University of Zurich[UZH] | ETH zürich

Institute of Informatics – Institute of Neuroinformatics

ROBOTICS & PERCEPTION GROUP

Lecture 05
Point Feature Detection and Matching

Davide Scaramuzza

Lab Exercise 3 - Today afternoon

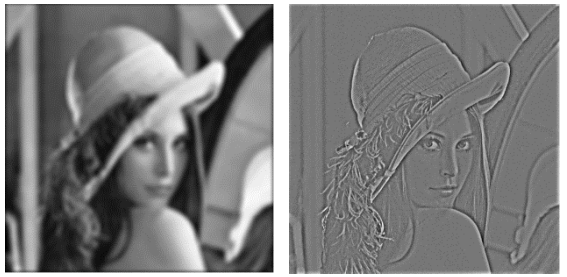
- Room ETH HG E 1.1 from 13:15 to 15:00
- Work description: implement a corner detector and tracker

Outline

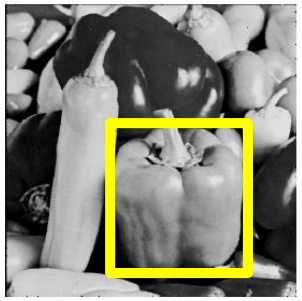
- Filters for Feature detection
- Point-feature extraction: today and next lecture

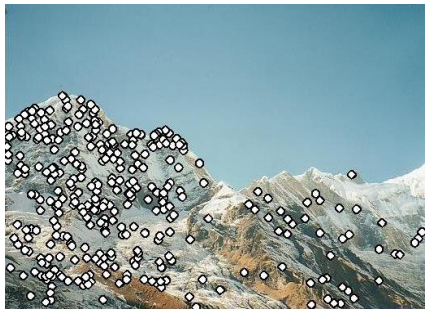
Filters for Feature Detection

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



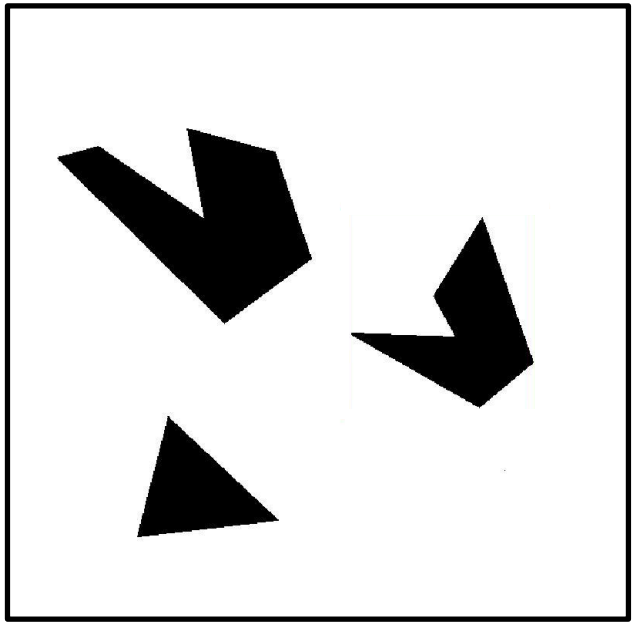
- However, filters can also be used to detect "**features**"
  - Goal: reduce amount of data to process in later stages, discard redundancy to preserve only what is useful (leads to **lower bandwidth and memory storage**)
    - **Edge detection** (we have seen this already; edges can enable line or shape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

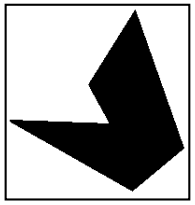




Filters for Template Matching

- 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

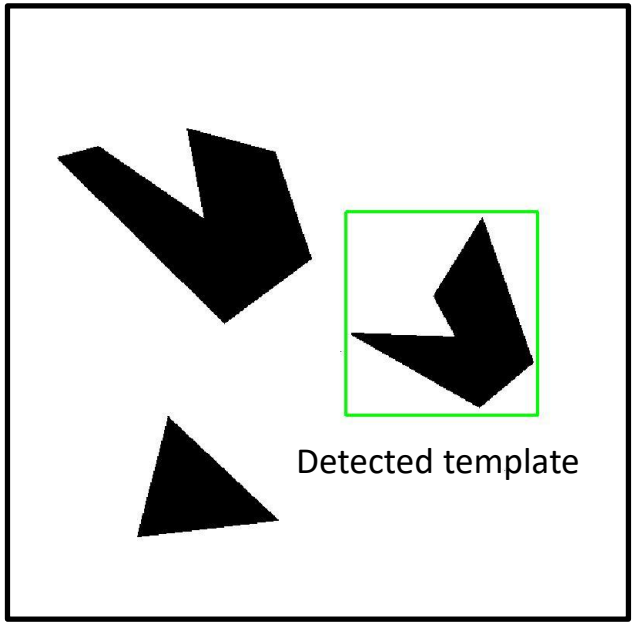


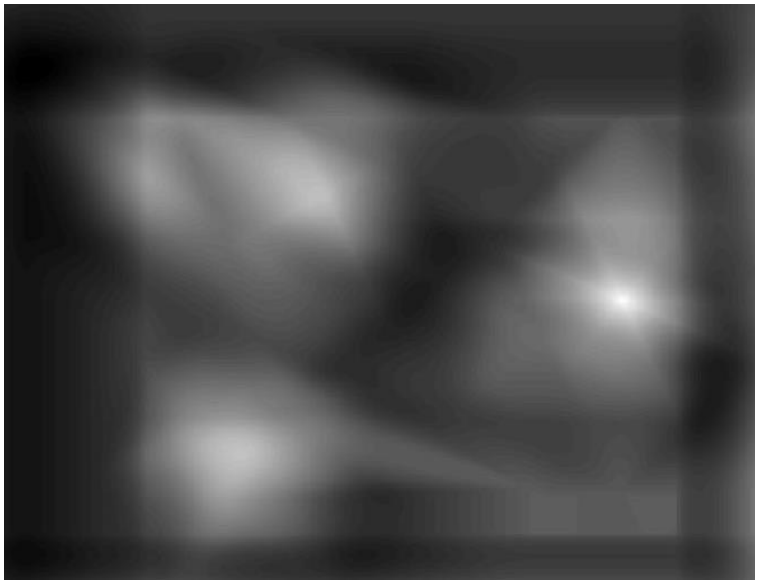


Template

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

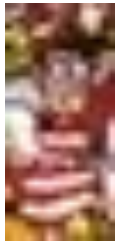




Correlation map

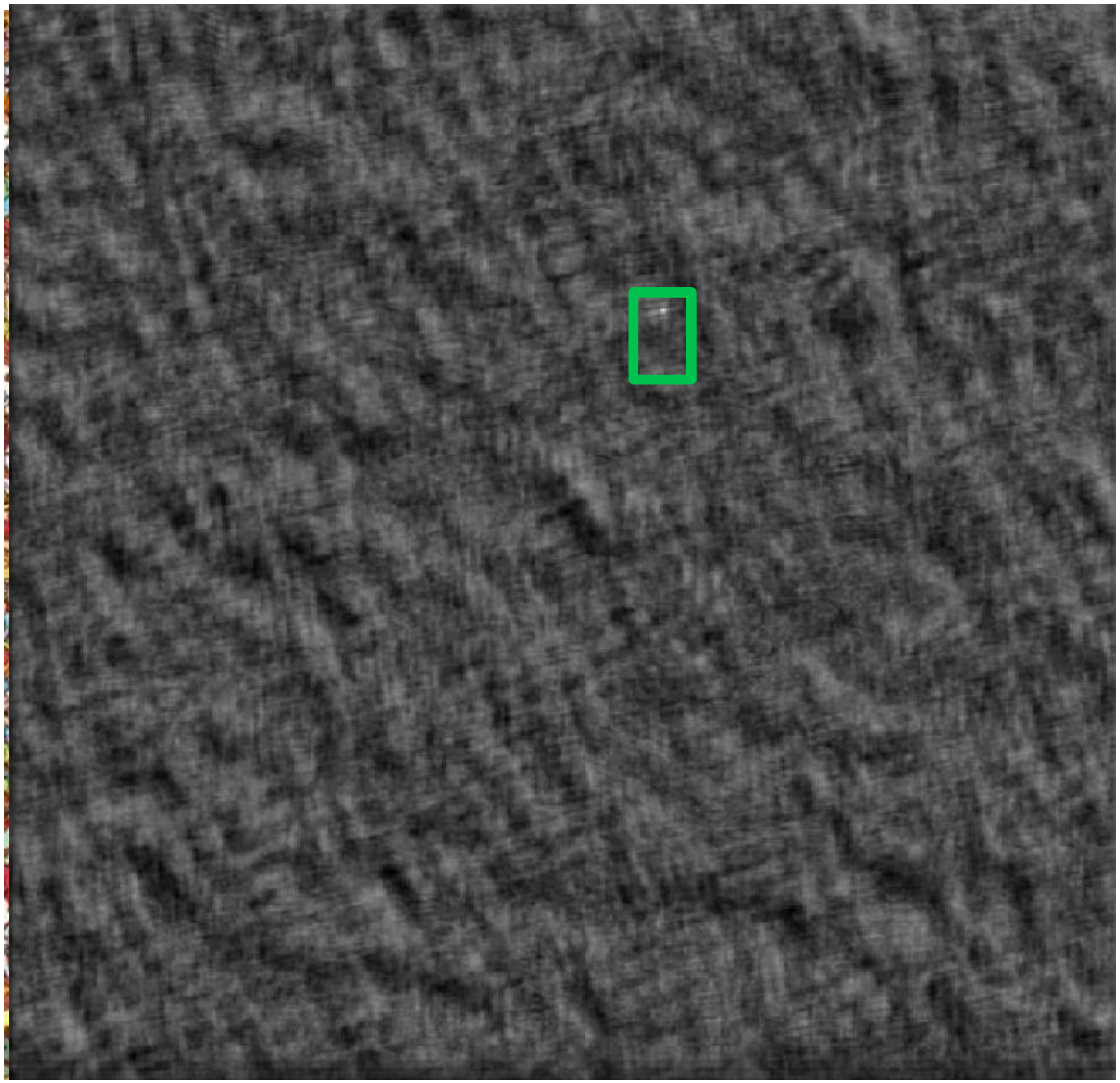
Where’s Waldo?

Scene

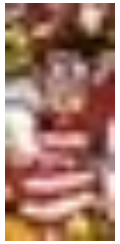


Template

Where’s Waldo?



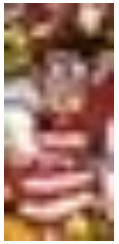
Scene



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*"

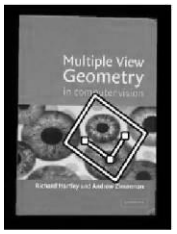
Template Matching

- 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

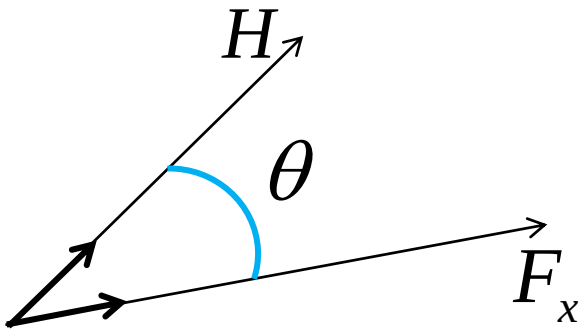


Template

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

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

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

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

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

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

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

Zero-mean SAD, SSD, NCC

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

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

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

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

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

Are these invariant to affine illumination changes?

$$\mu_H = \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k}H(u,v)}{(2N+1)^2}$$

$$\mu_F = \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k}F(u,v)}{(2N+1)^2}$$

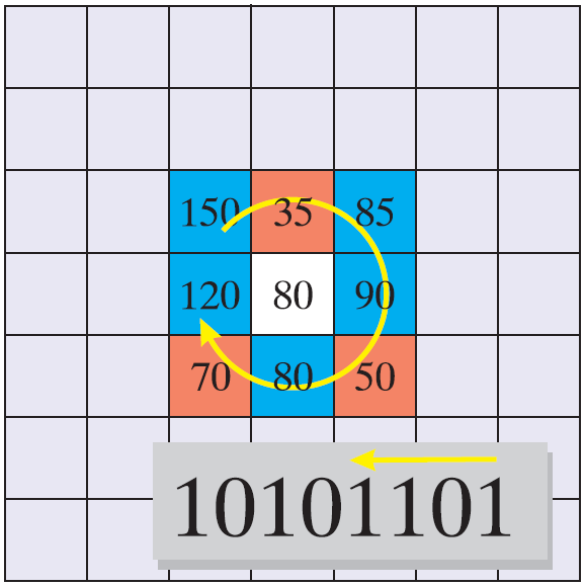
- **Zero-mean Normalized Cross Correlation (ZNCC)**

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

ZNCC is invariant to affine intensity changes!

