ROBOTICS & PERCEPTION GROUP

Universität Zürich[UZH]

Lecture 05
Point Feature Detection and Matching

Davide Scaramuzza

Mini-project

Goal: implement a Visual Odometry (VO) pipeline

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

Grading:

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

Lab Exercise 3 - Today afternoon

- Room ETH HG E 33.1 from 14:15 to 16:00
- Work description: implement a corner detector and tracker

Course Schedule update

For updates, slides, and additional material: http://rpg.ifi.uzh.ch/teaching.html

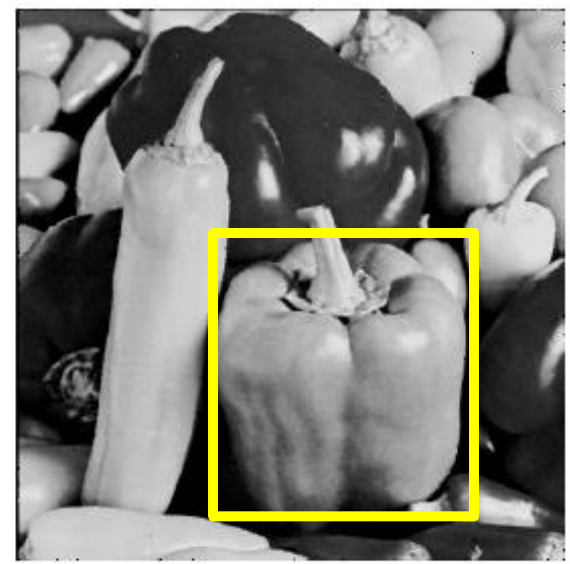
Date	Time	Description of the lecture/exercise	Lecturer
22.09.2016	10:15 - 12:00	01 – Introduction	Scaramuzza
29.09.2016	10:15 - 12:00	02 - Image Formation 1: perspective projection and camera models	Scaramuzza
06.10.2016	10:15 - 12:00	03 - Image Formation 2: camera calibration algorithms	Scaramuzza
	14:15 – 16:00	Lab Exercise 1: Augmented reality wireframe cube	Titus Cieslewski/Henri Rebecq
13.10.2016	10:15 - 12:00 14:15 – 16:00	04 - Filtering & Edge detection Lab Exercise 2: PnP problem	Gallego Titus Cieslewski/Henri Rebecq
20.10.2016	10:15 - 12:00	05 - Point Feature Detectors 1: Harris detector	Scaramuzza
	14:15 – 16:00	Lab Exercise 3: Harris detector + descriptor + matching	Titus Cieslewski/Henri Rebecq
27.10.2016	10:15 - 12:00	06 - Point Feature Detectors 2: SIFT, BRIEF, BRISK	Scaramuzza
3.11.2016	10:15 - 12:00 14:15 – 16:00	07 - Multiple-view geometry 1 Lab Exercise 4: Stereo vision: rectification, epipolar matching, disparity, triangulation	Scaramuzza Titus Cieslewski/Henri Rebecq
10.11.2016	10:15 - 12:00 14:15 – 16:00	08 - Multiple-view geometry 2 Exercise 5: Eight-point algorithm and RANSAC	Scaramuzza Titus Cieslewski/Henri Rebecq
17.11.2016	10:15 - 12:00 14:15 – 16:00	09 - Multiple-view geometry 3 Exercise 6: P3P algorithm and RANSAC	Scaramuzza Titus Cieslewski/Henri Rebecq
24.11.2016	10:15 - 12:00 14:15 – 16:00	10 - Dense 3D Reconstruction (Multi-view Stereo) Exercise 7: Intermediate VO Integration	Scaramuzza Titus Cieslewski/Henri Rebecq
01.12.2016	10:15 - 12:00 14:15 – 16:00	11 - Optical Flow and Tracking (Lucas-Kanade) Exercise 8: Lucas-Kanade tracker	Scaramuzza Titus Cieslewski/Henri Rebecq
08.12.2016	10:15 - 12:00	12 – Place recognition	Scaramuzza
	14:15 – 16:00	Exercise 9: Recognition with Bag of Words	Titus Cieslewski/Henri Rebecq
	10:15 - 12:00	13 – Visual inertial fusion	Scaramuzza
15.12.2016	14:15 – 16:00	Exercise 10: Pose graph optimization and Bundle adjustment	Titus Cieslewski/Henri Rebecq
22.12.2016	10:15 - 12:00 14:15 – 16:00	14 - Event based vision + lab visit and live demonstrations Exercise 11: final VO integration	Scaramuzza Titus Cieslewski/Henri Rebecq

Outline

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

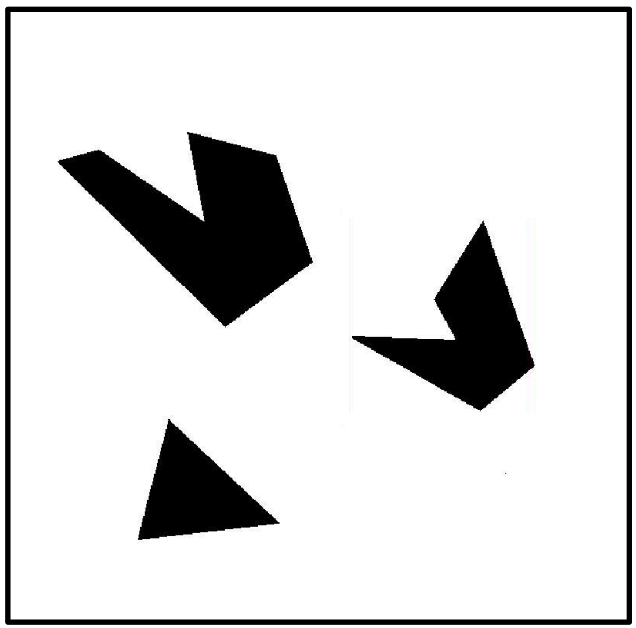
Filters for Feature Detection

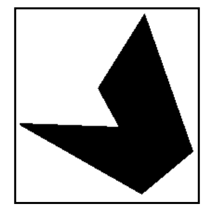
- Previously, we used filters as a way to remove or reduce **noise**
- However, filters can also be used to detect higher-level “**features**”.
  - Goal: reduce amount of data, discard redundancy, preserve only what is useful
    - Edge detection
    - Template matching
    - Keypoint detection



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** to detect these locations

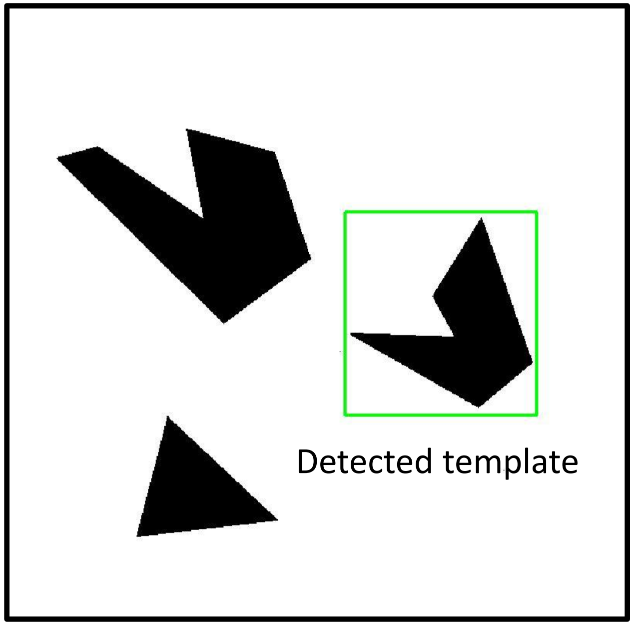


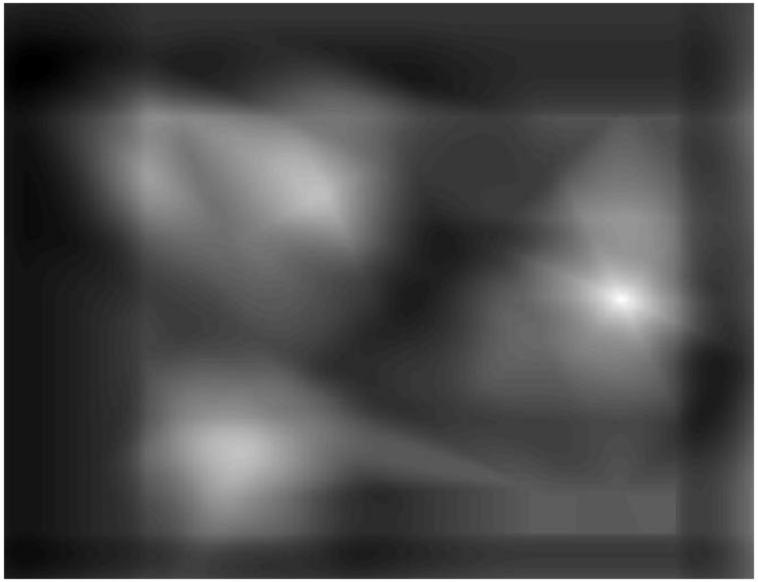


Template

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

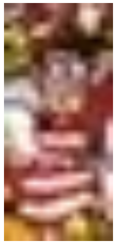




Correlation map

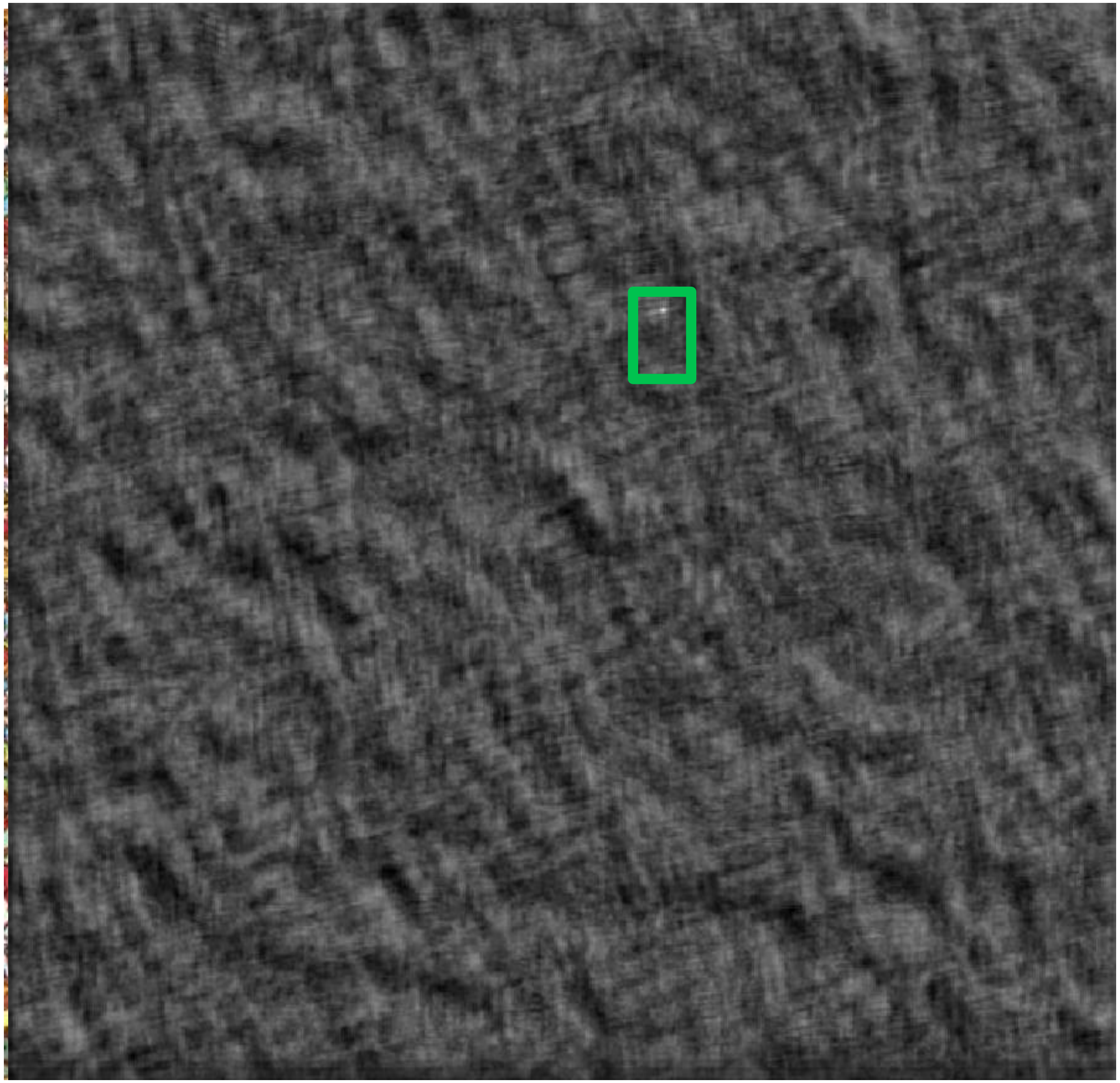
Where's Waldo?

