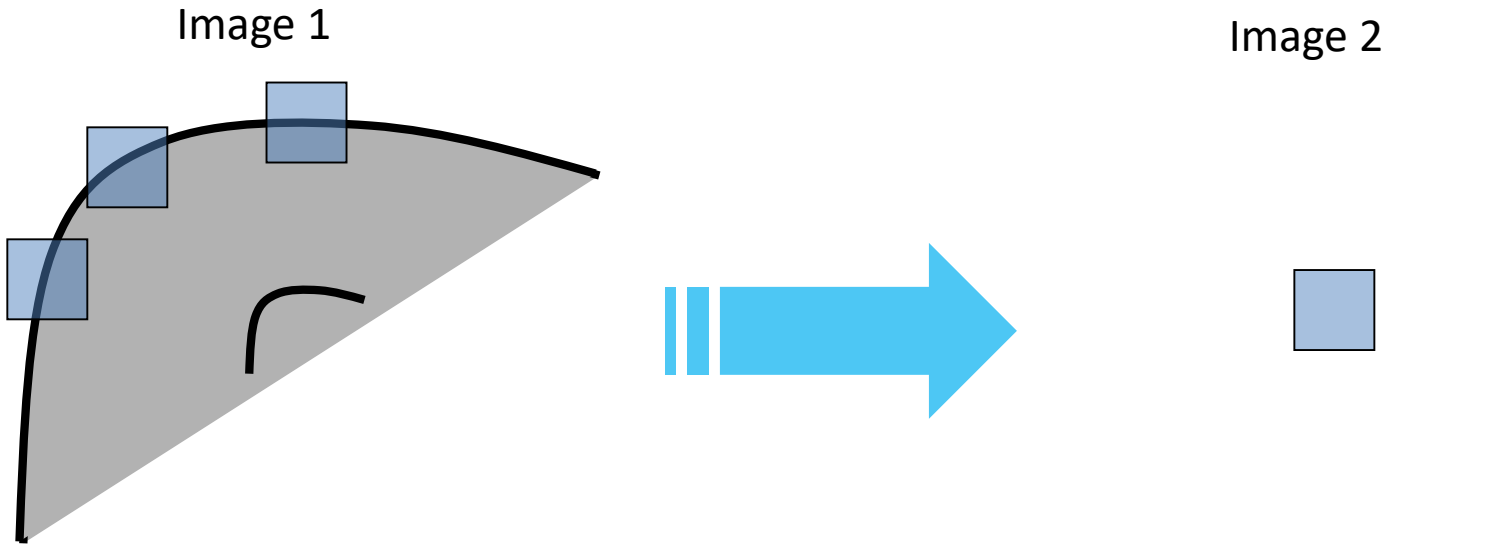


Ellipse rotates but its shape (i.e., eigenvalues) remains the same

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**

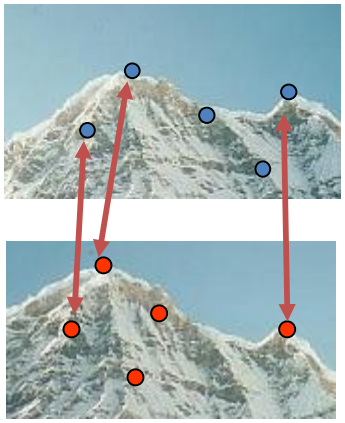
Corner!