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

- More **robust to *object-background*** *problem*
- **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

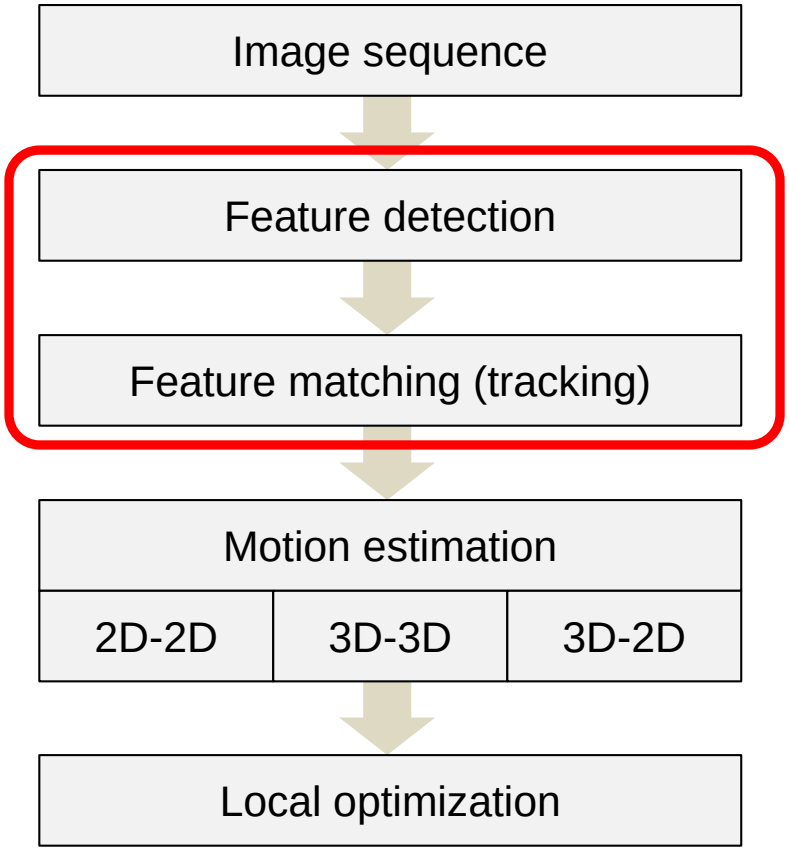
- Filters for feature extraction
- Point-feature (or keypoint) extraction: today and next lecture

Point-feature extraction and matching - Example

Video from “Forster, Pizzoli, Scaramuzza, *SVO: Semi-Direct Visual Odometry*, ICRA’14”

What do we need point features for?

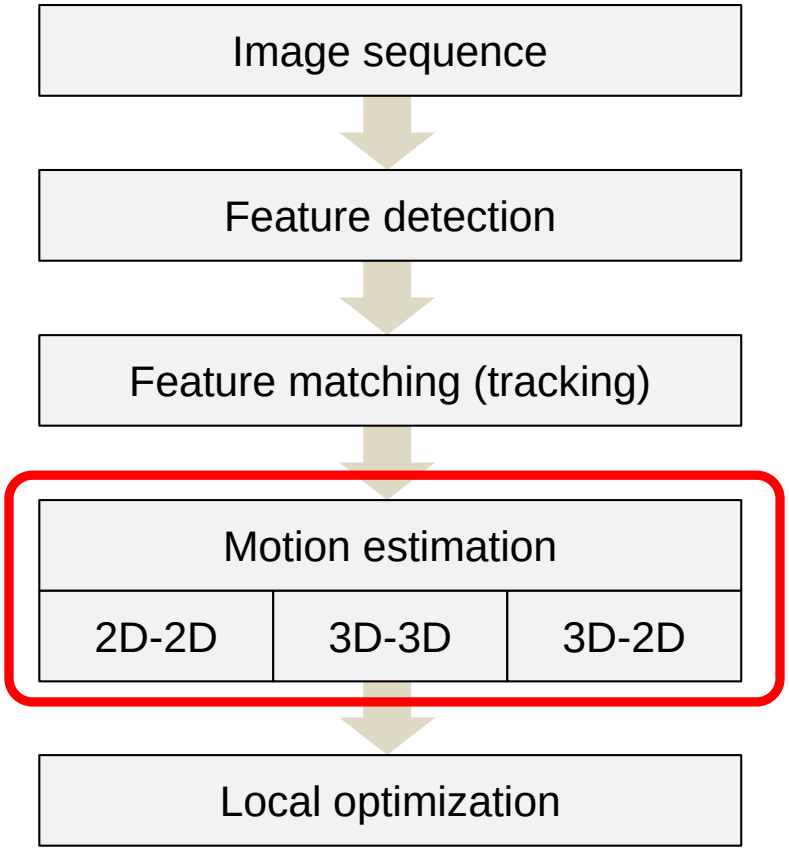
Recall the Visual-Odometry flow chart:

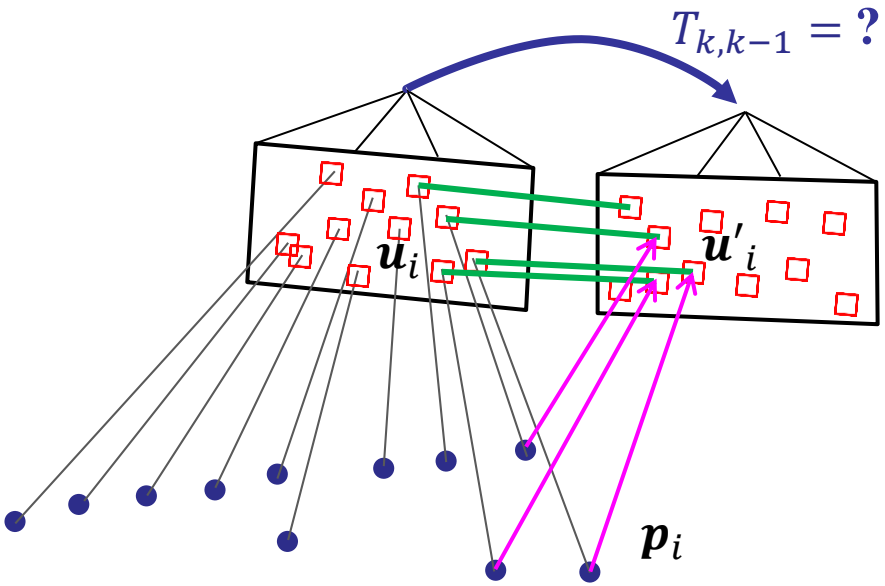


Example of features tracks

What do we need point features for?

Keypoint extraction is the key ingredient of motion estimation!



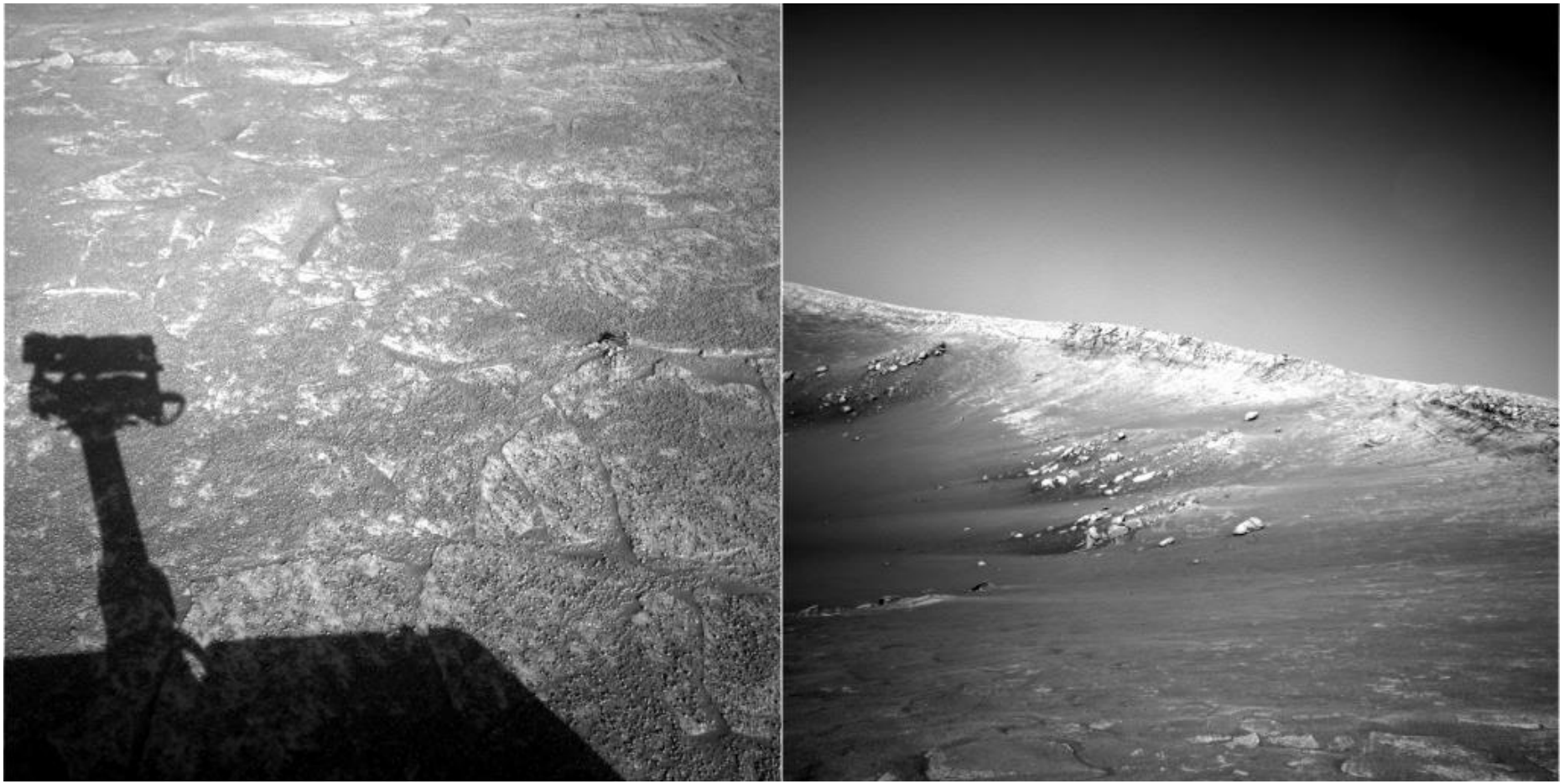


Point Features are also used for:

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

Image matching: why is it challenging?