Scene

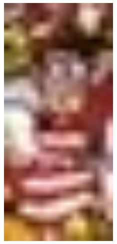


Template

Where's Waldo?



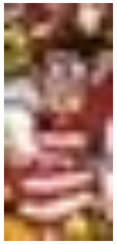
Scene



Template

Where's Waldo?

Scene



Template

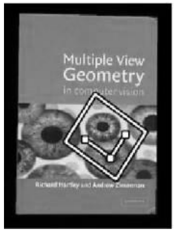
Template Matching

- What if the template is not identical to the object we want to detect?
- Matching can be meaningful if **scale**, **orientation**, **illumination**, and, in general, appearance between template and object to detect are very close. What about the pixels in **template background** (*mixed-pixel problem*)?

Scene

Template

Scene

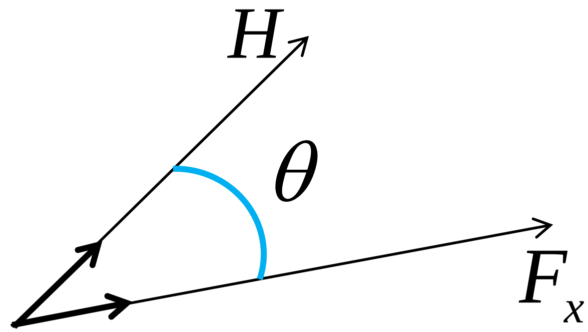


Template

Correlation as an Inner Product

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

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



- In **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

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

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”
  - Correlation as Scalar Product

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 illlumination 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**)

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

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

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

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

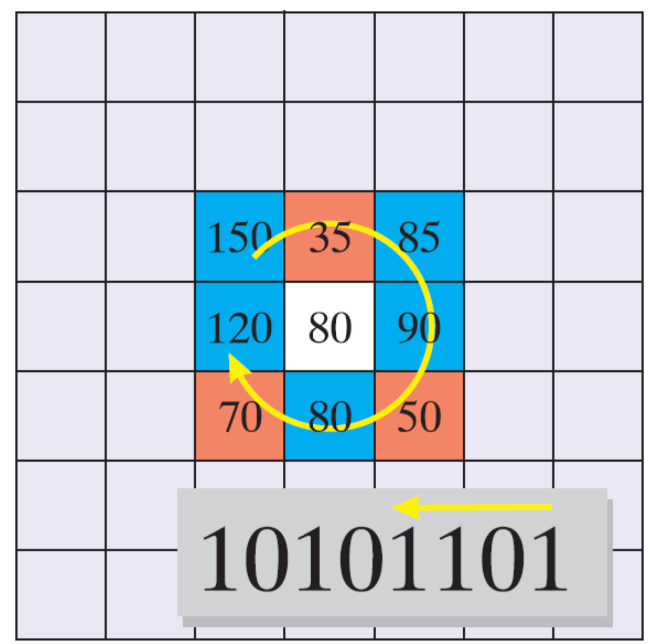
$$ZNCC = \frac{\sum_{u=-k}^{k} \sum_{v=-k}^{k} \left(H(u,v) - \mu_H \right) \left(F(u,v) - \mu_F \right)}{\sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} \left(H(u,v) - \mu_H \right)^2} \sqrt{\sum_{u=-k}^{k} \sum_{v=-k}^{k} \left(F(u,v) - \mu_F \right)^2}}$$

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 $w^2 - 1$ 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 set bits in the Exclusive-OR of the two bit strings

Advantages

- More robust to *mixed-pixel problem*
- No square roots or division 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 overall changes in intensity or gradual intensity gradients



Outline

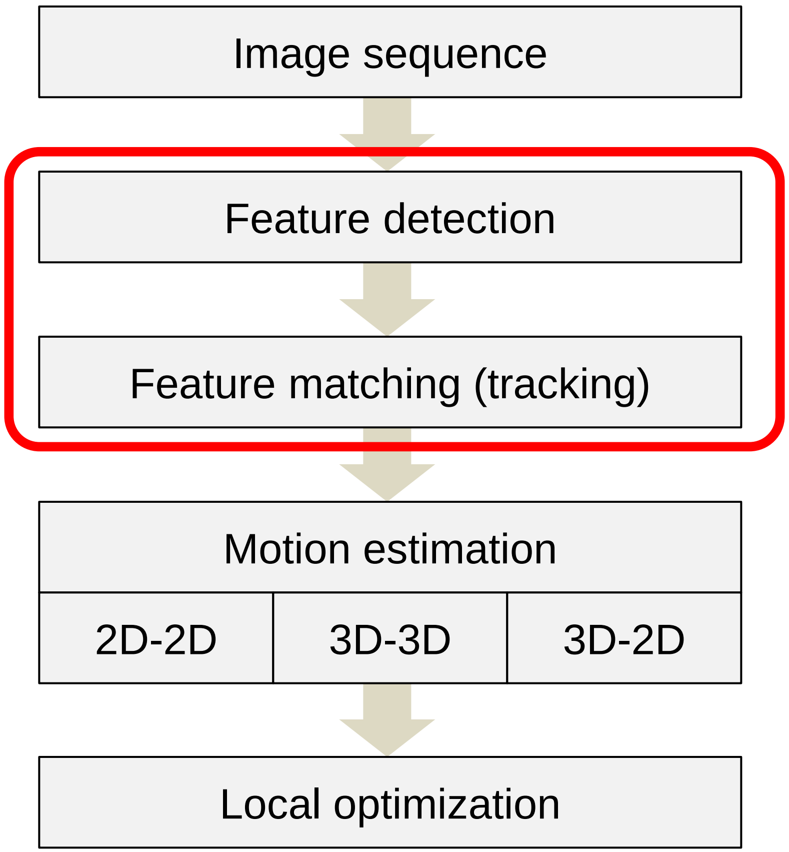
- Filters for feature extraction
- Point-feature extraction: today and next lecture

Point-feature extraction and matching Example

Video from "Forster, Pizzoli, Scaramuzza, *SVO: Semi-Direct Visual Odometry*, ICRA'16]"

Why do we need to extract keypoints?

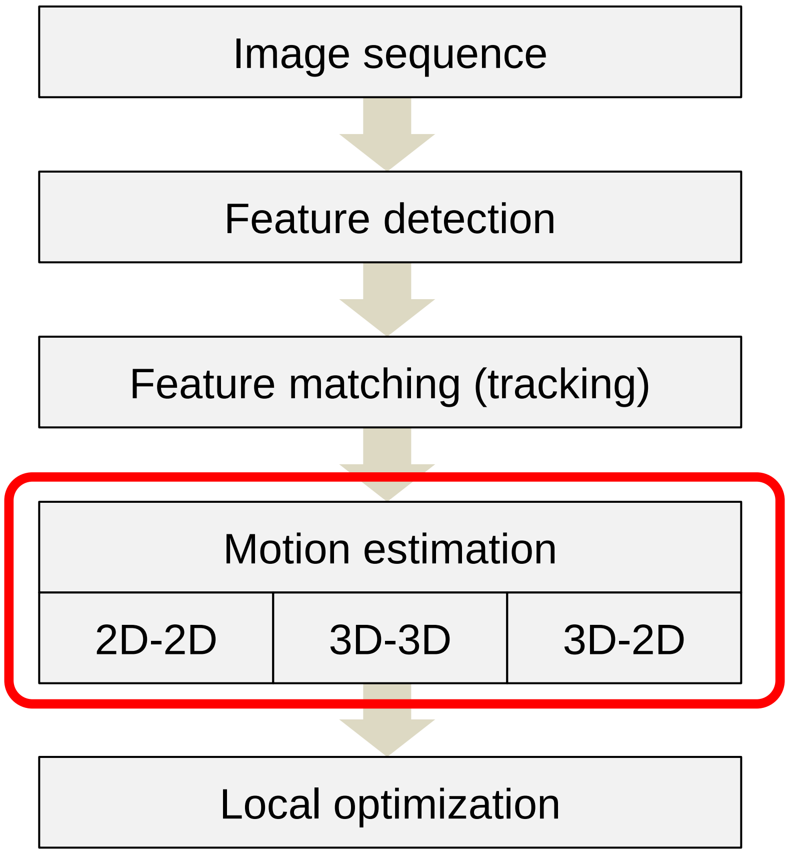
Recall the Visual-Odometry flow chart:

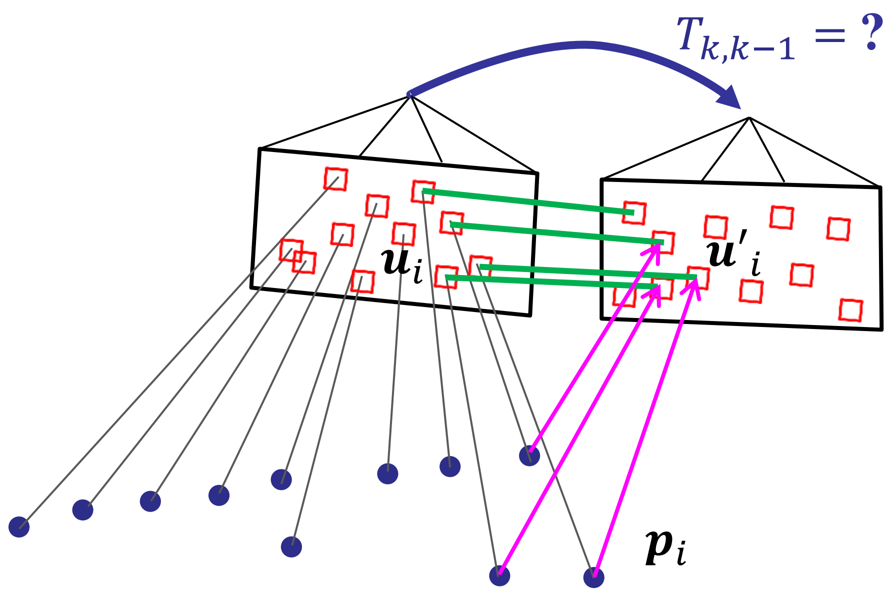


Example features tracks

Why do we need to extract keypoints?