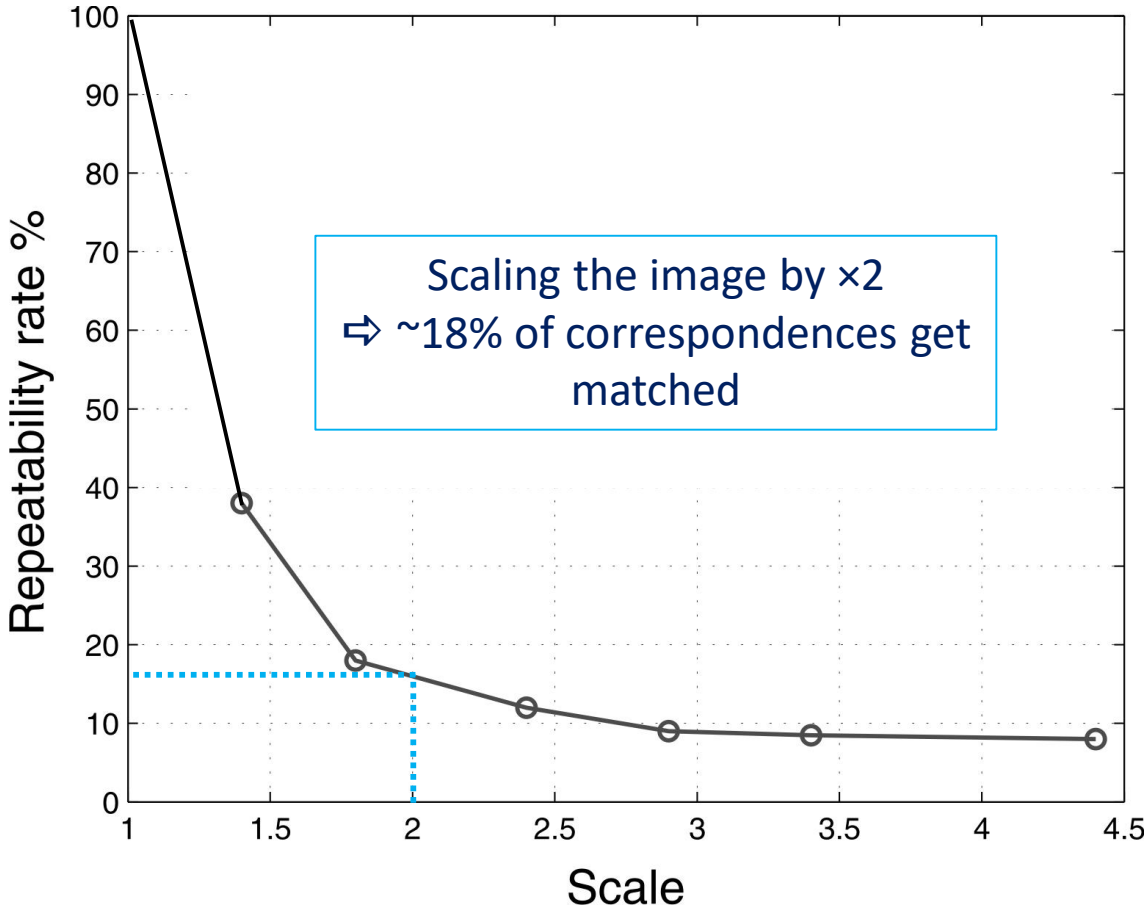
Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability=

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





Harris Detector: Some Properties

- Is it invariant to illumination and view point changes?
  - Affine illumination changes:
    - yes, why?
  - Non linear, but monotonic illumination changes:
    - yes, why?
  - View point invariance?
    - Does the same corner look like a corner from a different view point?
      - It depends on the view point change, why?

Hint:
remember that Harris corners are local maxima of the cornerness response function

Summary (things to remember)

- Filters as templates
- Correlation as a scalar product
- Similarity metrics: NCC (ZNCC), SSD (ZSSD), SAD (ZSAD), Census Transform
- Point feature detection
  - Properties and invariance to transformations
    - Challenges: rotation, scale, view-point, and illumination changes
  - Extraction
    - Moravec
    - Harris and Shi-Tomasi
      - Rotation invariance
- Reading:
  - Ch. 4.1 and Ch. 8.1 of Szeliski book
  - Ch. 4 of Autonomous Mobile Robots book
  - Ch. 13.3 of Peter Corke book

Understanding Check:

Are you able to:

- Explain what is template matching and how it is implemented?
- Explain what are the limitations of template matching? Can you use it to recognize cars?
- Illustrate the similarity metrics SSD, SAD, NCC, and Census transform?
- What is the intuitive explanation behind SSD and NCC?
- Explain what are good features to track? In particular, can you explain what are corners and blobs together with their pros and cons?
- Explain the Harris corner detector? In particular:
  - Use the Moravec definition of corner, edge and flat region.
  - Show how to get the second moment matrix from the definition of SSD and first order approximation (show that this is a quadratic expression) and what is the intrinsic interpretation of the second moment matrix using an ellipse?
  - What is the *M* matrix like for an edge, for a flat region, for an axis-aligned 90-degree corner and for a non-axis—aligned 90-degree corner?
  - What do the eigenvalues of *M* reveal?
  - Can you compare Harris detection with Shi-Tomasi detection?
  - Can you explain whether the Harris detector is invariant to illumination or scale changes? Is it invariant to view point changes?
  - What is the repeatability of the Harris detector after rescaling by a factor of 2?