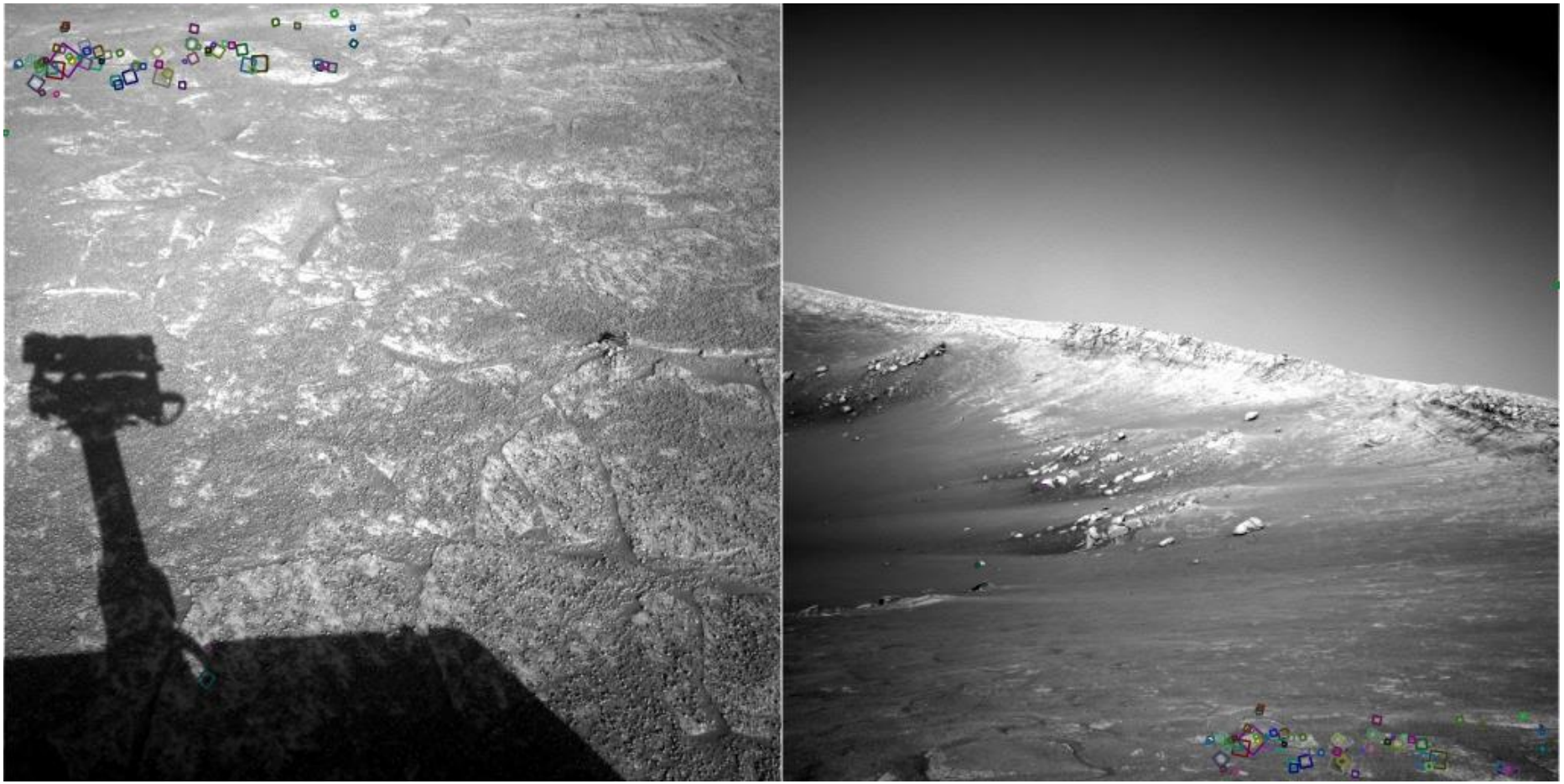
Image matching: why is it challenging?



NASA Mars Rover images

Image matching: why is it challenging?

Answer below



NASA Mars Rover images with SIFT feature matches

Example: panorama stitching

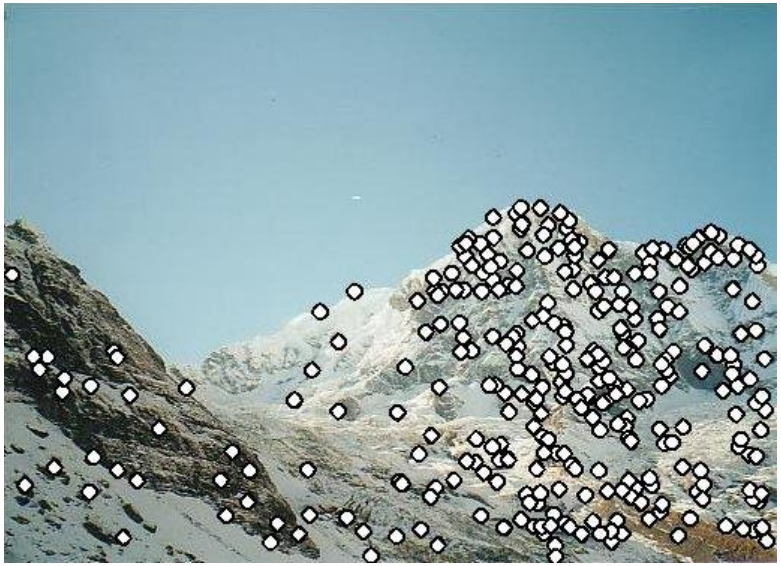
This panorama was generated using AUTOSTITCH:
http://matthewalunbrown.com/autostitch/autostitch.html

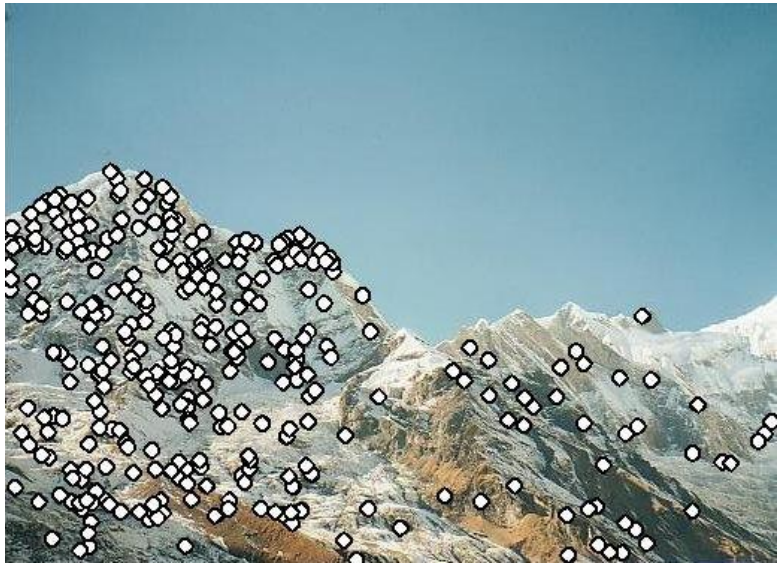
Local features and alignment

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

Local features and alignment

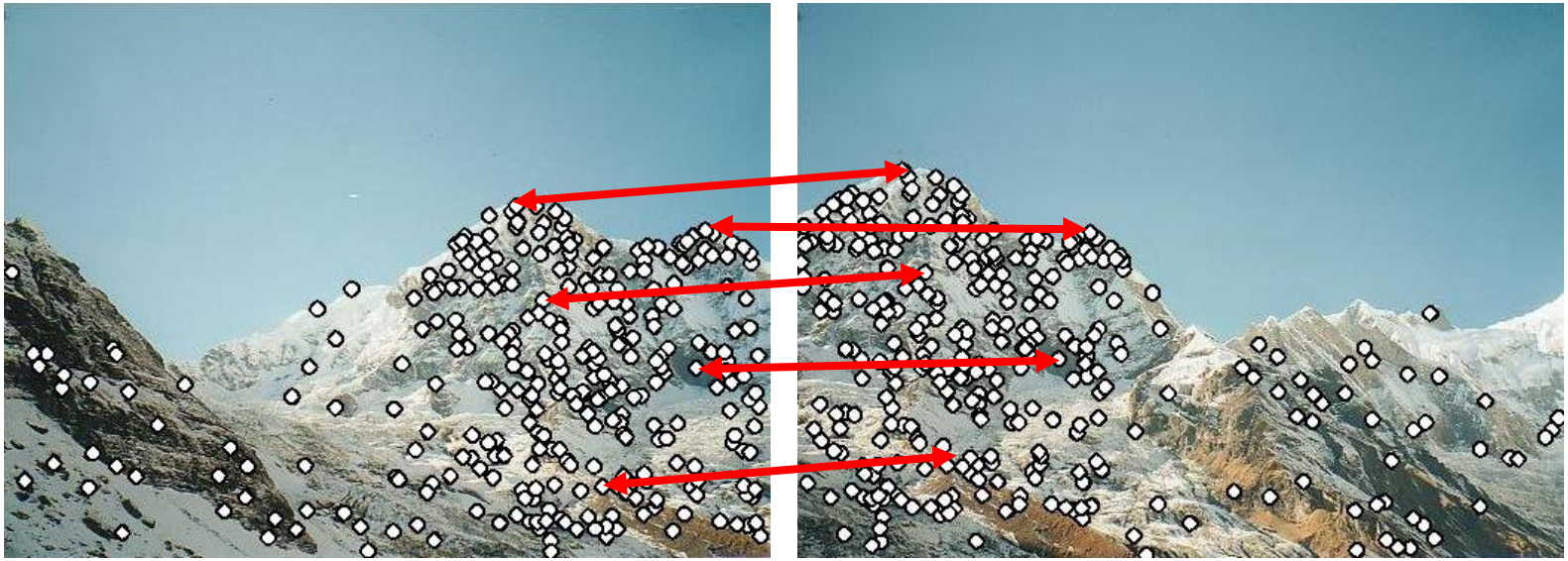
- Detect point features in both images





Local features and alignment

- Detect point features in both images
- Find corresponding pairs

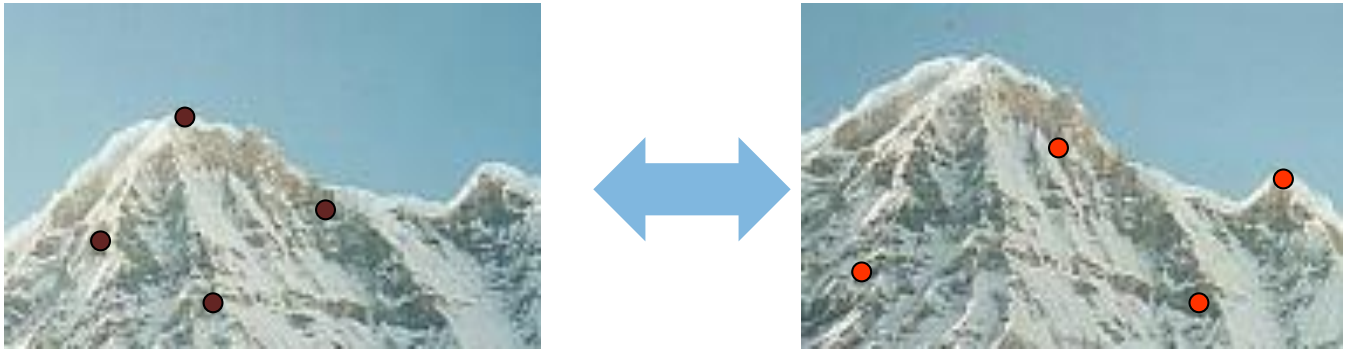


Local features and alignment

- 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

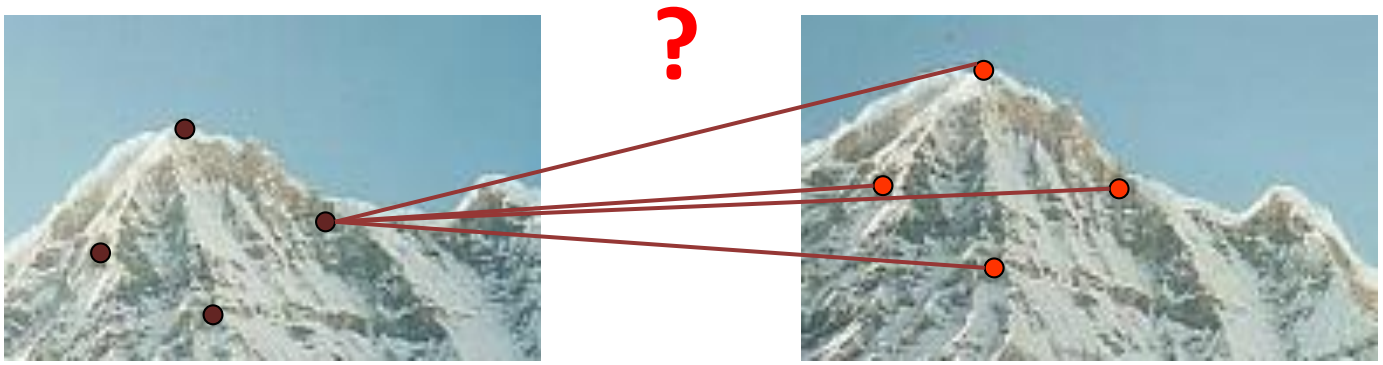


no chance to match!