Keypoint extraction is the key ingredient of motion estimation!





Point Features in image stitching

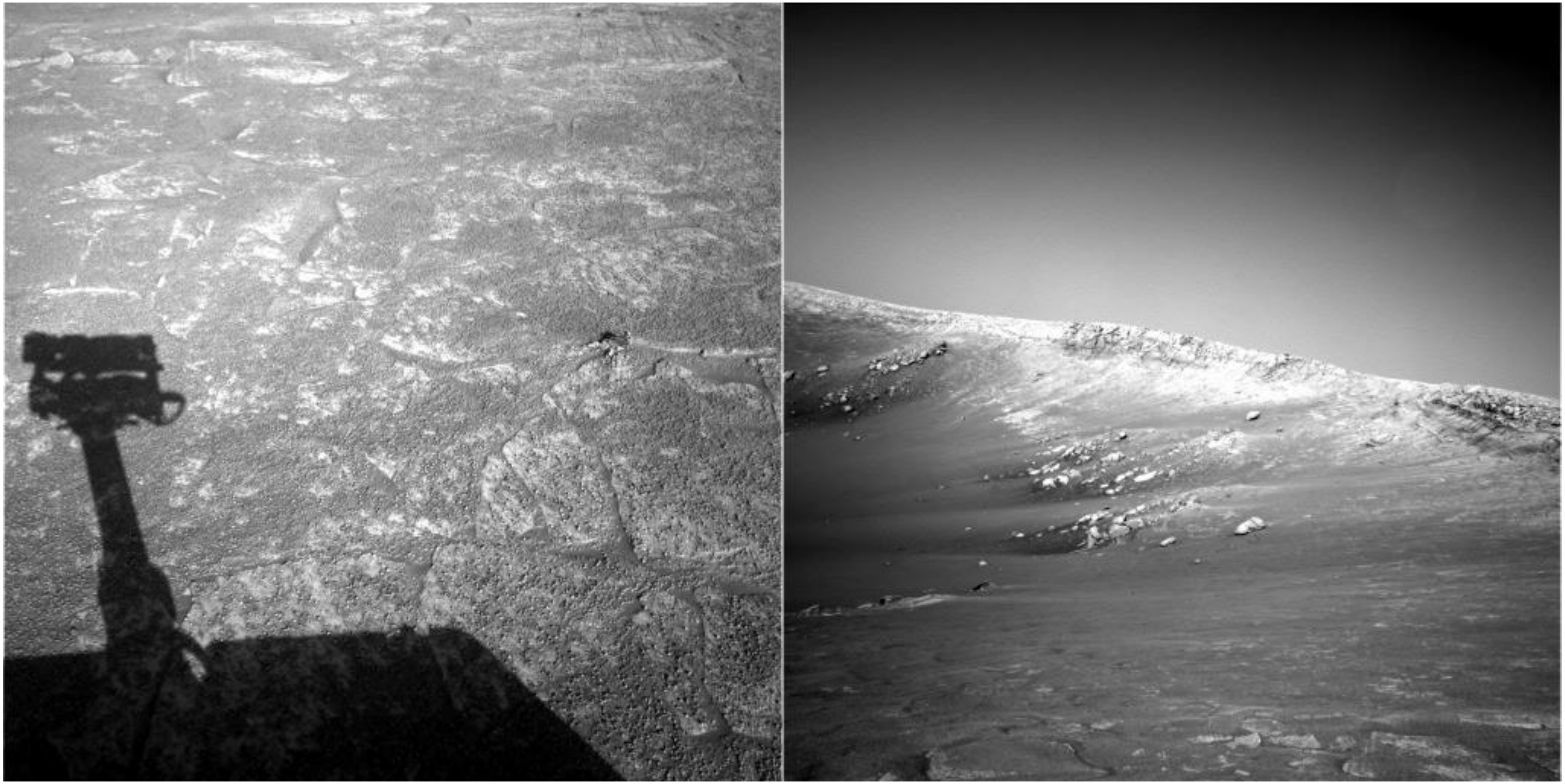
This panorama was generated using AUTOSTITCH:
http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Point Features are also used for:

- Object recognition
- 3D reconstruction
- Place recognition
- Indexing and database retrieval Google Images or http://tineye.com

Image matching: why is it hard?

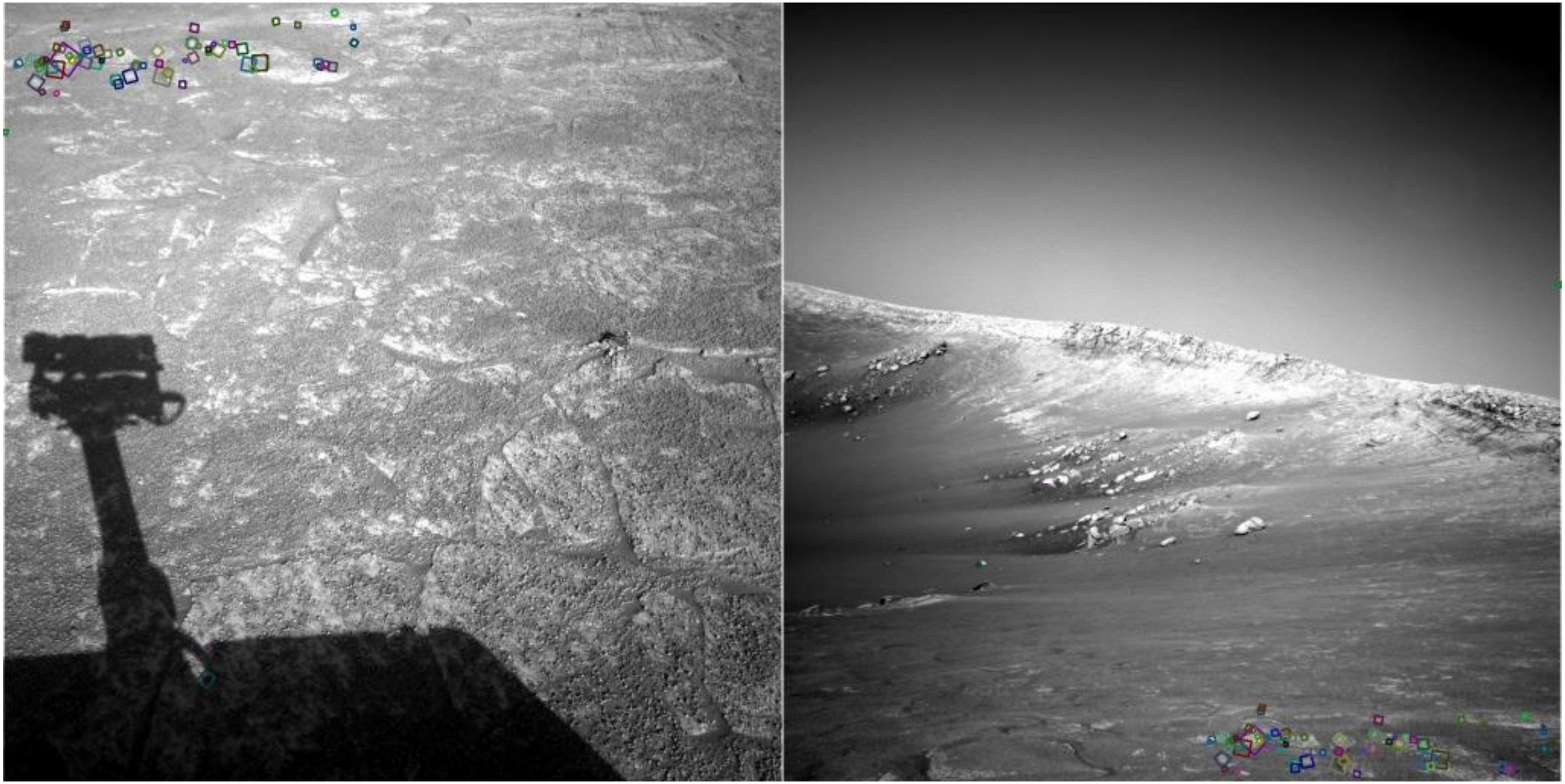
Image matching: why is it hard?



NASA Mars Rover images

Image matching: why is it hard?

Answer below



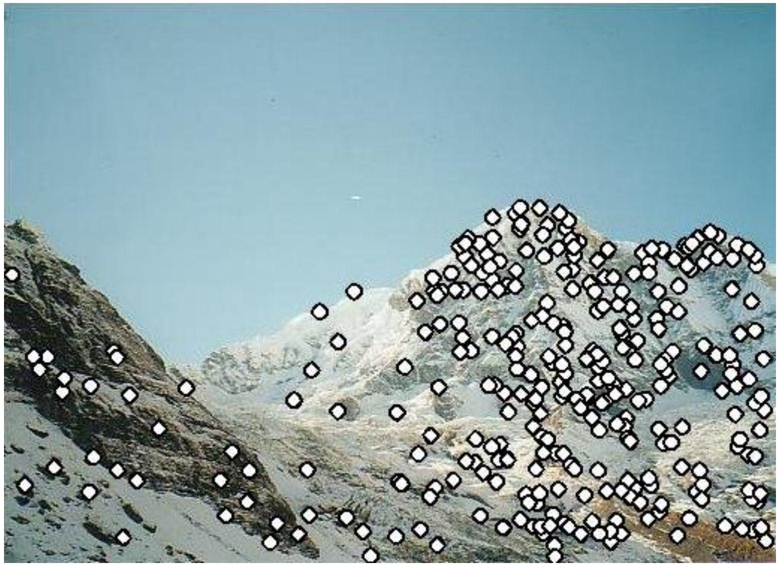
NASA Mars Rover images with SIFT feature matches