We need a **repeatable** feature detector

Matching with Features

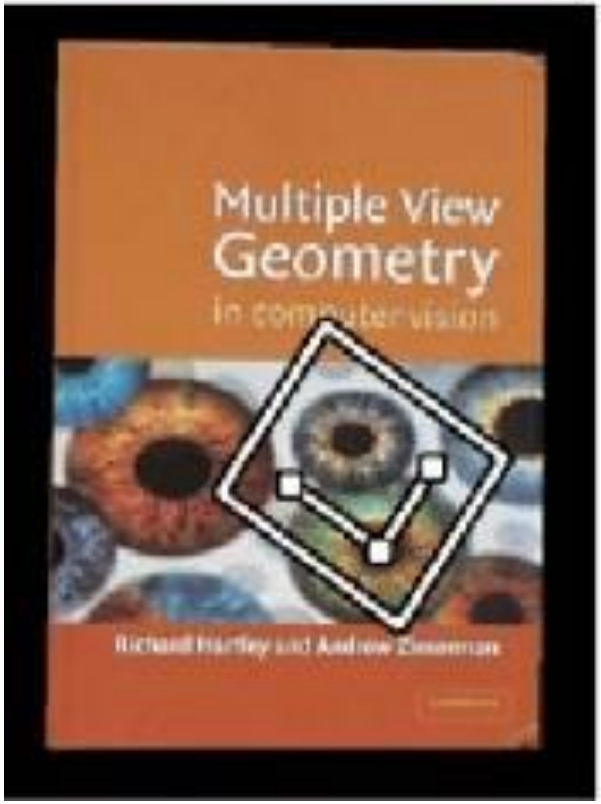
- Problem 2:
  - For each point, identify its correct correspondence in the other image(s)



We need a **reliable** and **distinctive** feature descriptor that is 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)*:

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

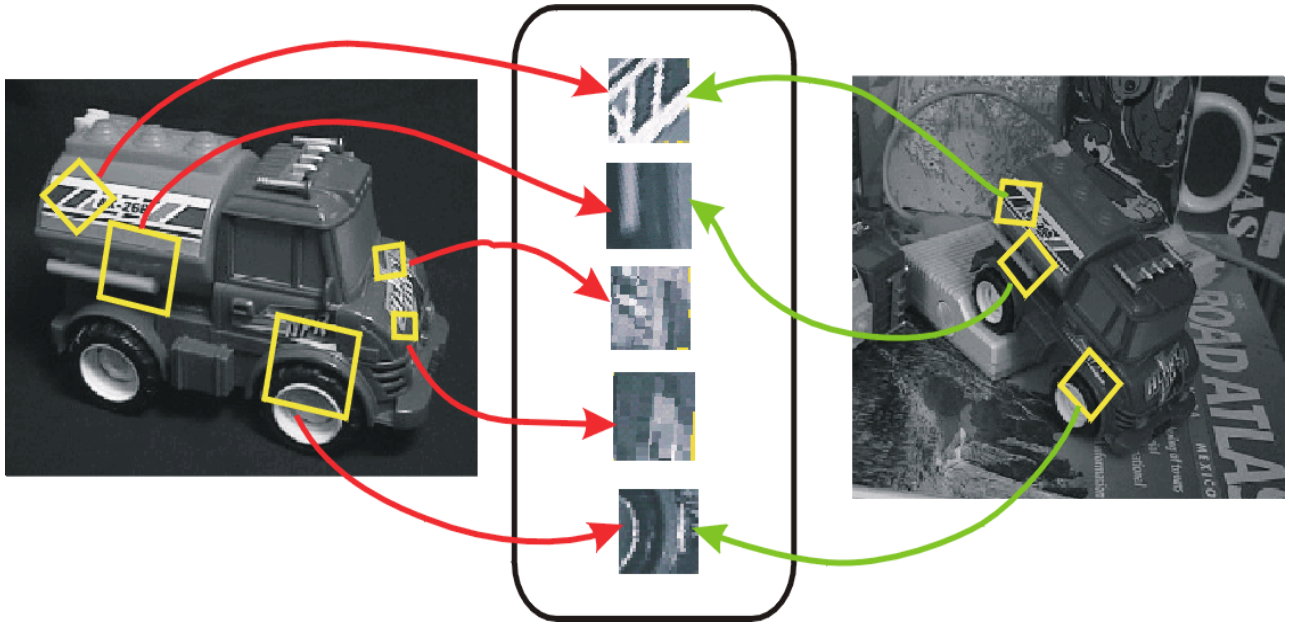
Invariant local features

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

Basic steps:

1) Detect distinctive interest points

2) Extract invariant descriptors

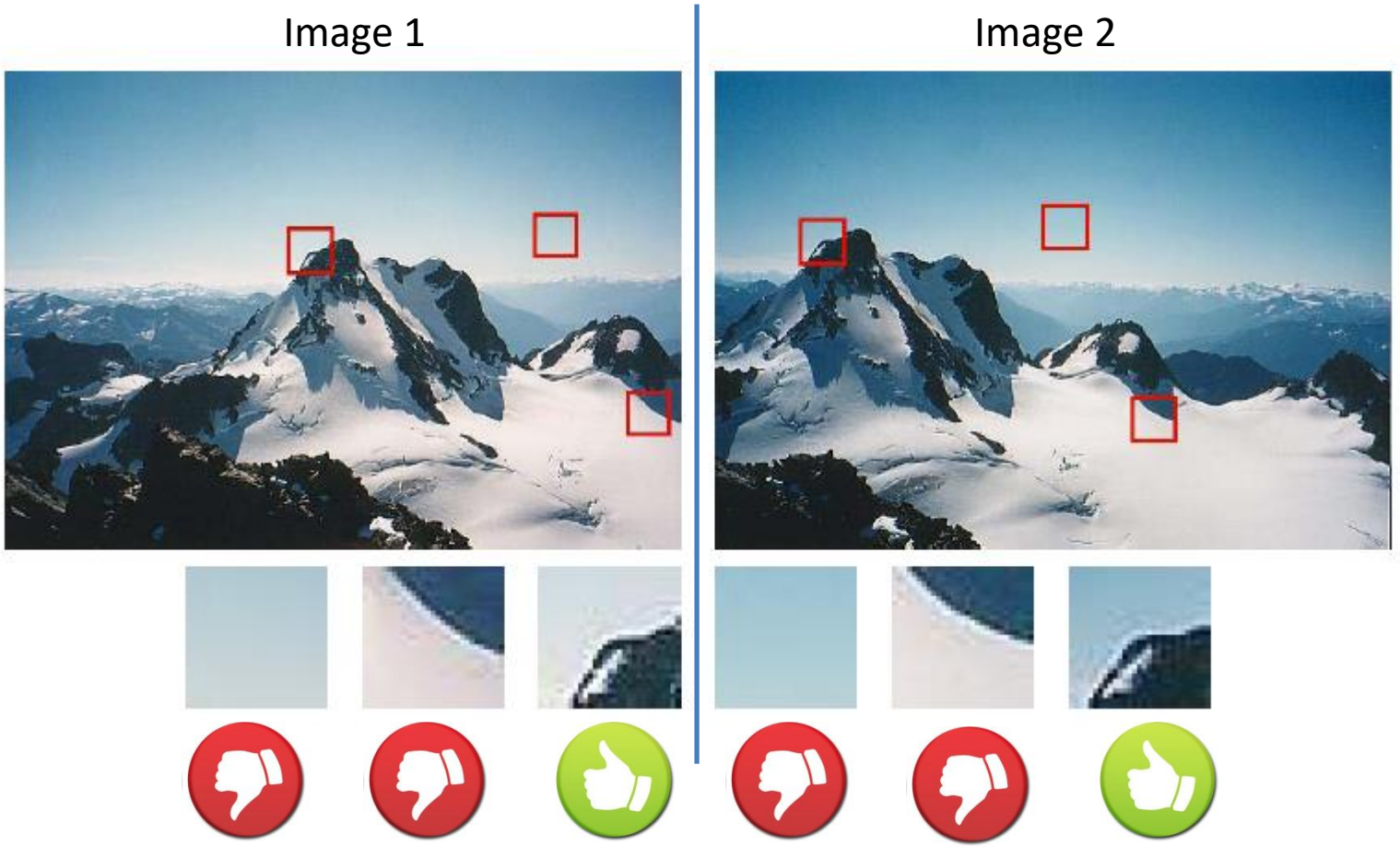


Main questions

- What points are distinctive (i.e., *features*, *keypoints, salient* points), such that they are *repeatable? (i.e.,* 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?

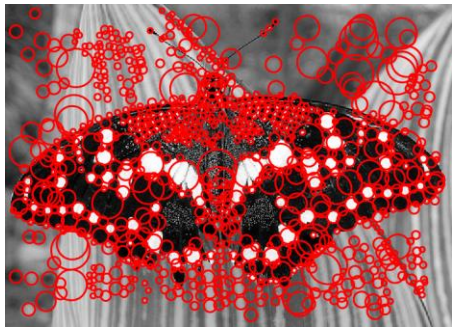
- Consider the image pair below with extracted patches
- Notice how some patches can be localized or matched with higher accuracy than others



Point Features: Corners vs Blob detectors

- A **corner** is defined as the intersection of one or more edges
  - A corner has high localization accuracy
    - Corner detectors are good for VO
  - It's **less distinctive than a blob**
  - E.g., *Harris, Shi-Tomasi, SUSAN, FAST*

- A **blob** is any other image pattern, **which is not a corner**, that differs significantly from its neighbors in intensity and texture (e.g., a connected region of pixels with similar color, a circle, etc.)
  - **Has less localization accuracy than a corner**
  - **Blob detectors are better for place recognition**
  - It's **more distinctive than a corner**
  - E.g., *MSER, LOG, DOG (SIFT), SURF, CenSurE*



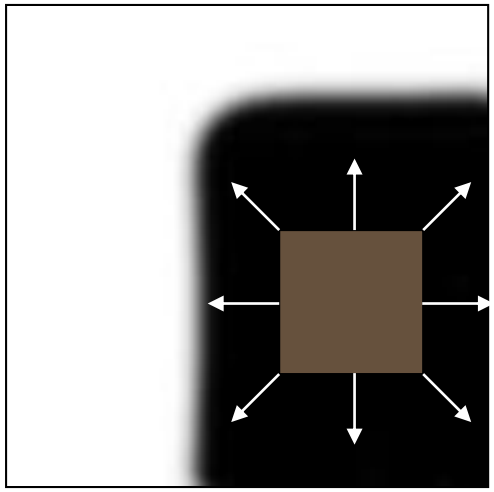
Corner detection

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

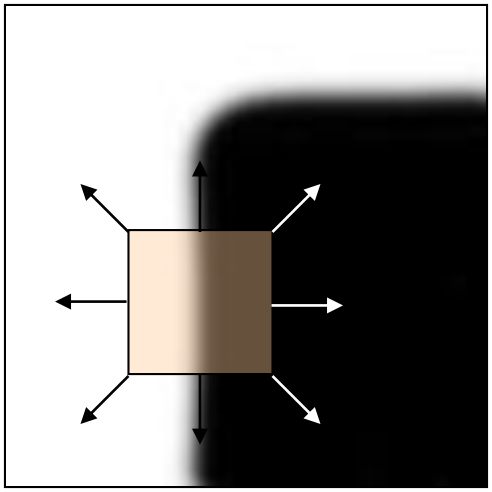
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." , 1988 *Proceedings of the 4th Alvey Vision Conference*: pages 147--151.

The Moravec Corner detector (1980)

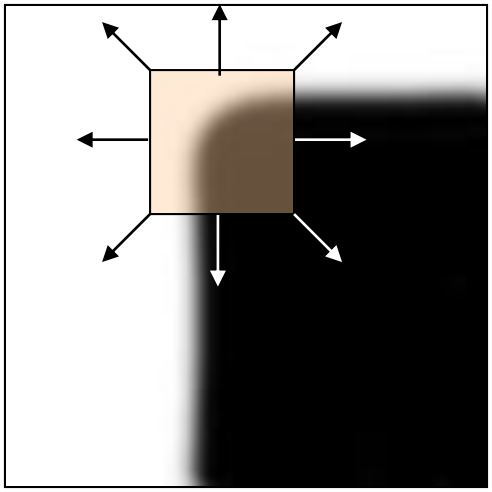
- How do we identify corners?
- We can easily recognize the point by looking through a small window
- Shifting a window in **any direction** should give a **large change** in intensity (e.g., in SSD) in at least 2 directions



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



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

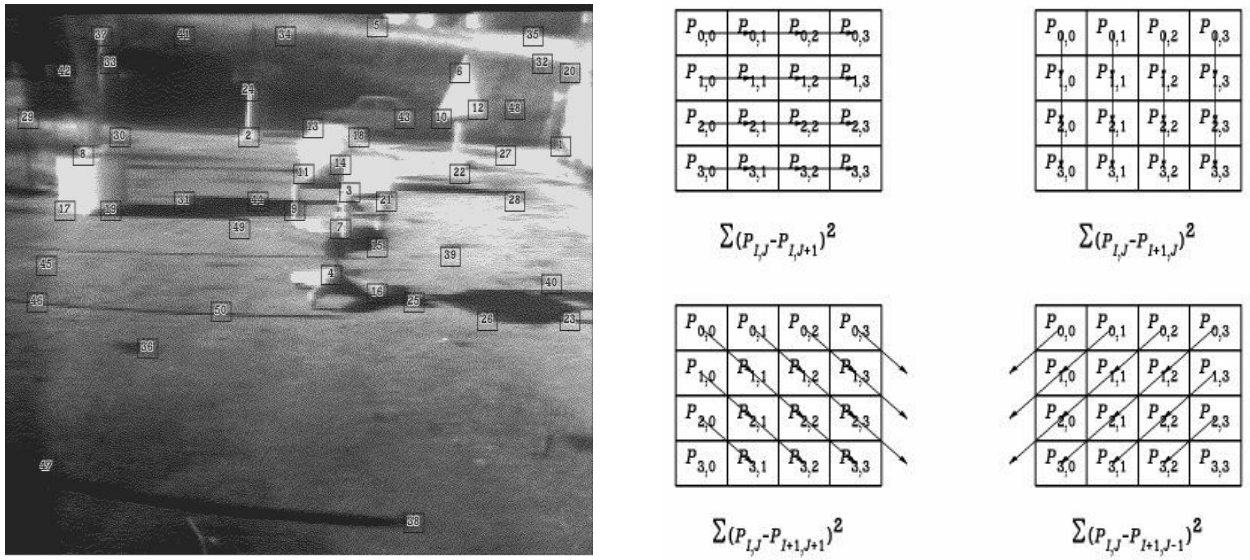


"corner":
significant change in at least 2 directions
(i.e., SSD $\gg 0$ in all directions)

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)

"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." [Moravec'80, Ch. 5]



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:

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

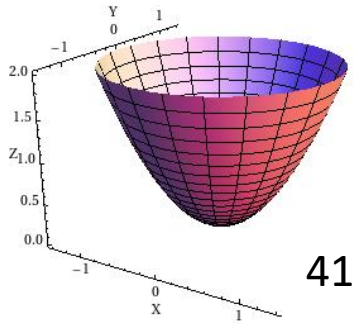
- Let $I_x = \frac{\partial I(x, y)}{\partial x}$ and $I_y = \frac{\partial I(x, y)}{\partial y}$. Approximating with a 1st order Taylor expansion:

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

- 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

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

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

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

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

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

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

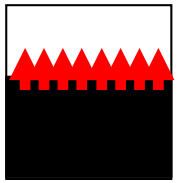
Notice that these are NOT matrix products but **pixel-wise** products!

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

2nd moment matrix — Alternative way to write M

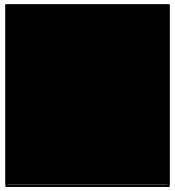
What does this matrix reveal?

- First, consider an edge or a flat region.



Edge

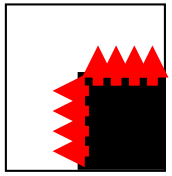
$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



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 either λ is close to 0, then this is **not** a corner.
- Now, let's consider an axis-aligned corner:



Corner

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

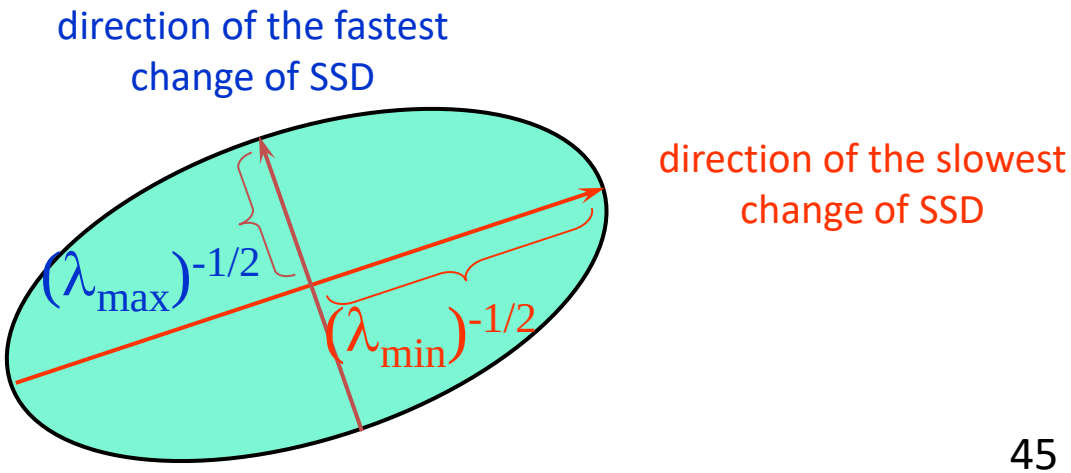
- This means dominant gradient directions are at 45 degrees with x and y axes
- What if we have a corner that is **not aligned** with the image axes?

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

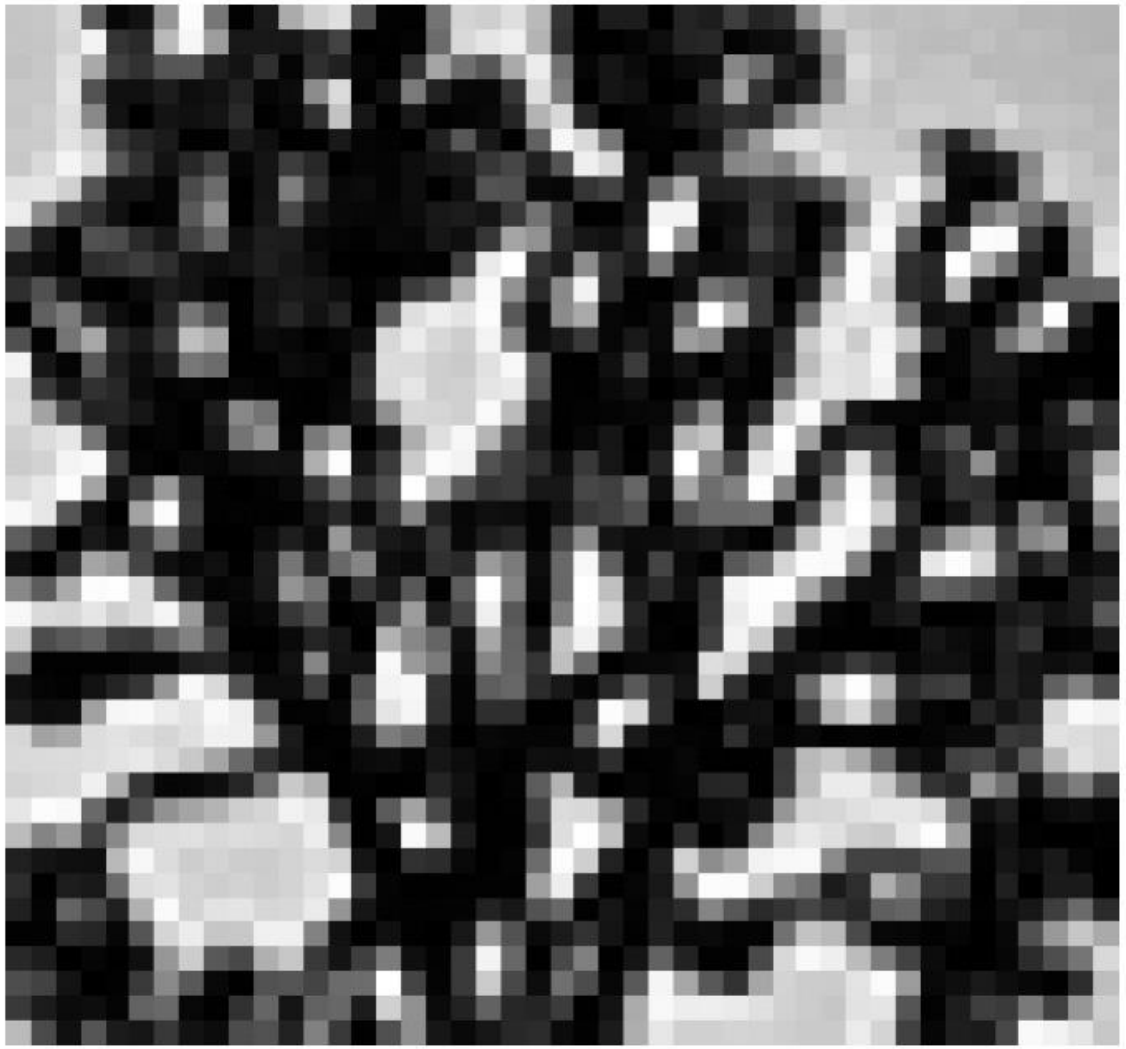
- You can easily proof that λ_1, λ_2 are the **eigenvalues** of M.
- The **eigenvectors** and **eigenvalues** of a matrix **A** are the vectors **x** and scalars λ that satisfy:

$$Ax = \lambda x$$

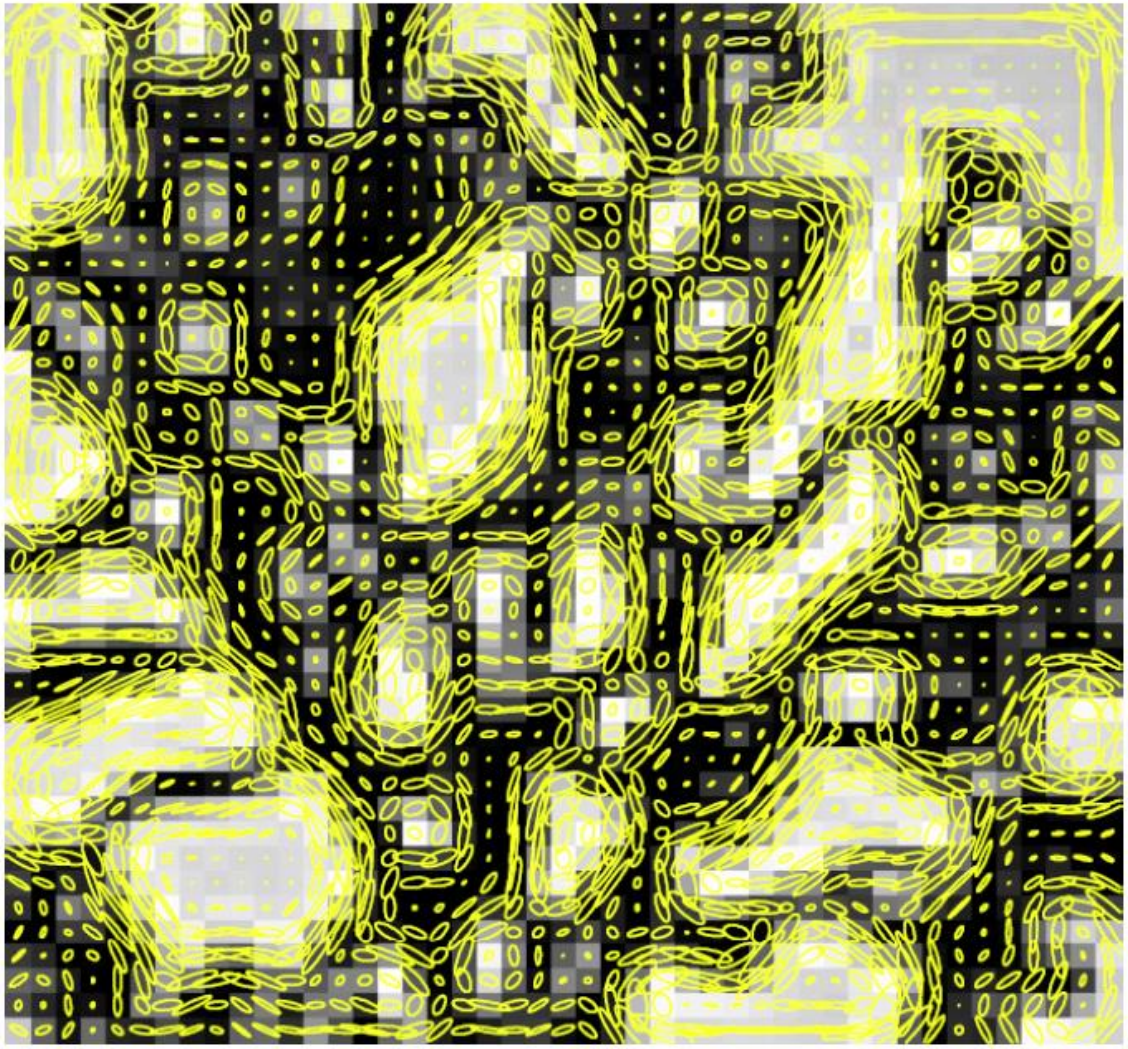
- The scalar λ is the **eigenvalue** corresponding to **x**
  - The eigenvalues are found by solving: $\det(A - \lambda I) = 0$
  - In our case, ***A*** = ***M*** is a 2x2 matrix, so we have $\det\begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix} = 0$
  - The solution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



NB: the ellipses here are plot 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.