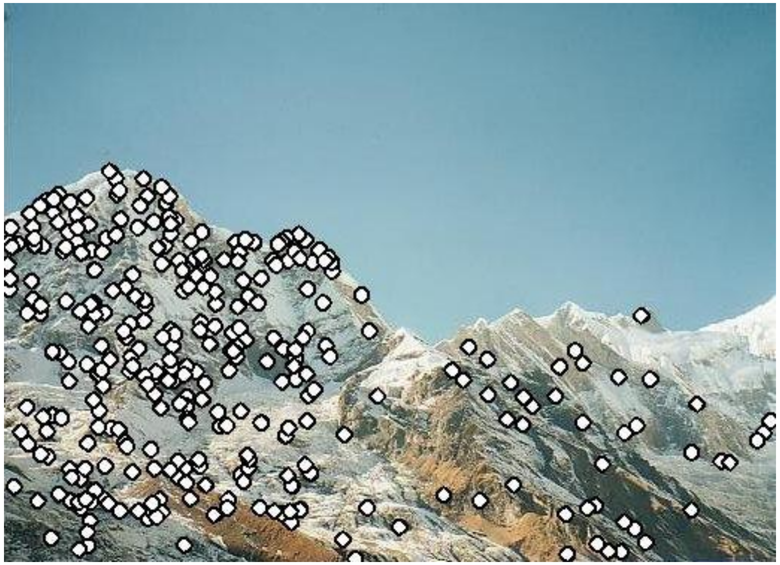
Local features and alignment

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

Local features and alignment

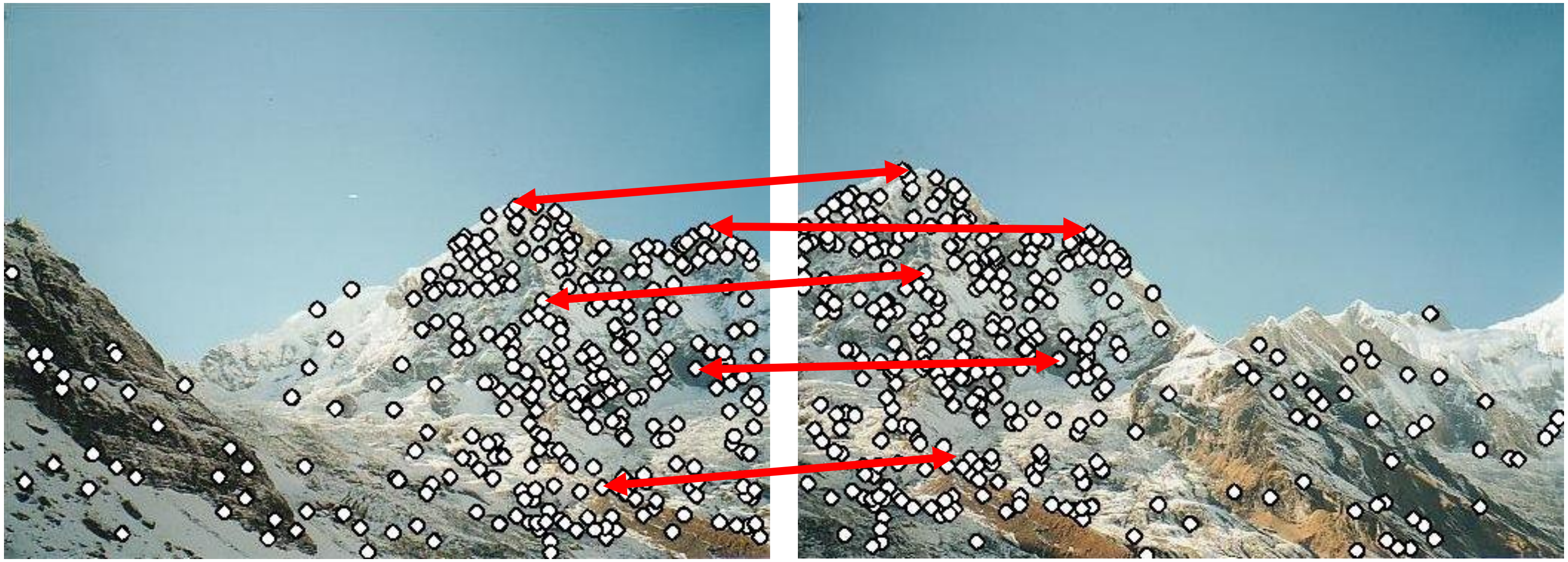
- Detect feature points in both images





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

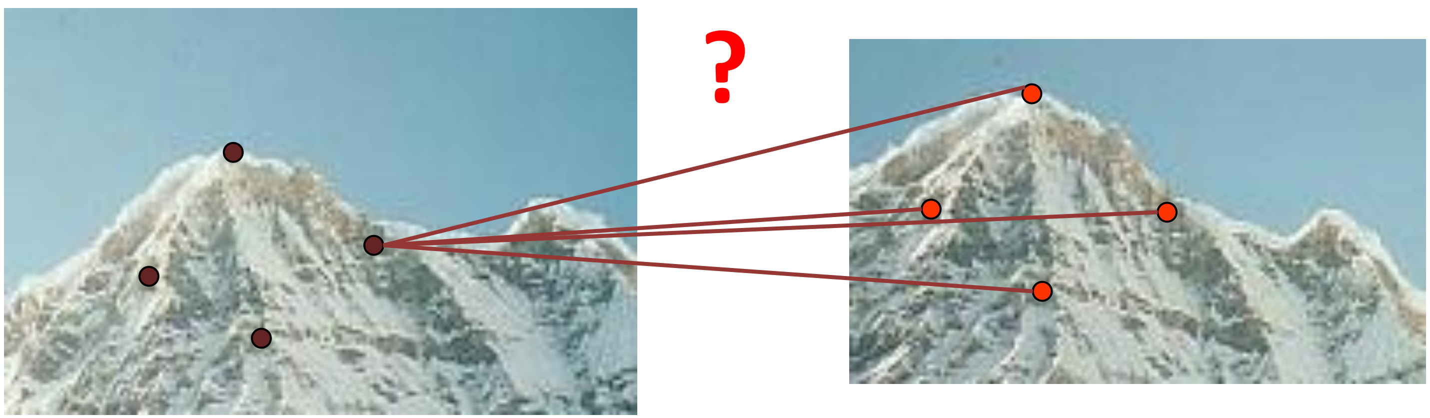
- Problem 1:
  - Detect the **same** points **independently** in both images

no chance to match!

We need a **repeatable** feature detector

Matching with Features

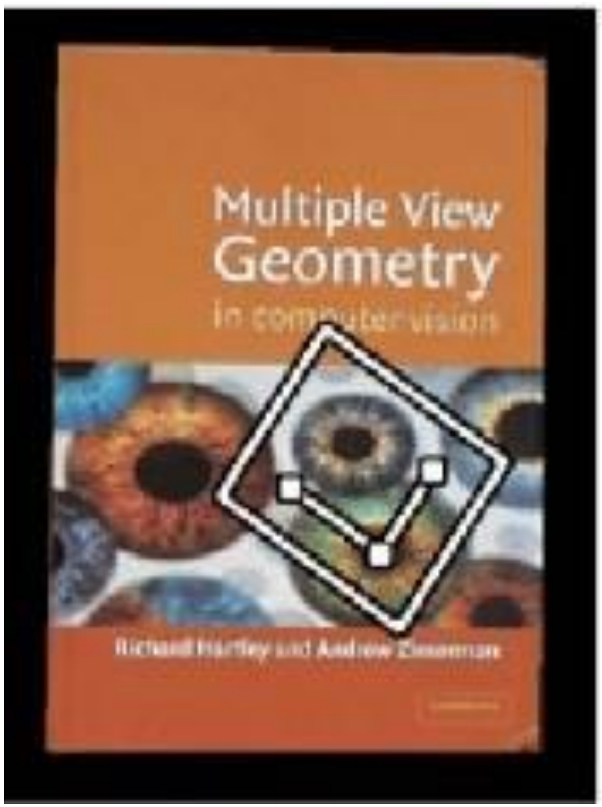
- Problem 2:
  - For each point, identify its correct correspondence in the other image(s)



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

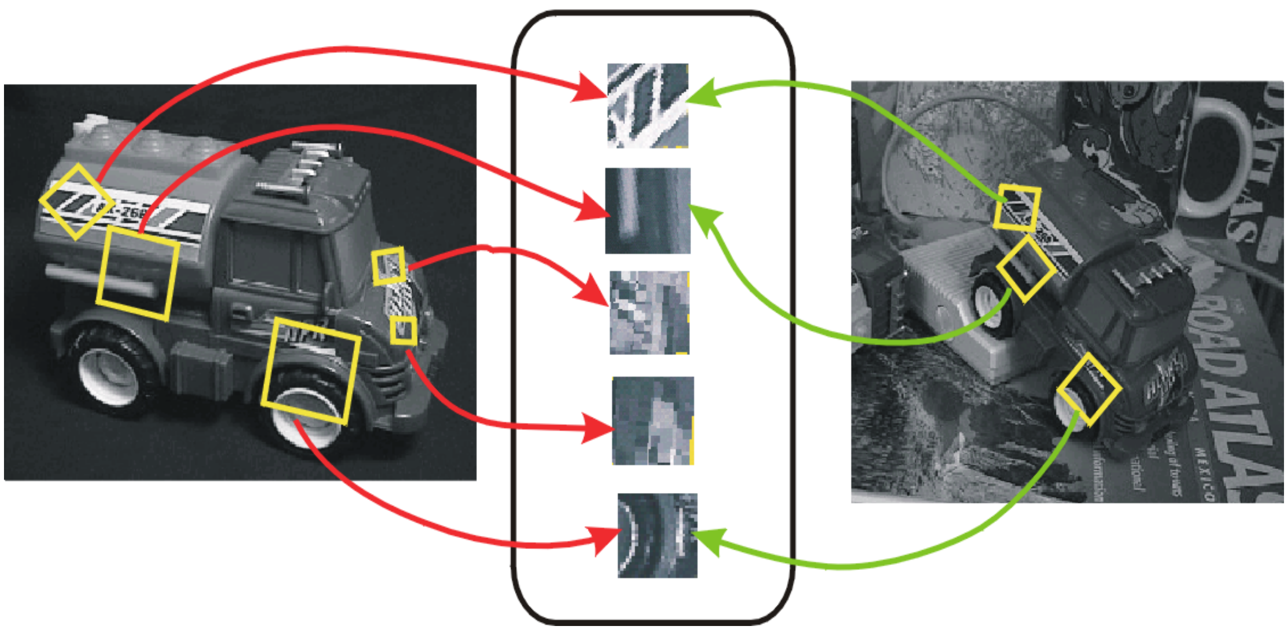
Invariant local features

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

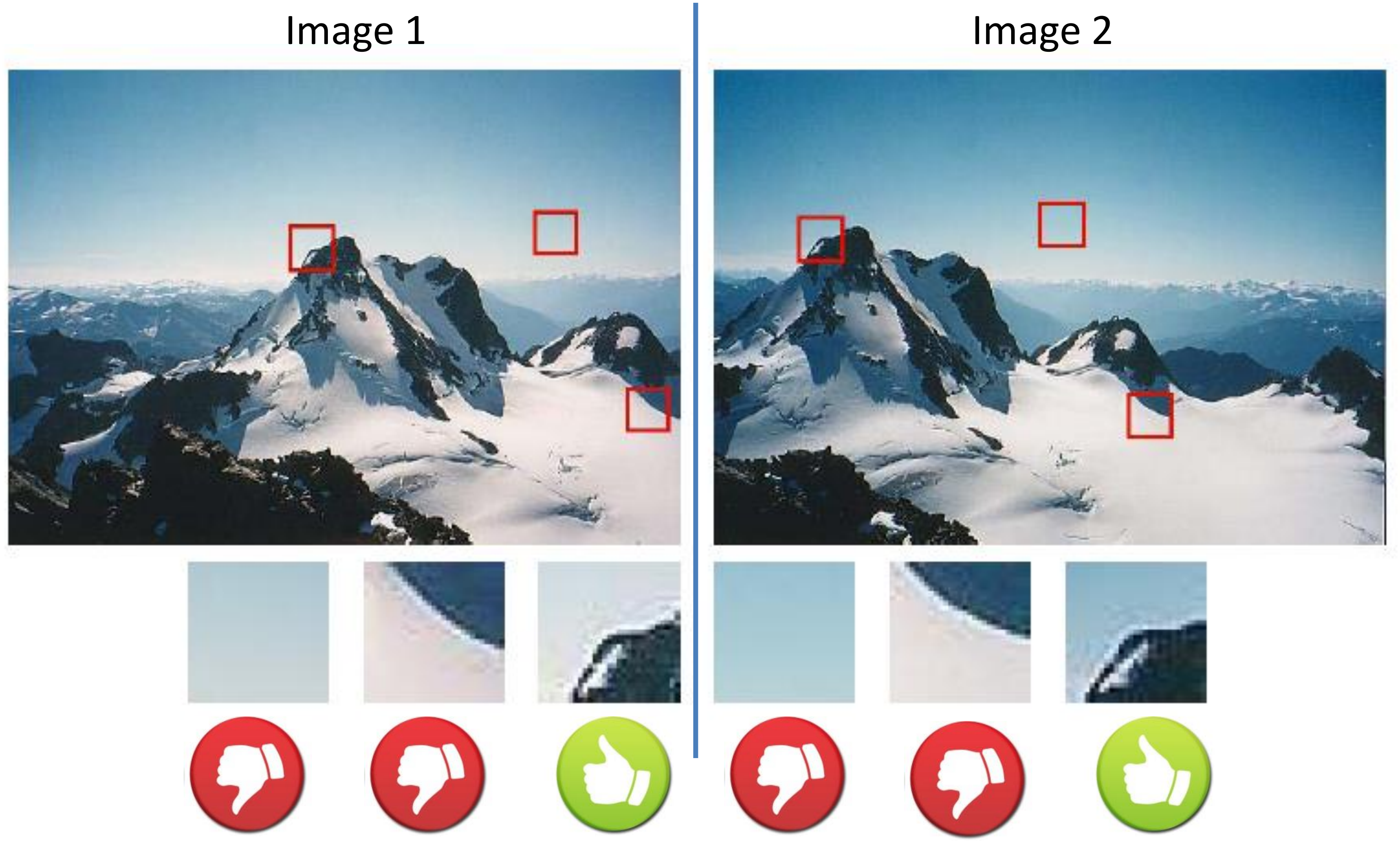


Main questions

- 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?