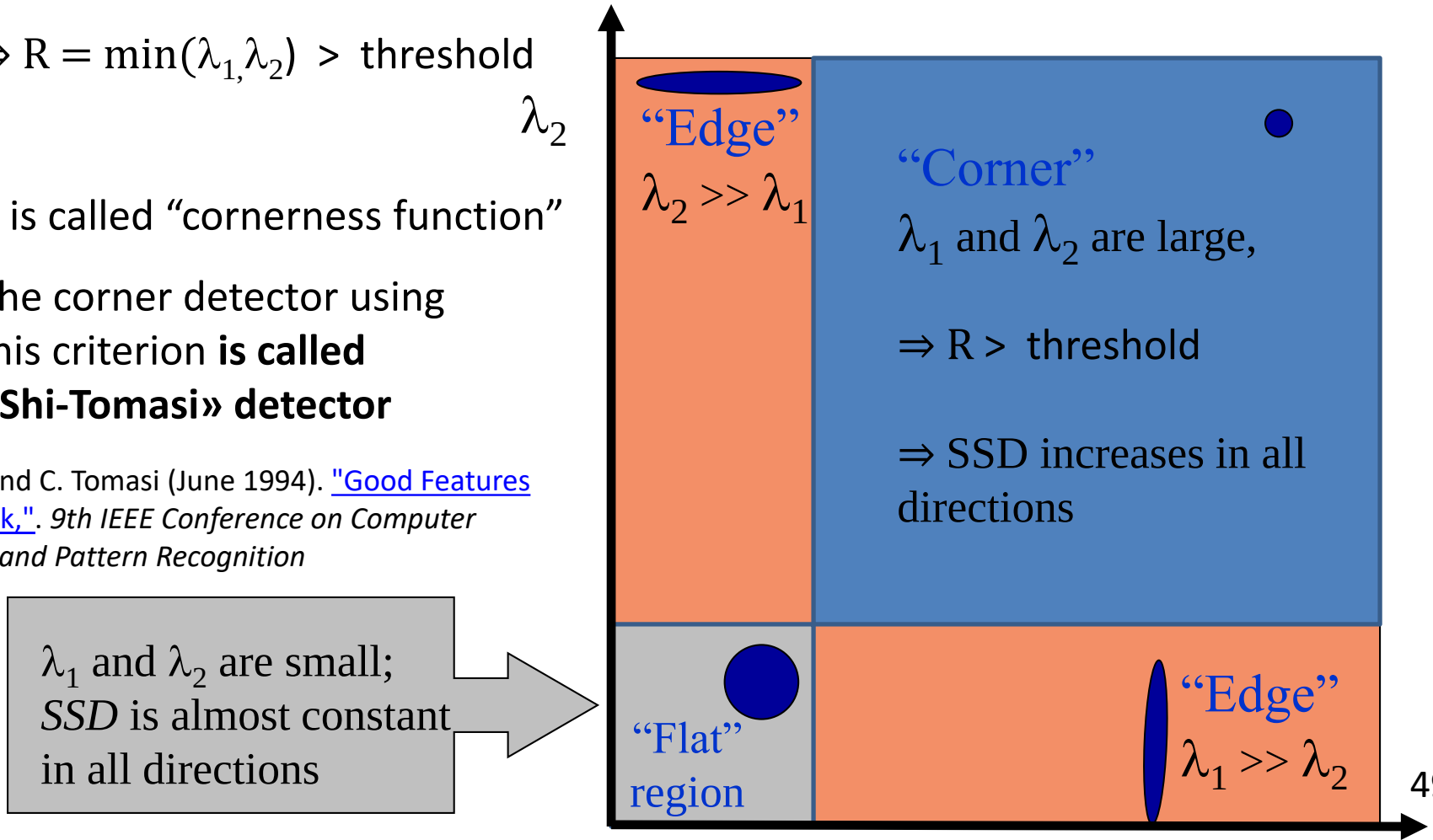
Interpreting the eigenvalues

- 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*

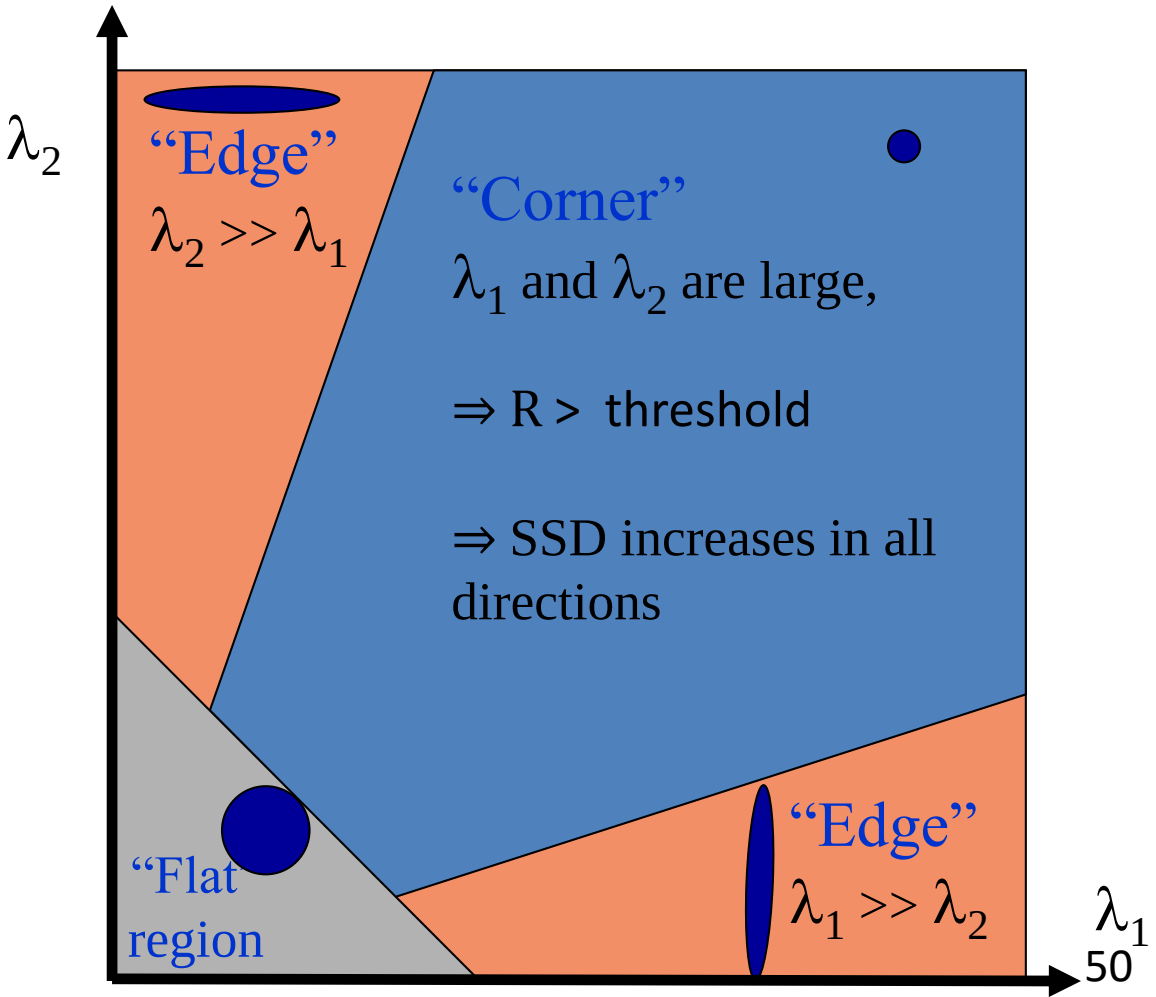


Interpreting the eigenvalues

- 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)



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

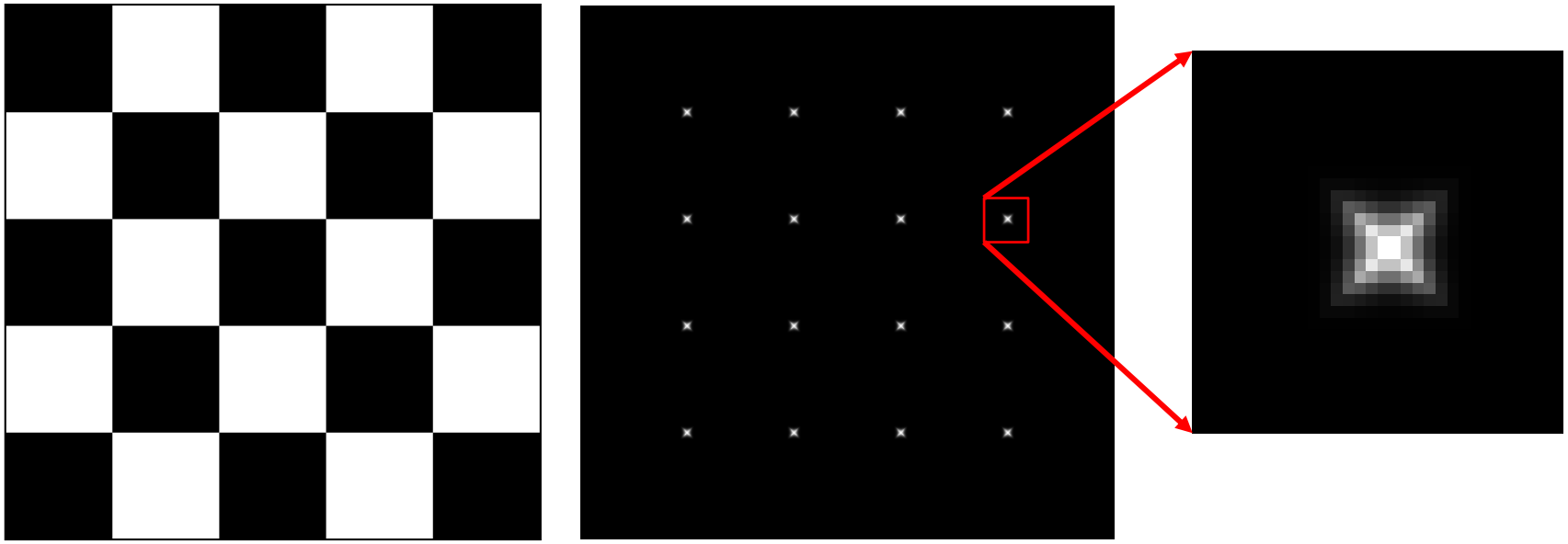
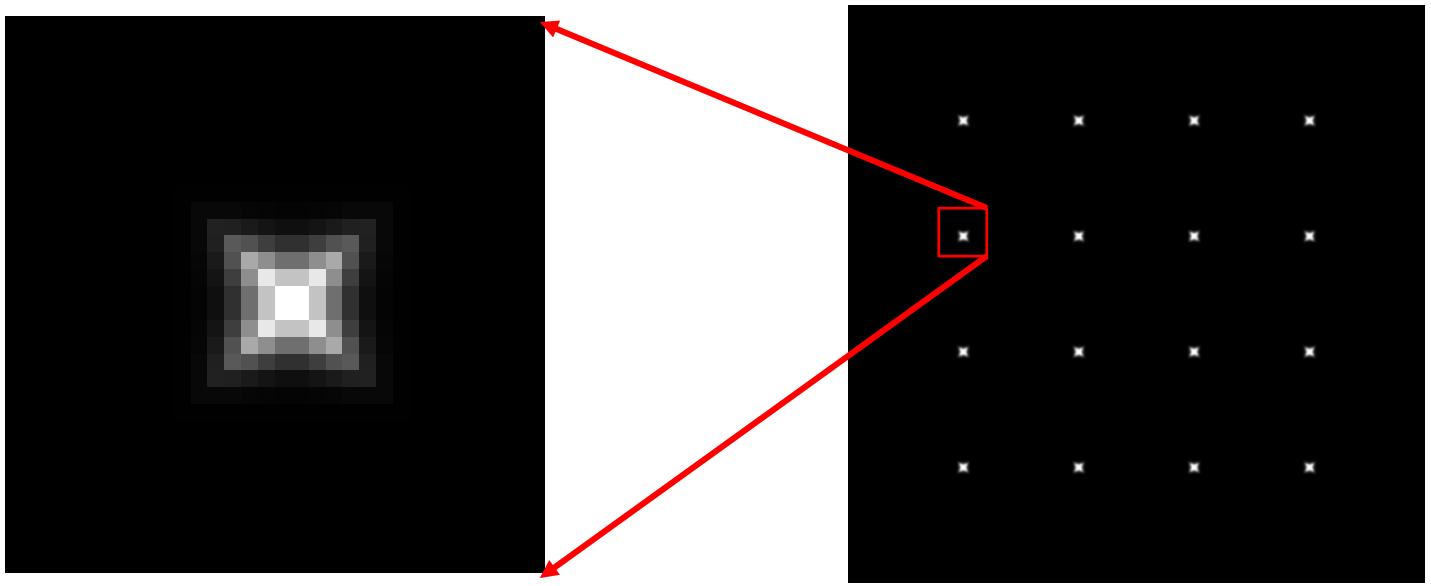


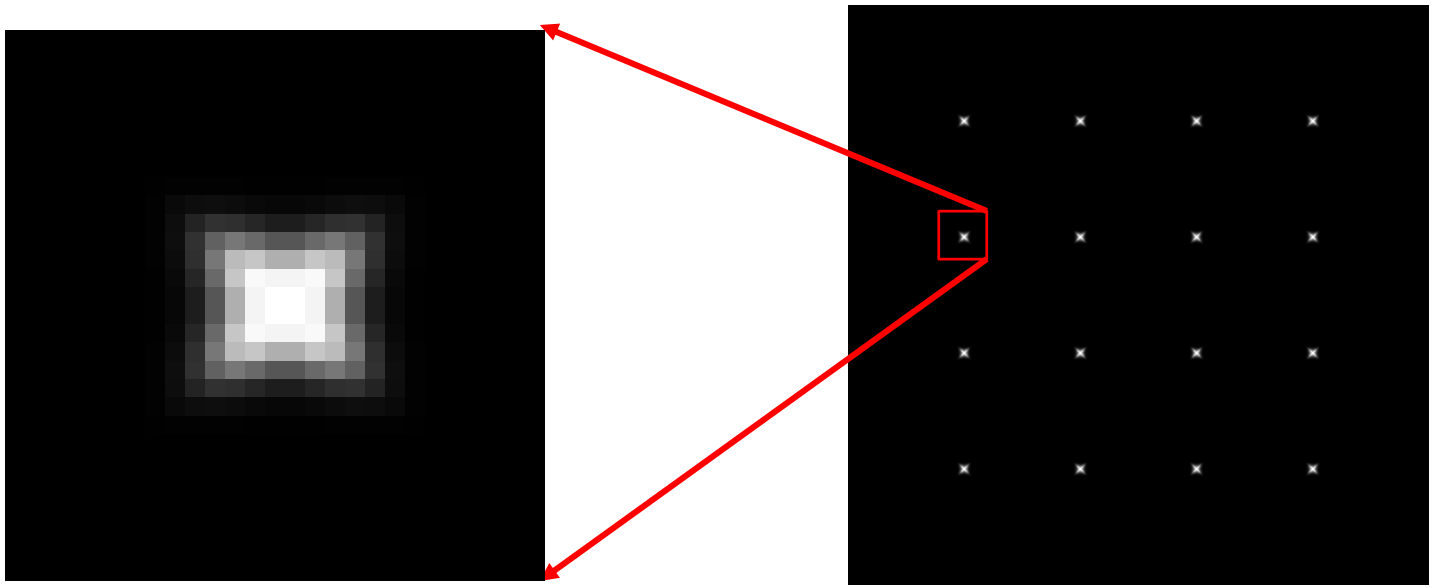
Image I

Cornerness response R

Harris vs. Shi-Tomasi



Shi-Tomasi operator

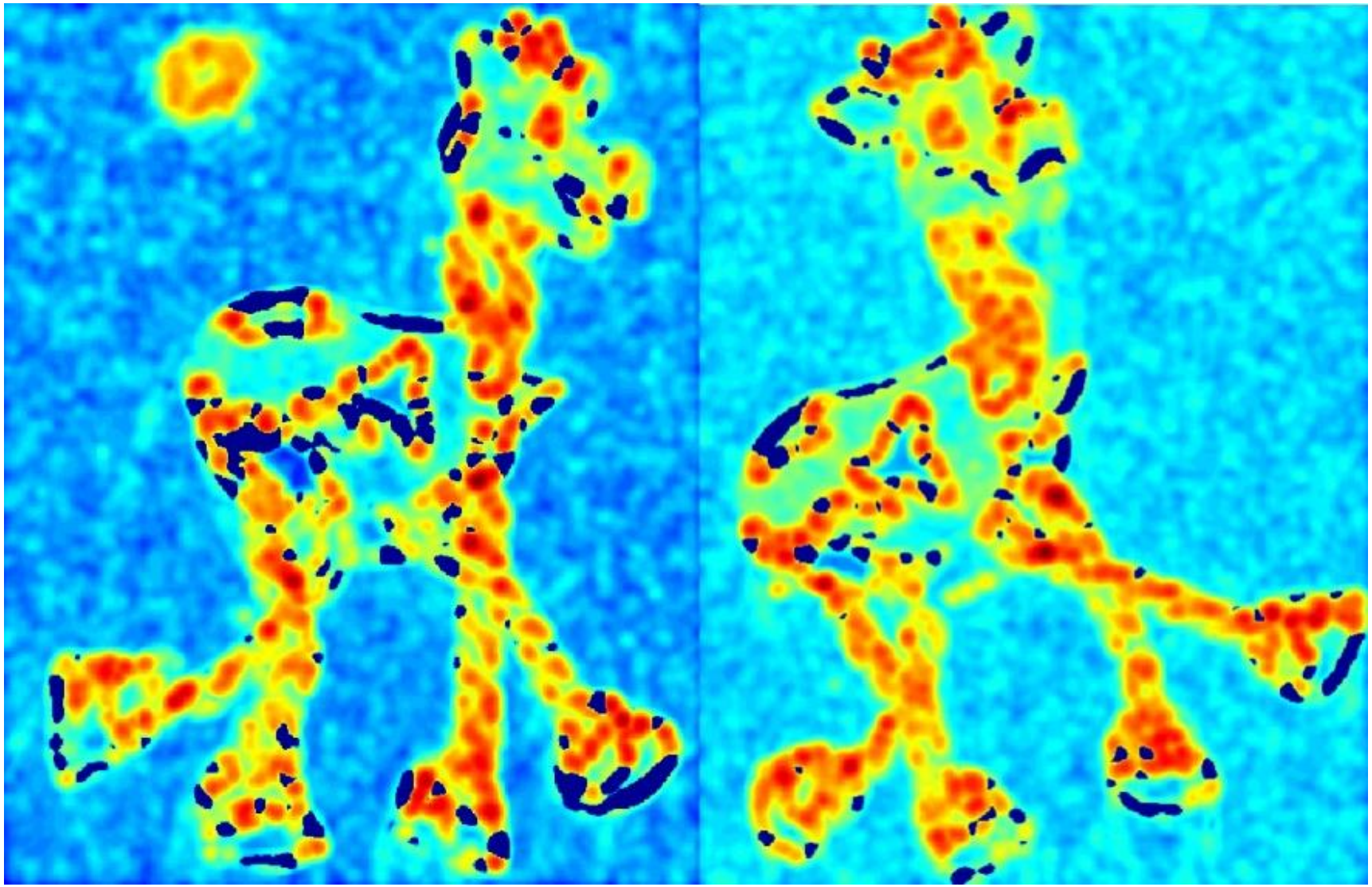


Harris operator

Harris Detector: Workflow

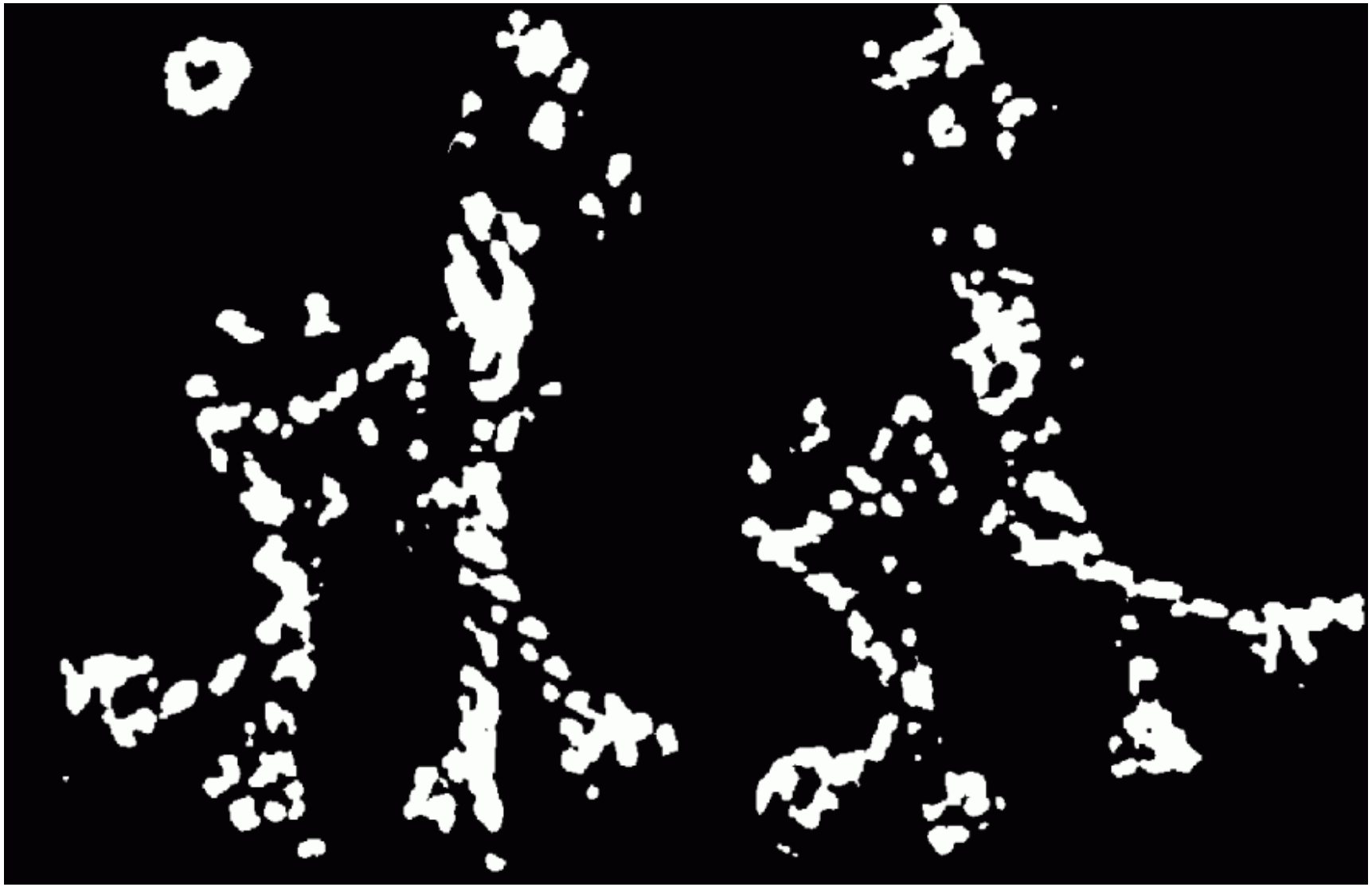
Harris Detector: Workflow

- Compute corner response R



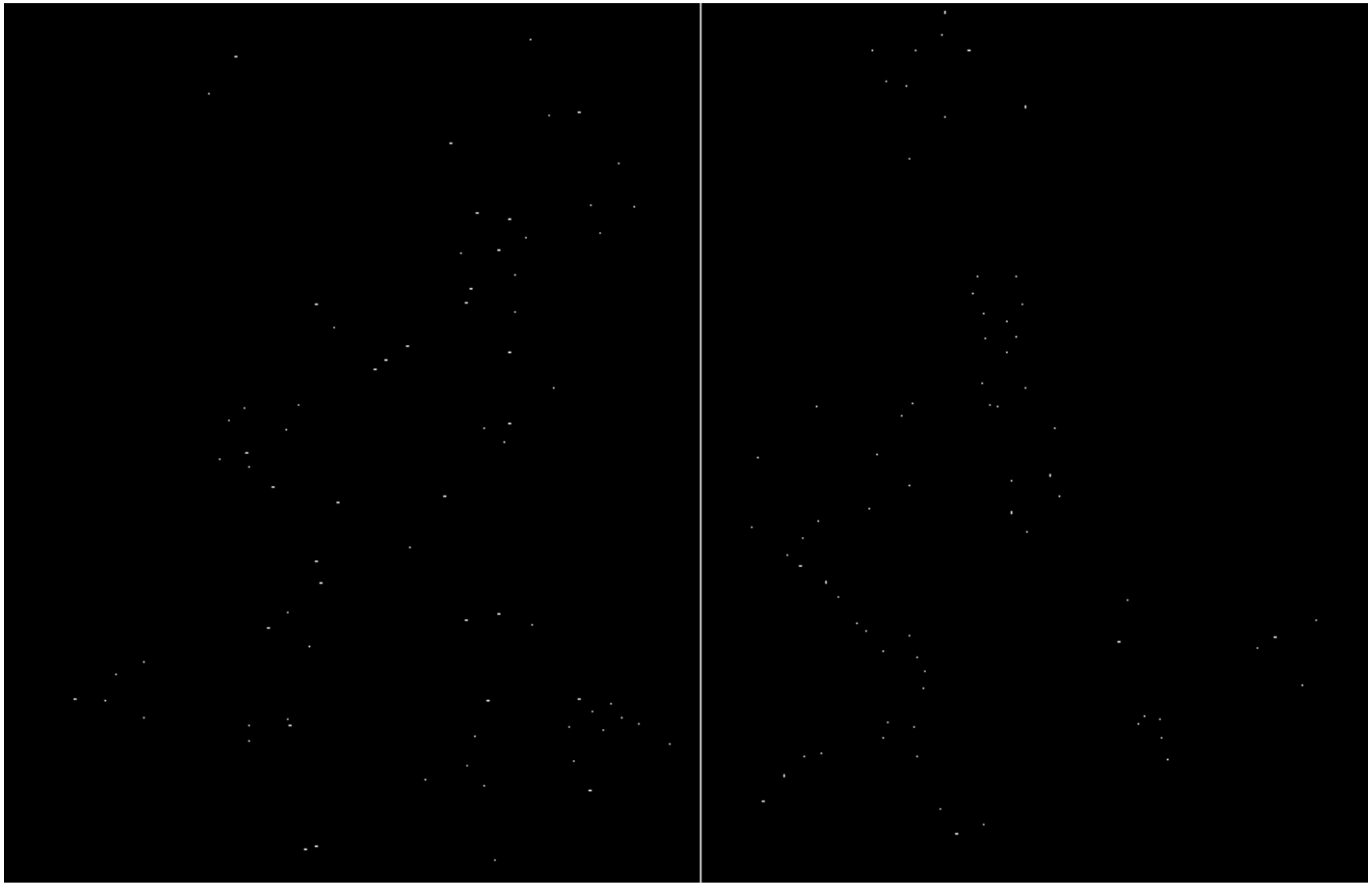
Harris Detector: Workflow

- 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

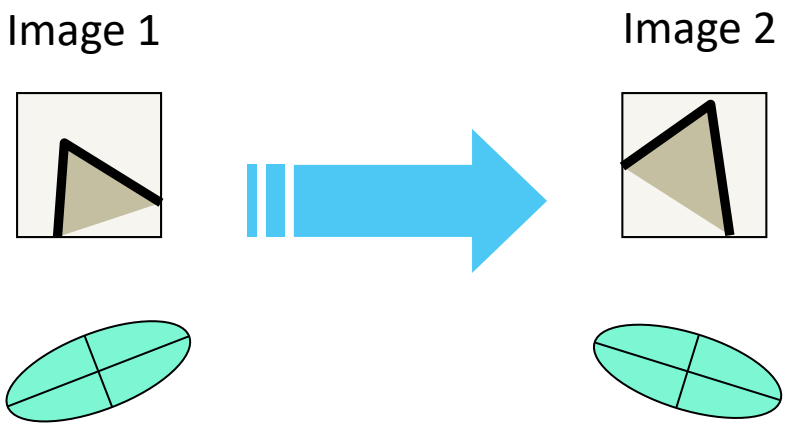
How does the size of the Harris detector affect the performance?