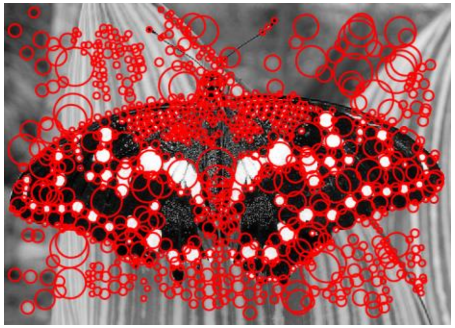
- Consider the image pair below with extracted patches
- Notice how some patches can be localized or matched with higher accuracy than others



Point Features: Corners vs 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*



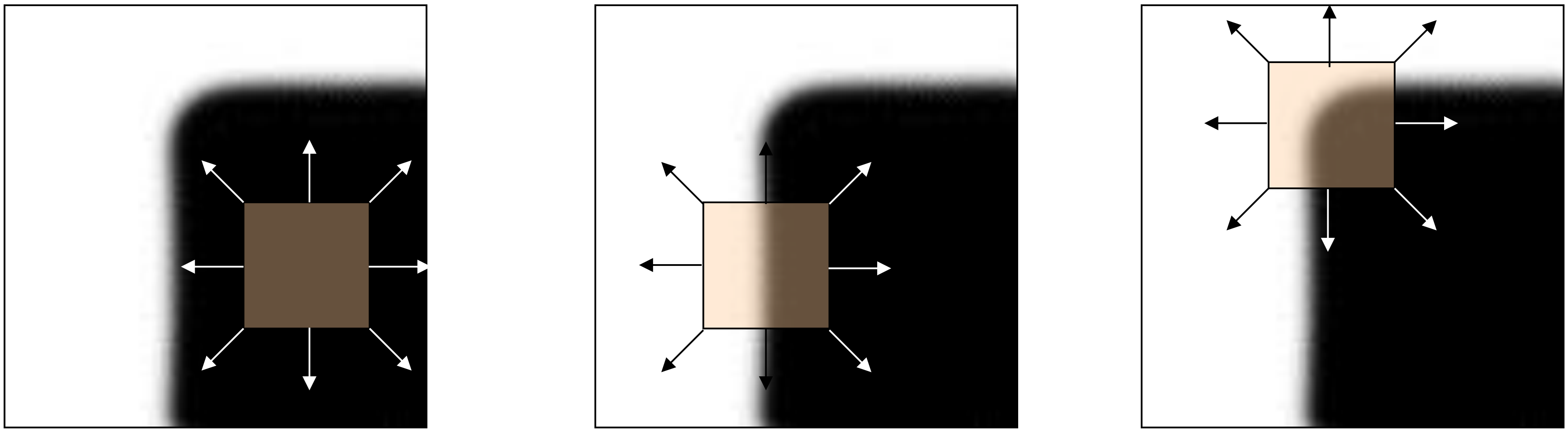
Corner detection

- 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.

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., $SSD \approx 0$ in all directions)

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

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

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:

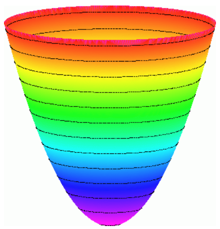
$$SSD(\Delta x, \Delta y) = \sum_{x,y\in P} \left(I(x,y) - I(x+\Delta x, y+\Delta y)\right)^2$$

- Let $I_x = \dfrac{\partial I(x,y)}{\partial x}$ and $I_y = \dfrac{\partial I(x,y)}{\partial y}$. 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

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

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

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

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

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

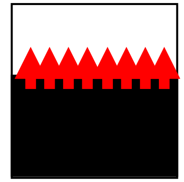
$$\Rightarrow SSD(\Delta x, \Delta y) \approx \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

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

$$M = \sum_{x,y \in P} \underbrace{\begin{bmatrix} I_x{}^2 & I_x I_y \\ I_x I_y & I_y{}^2 \end{bmatrix}}_{\text{2nd moment matrix}} = \underbrace{\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}}_{\text{Alternative way to write M}}$$

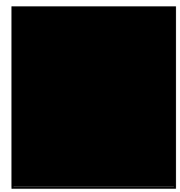
What does this matrix reveal?

- First, consider an edge or a flat region.



Edge

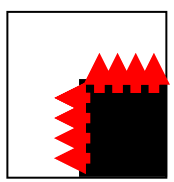
$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



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

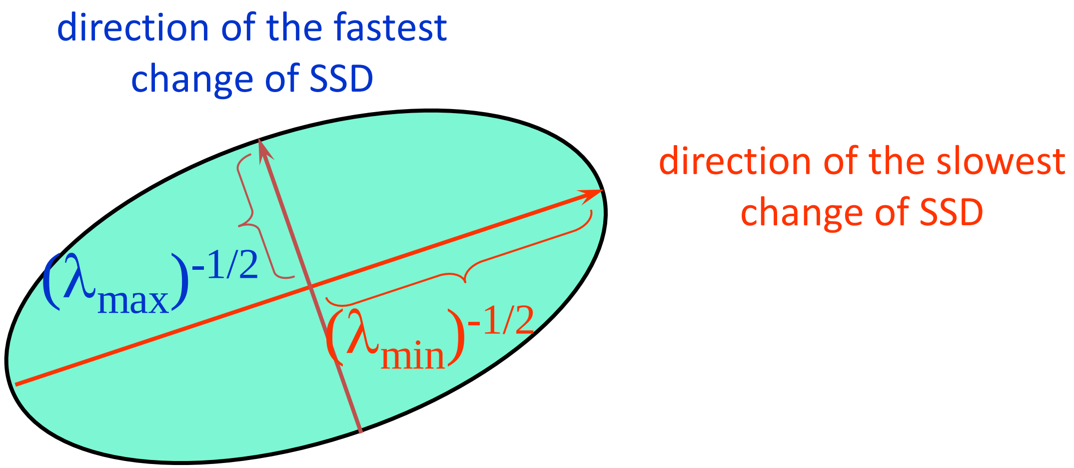
$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4}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

- 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 $\begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = const$ 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 $\lambda_1, \lambda_2, \mathrm{R}$ from M

Eigenvalue/eigenvector review

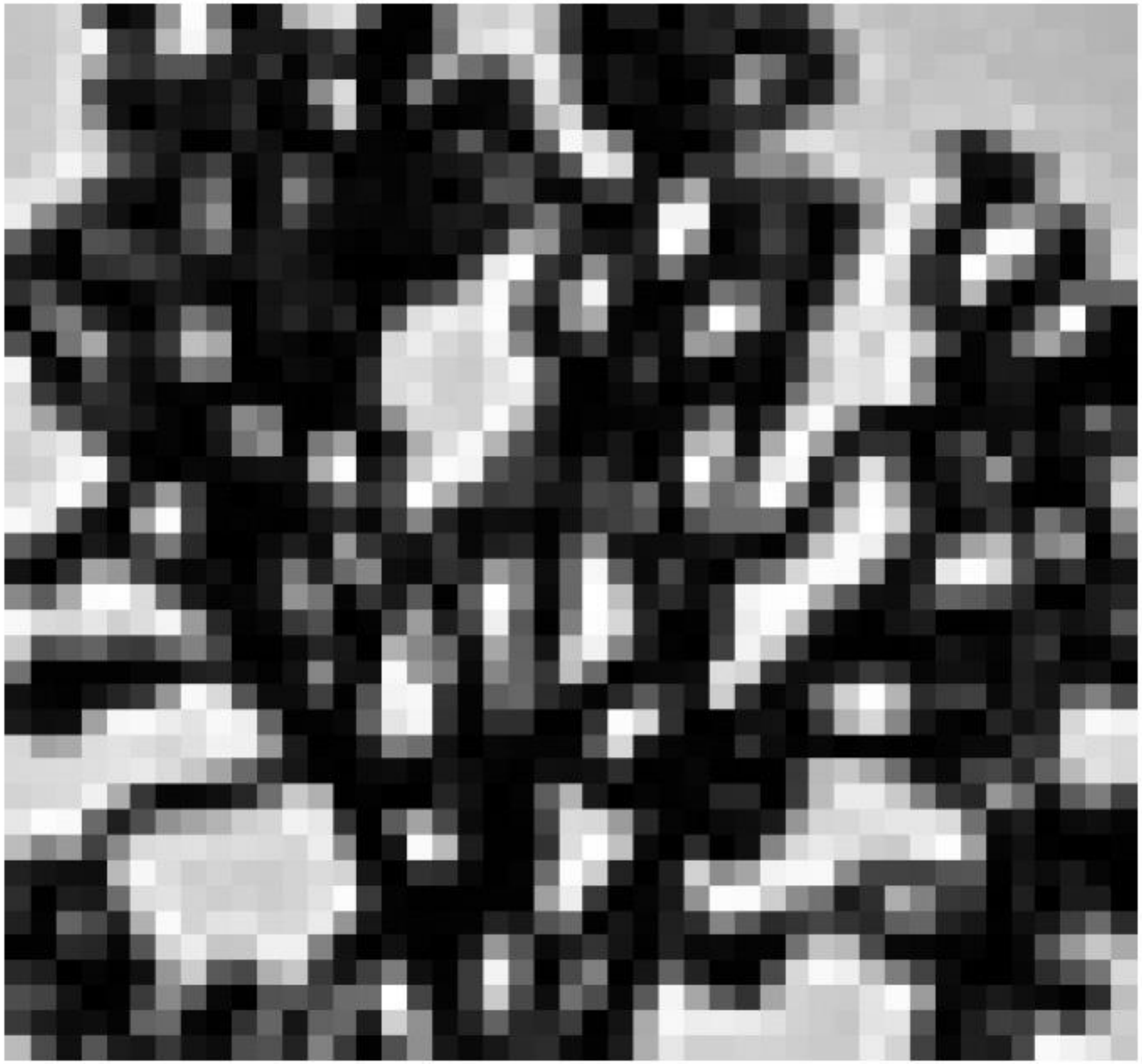
- You can easily proof that λ_1, λ_2 are the **eigenvalues** of M.
- The **eigenvectors** and **eigenvalues** of a matrix **A** are the vectors **x** and scalars λ that satisfy:

$$Ax = \lambda x$$

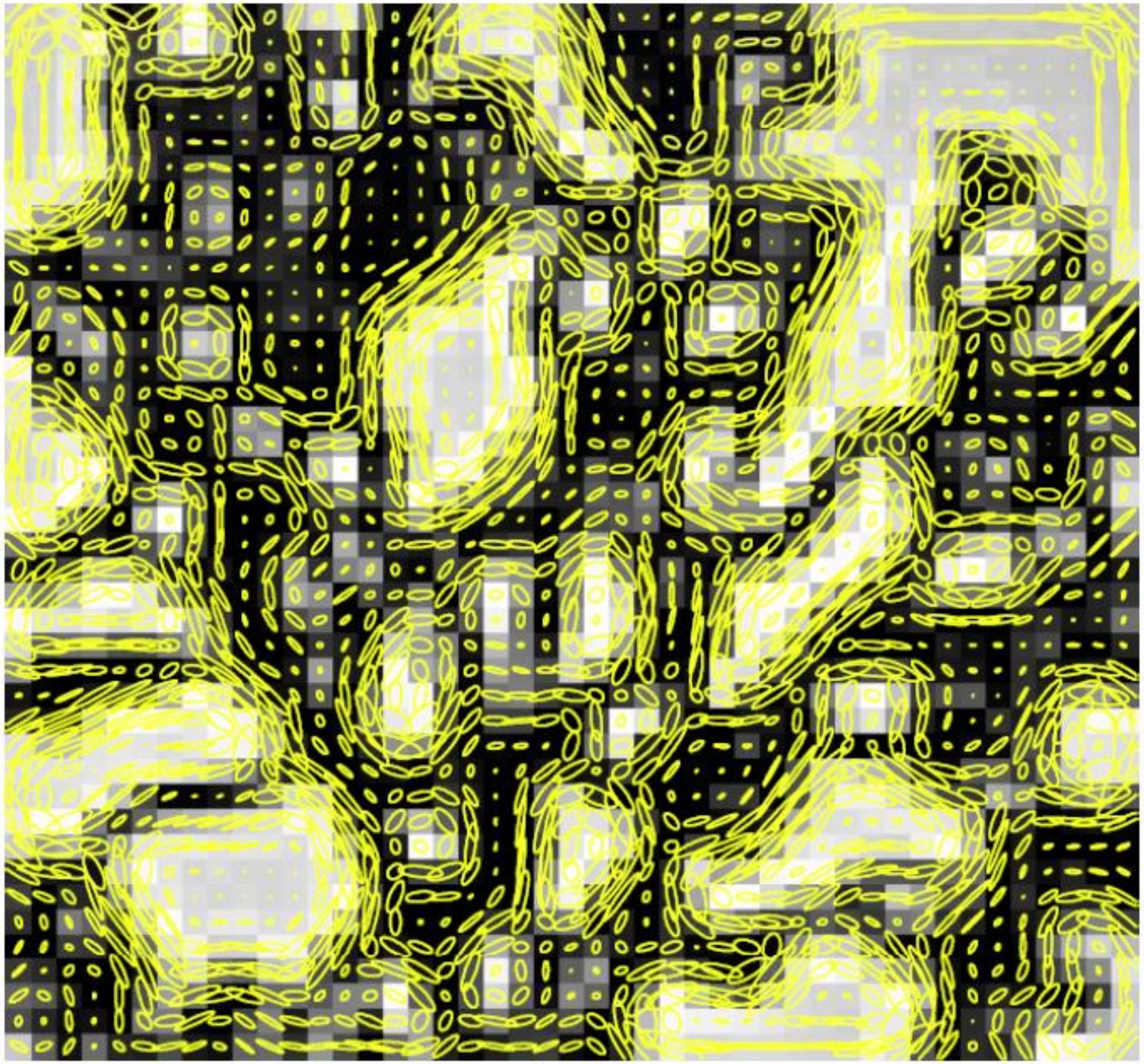
- The scalar λ is the **eigenvalue** corresponding to **x**
  - The eigenvalues are found by solving: $\det(A - \lambda I) = 0$
  - In our case, ***A*** = ***M*** is a 2x2 matrix, so we have $\det\begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix} = 0$
  - The solution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



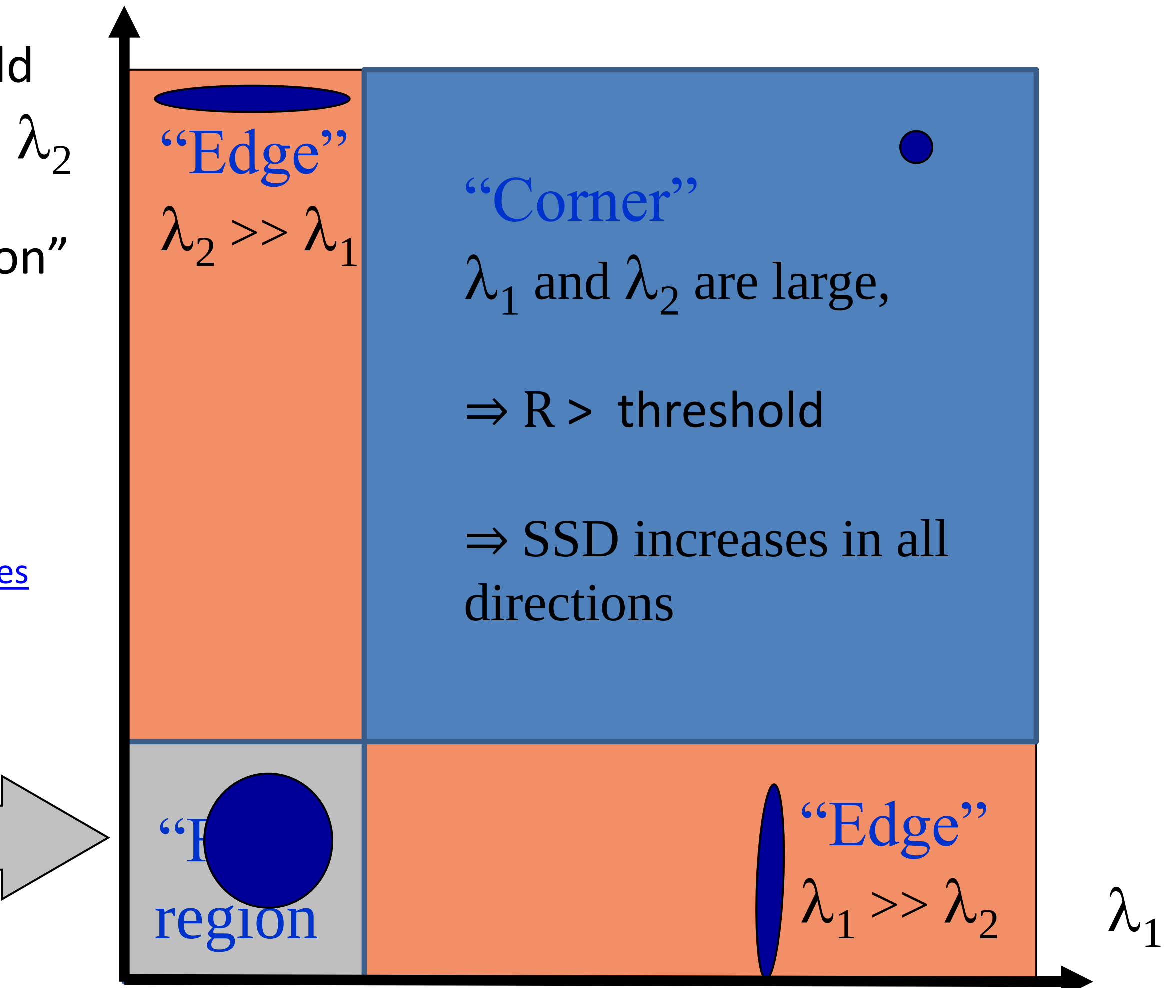
Interpreting the eigenvalues

- 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*



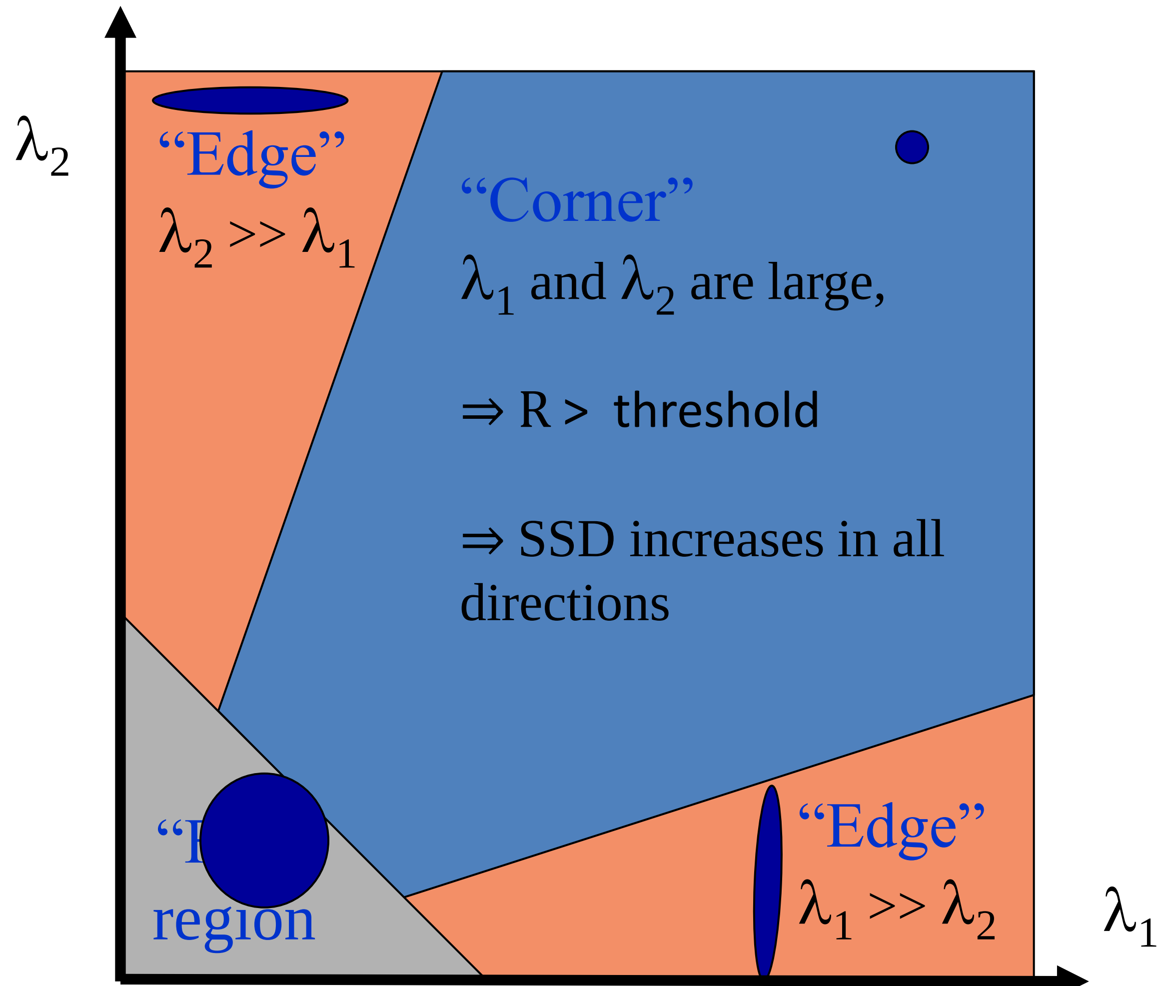
λ_1 and λ_2 are small;
SSD is almost constant in all directions