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**

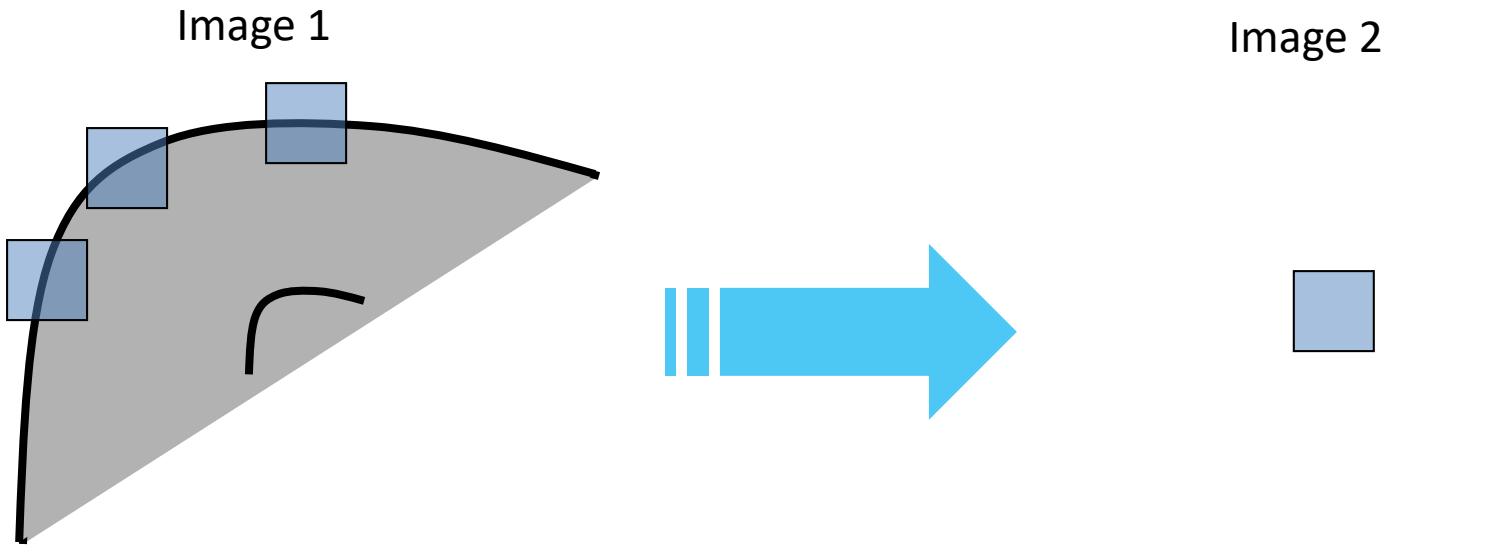


Ellipse rotates but its shape (i.e., eigenvalues) remains the same

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**

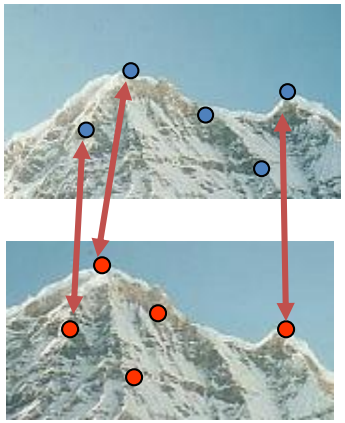
Corner!

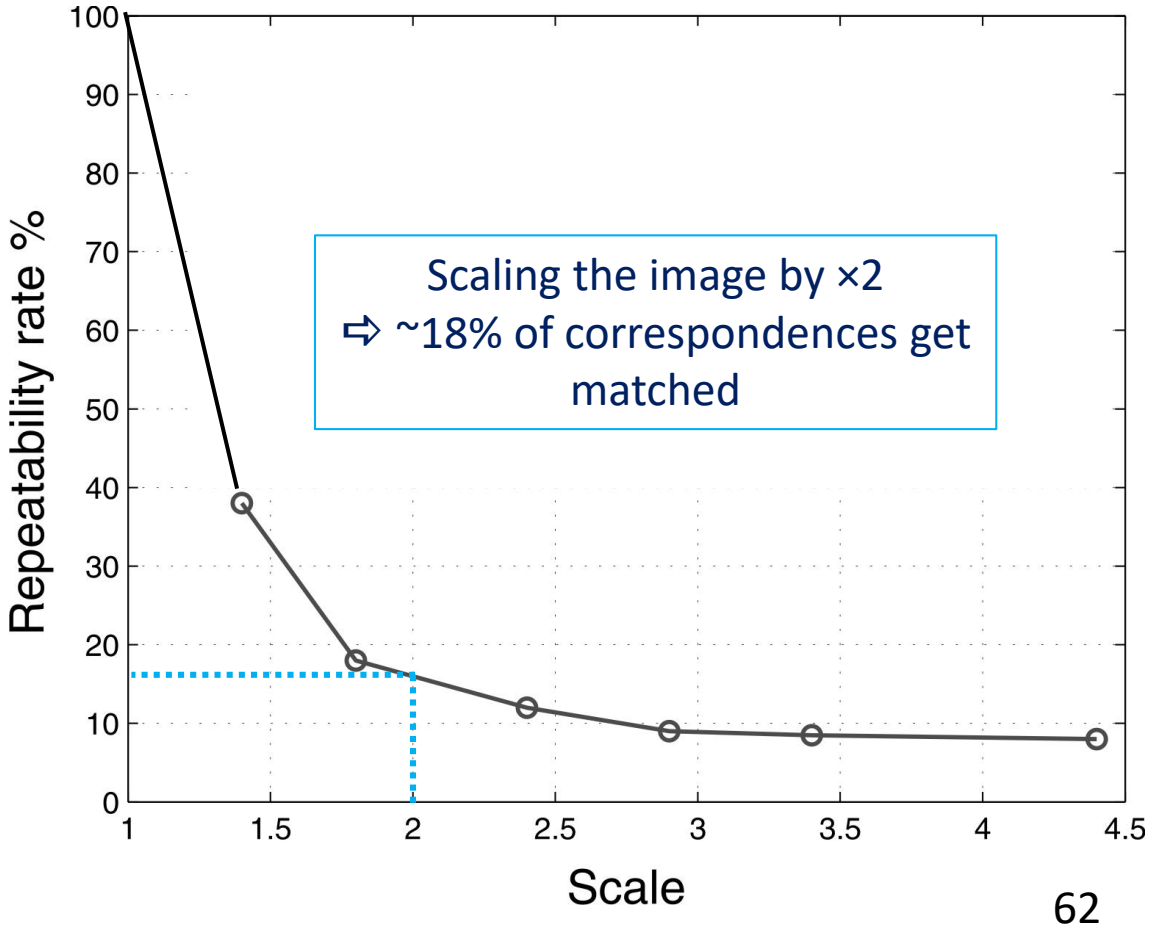
Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability=

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





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 Szeliski book and Ch. 8.1
  - Ch. 4 of Autonomous Mobile Robots book
  - Ch. 13.3 of Peter Corke book