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

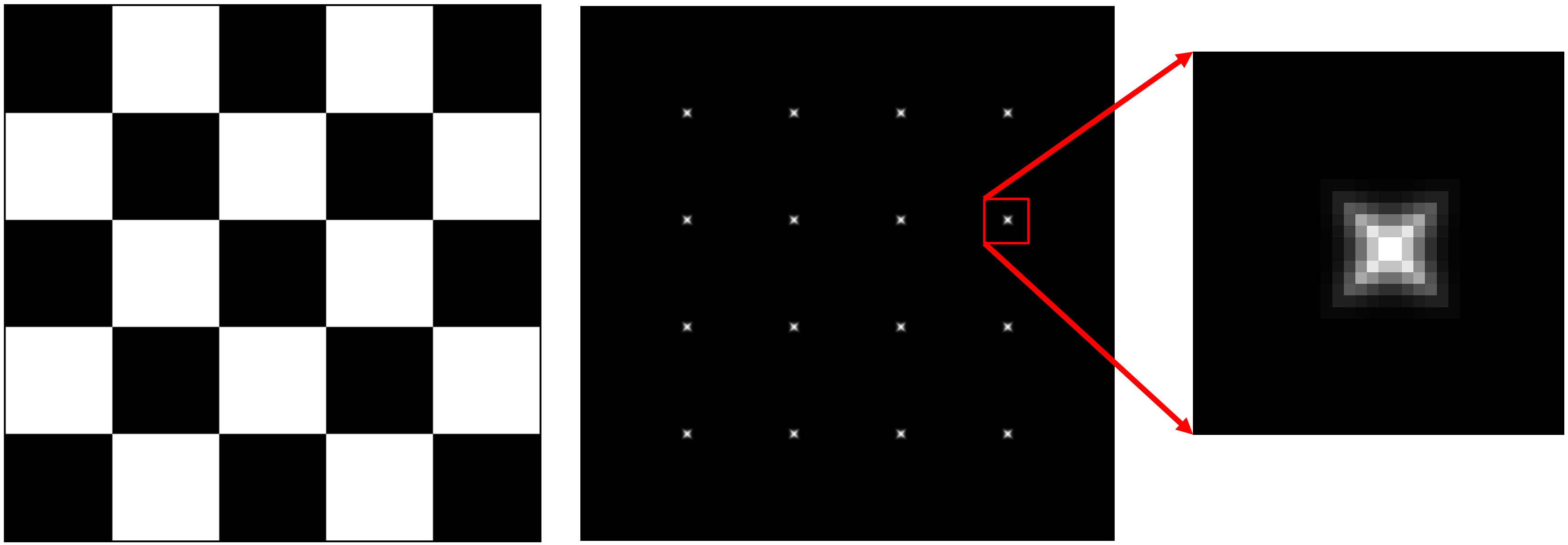
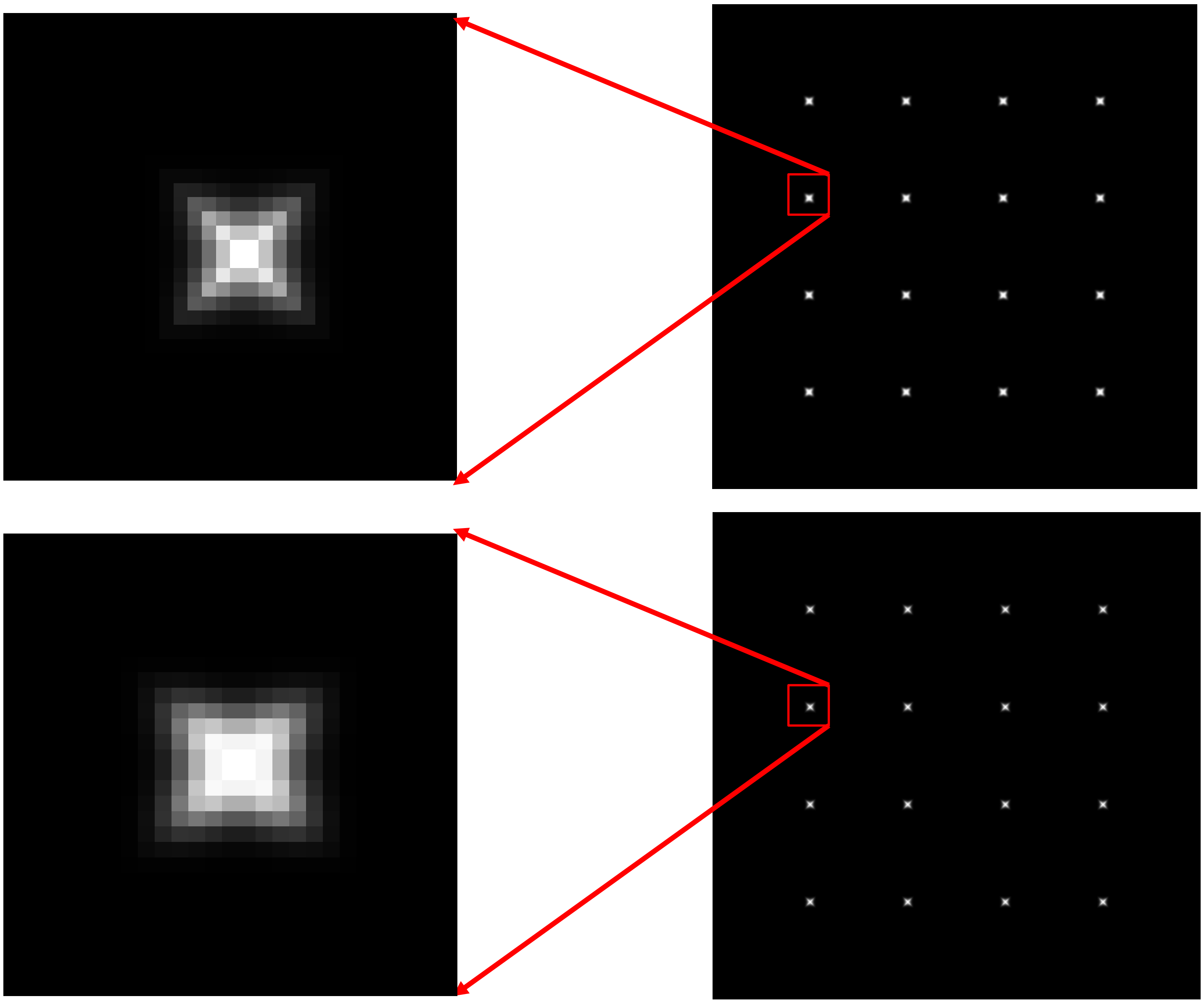


Image I

Cornerness response R

Harris vs. Shi-Tomasi



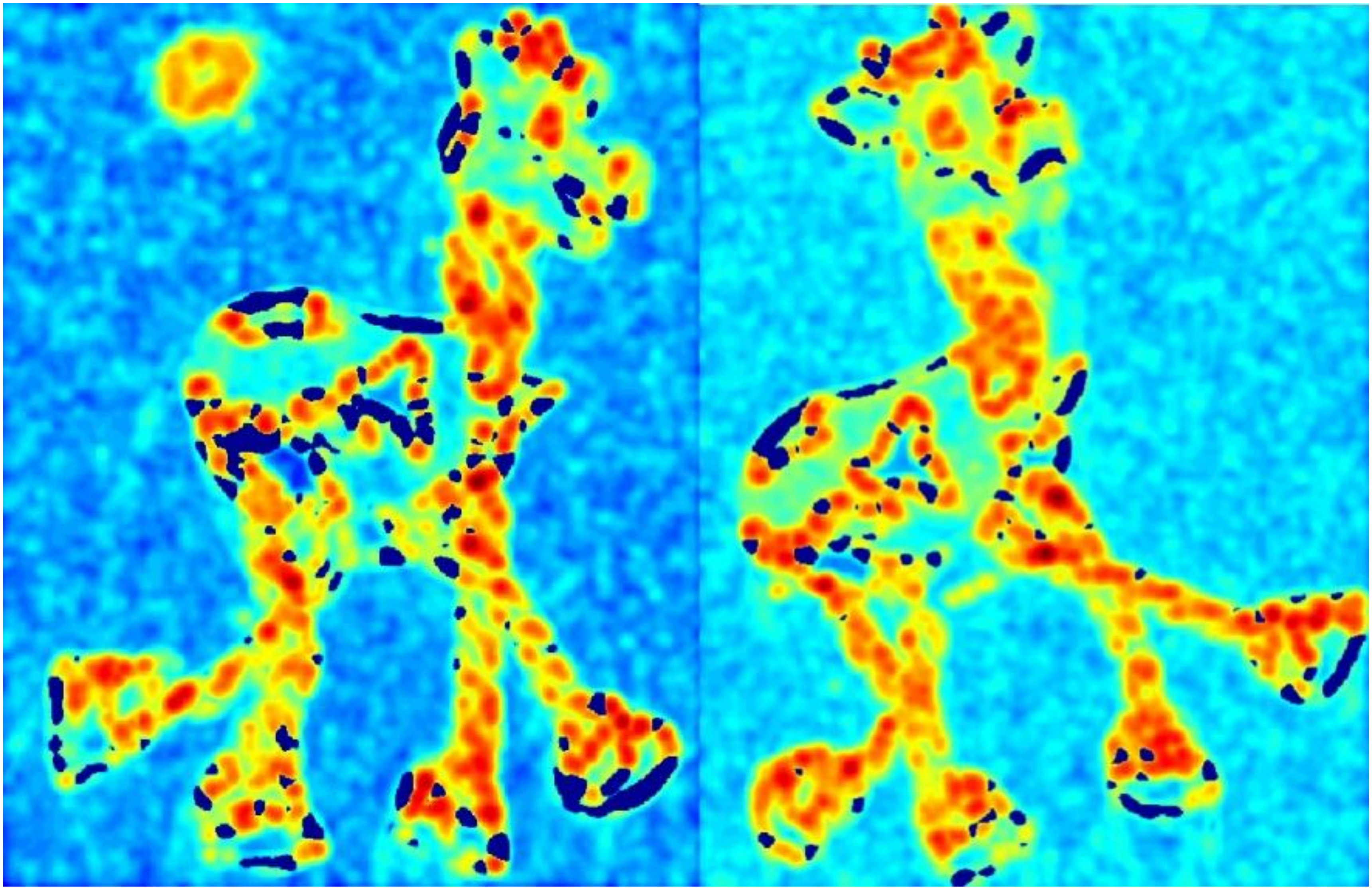
Shi-Tomasi operator

Harris operator

Harris Detector: Workflow

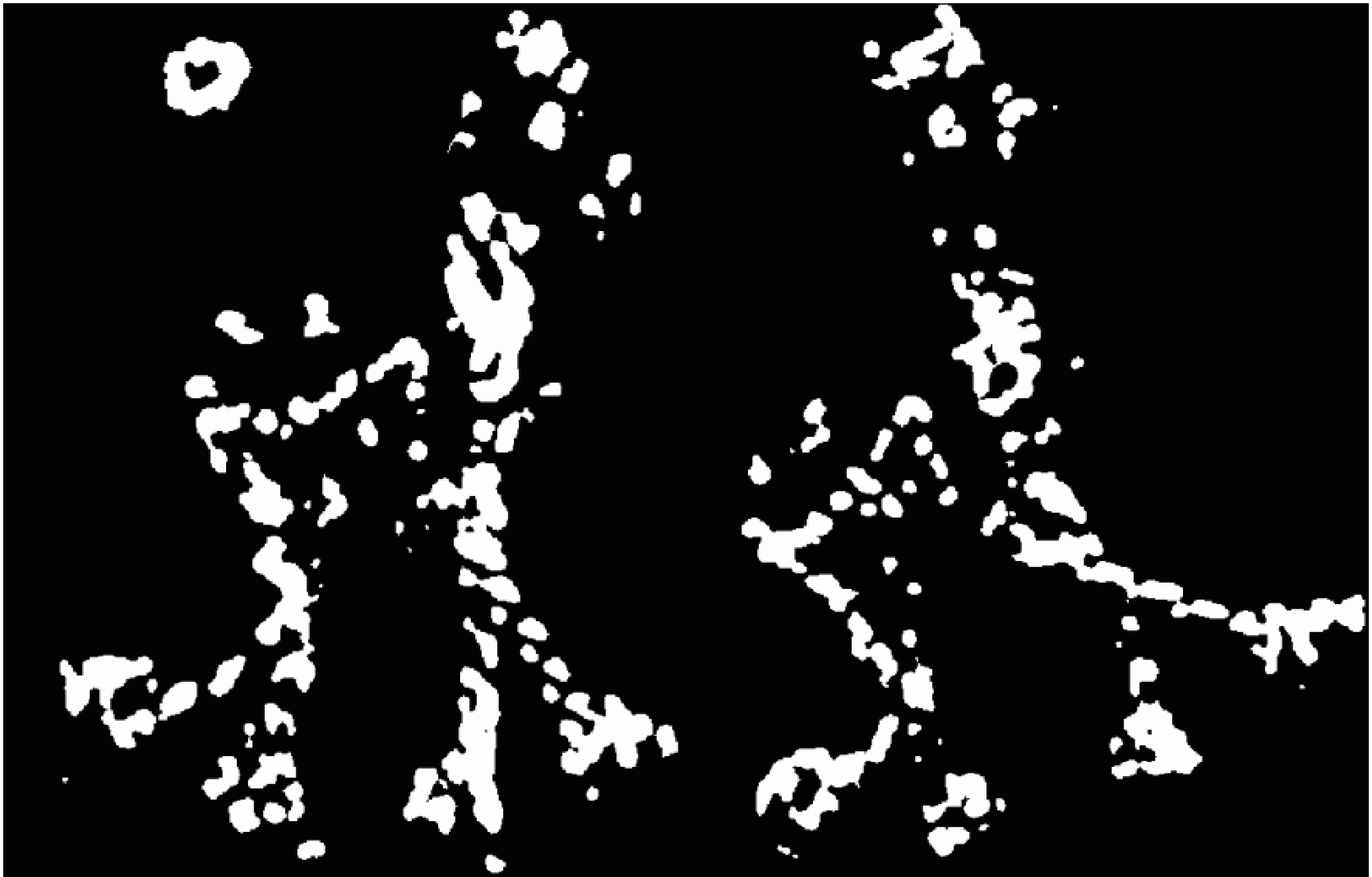
Harris Detector: Workflow

- Compute corner response R



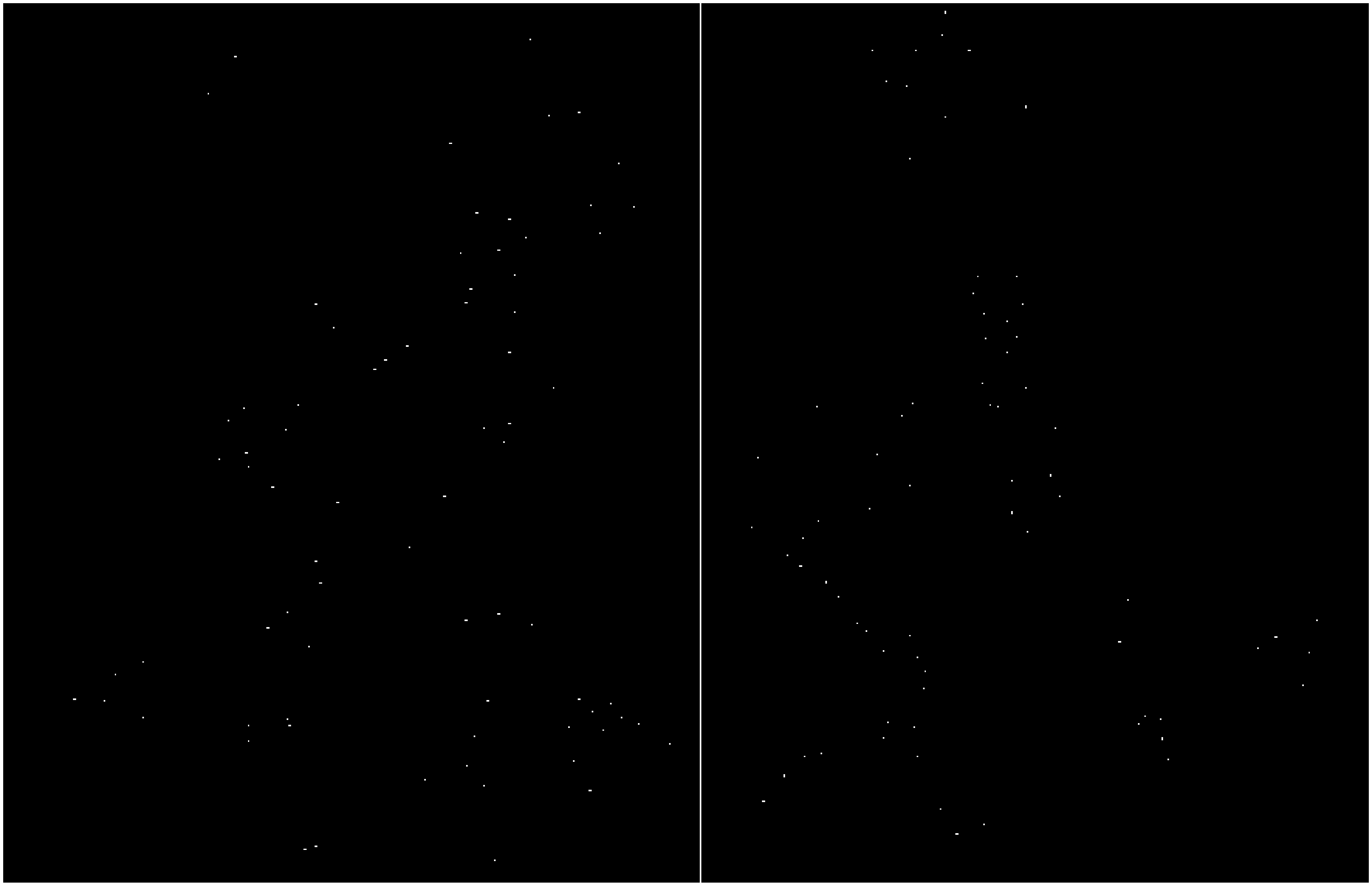
Harris Detector: Workflow

- 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

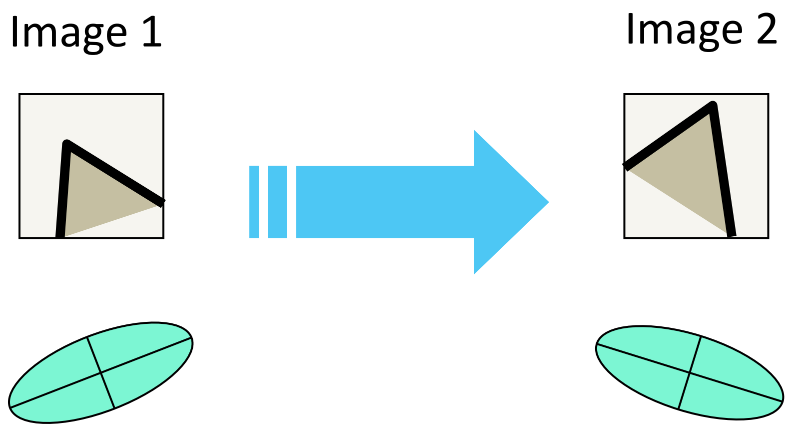
How does the size of the Harris detector affect the performance?

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**

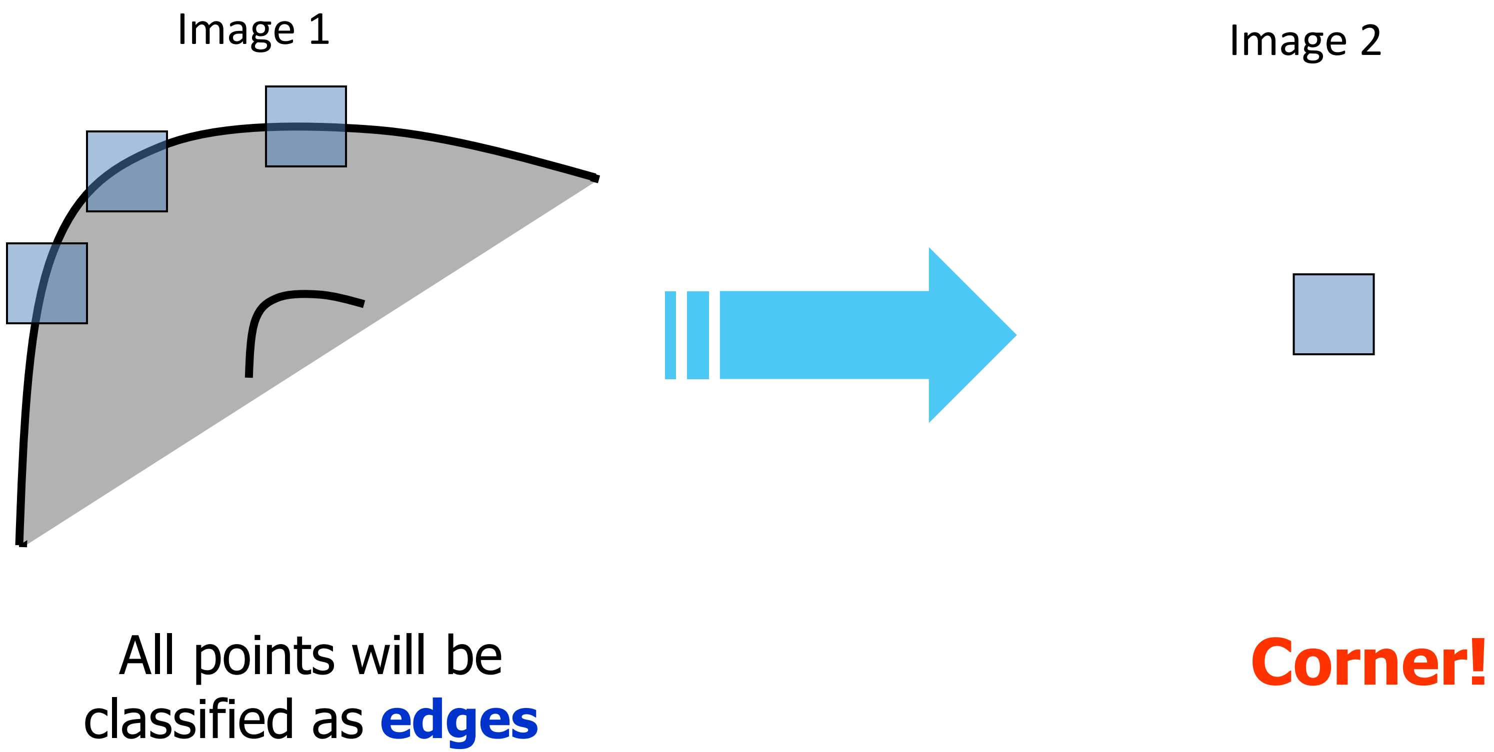


Ellipse rotates but its shape (i.e., eigenvalues) remains the same

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

Harris Detector: Some Properties

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



Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability=

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