



# Vision Algorithms for Mobile Robotics

## Lecture 05 Point Feature Detection and Matching – Part 1

Davide Scaramuzza

<http://rpg.ifi.uzh.ch>

# Lab Exercise 3 - Today

- Implement the Harris corner detection and matching



I will be away on October 28<sup>th</sup> and November 11<sup>h</sup>.  
The lecture will be replaced by a video recording. Q&A session at exercise.

23.09.2021	Lecture 01 - Introduction to Computer Vision and Visual Odometry No Exercise today.
30.09.2021	Lecture 02 - Image Formation: perspective projection and camera models Exercise 01- Augmented reality wireframe cube
07.10.2021	Lecture 03 - Camera Calibration Exercise 02 - PnP problem
14.10.2021	Lecture 04 - Filtering & Edge detection No Exercise today.
21.10.2021	Lecture 05 - Point Feature Detectors, Part 1 Exercise 03 - Harris detector + descriptor + matching
28.10.2021	Lecture 06 - Point Feature Detectors, Part 2 – Scaramuzza away – Video recording Exercise 04 - SIFT detector + descriptor + matching + Q&A session
04.11.2021	Lecture 07 - Multiple-view geometry 1 Exercise 05 - Stereo vision: rectification, epipolar matching, disparity, triangulation
11.11.2021	Lecture 08 - Multiple-view geometry 2 – Scaramuzza away – Video recording Exercise 06 - Eight-Point Algorithm + Q&A session
18.11.2021	Lecture 09 - Multiple-view geometry 3 Exercise 07 - P3P algorithm and RANSAC
25.11.2021	Lecture 10 - Multiple-view geometry 4 Exercise session: Intermediate VO Integration
02.12.2021	Lecture 11 Optical Flow and KLT Tracking Exercise 08 - Lucas-Kanade tracker
09.12.2021	Lecture 12a (1st hour) - Place recognition Lecture 12b (2nd hour) - Dense 3D Reconstruction and Place recognition Lecture 12c (3rd and 4th hour, replaces exercise) - Deep Learning Tutorial Optional Exercise on Place Recognition
16.12.2021	Lecture 13 - Visual inertial fusion Exercise 09 - Bundle Adjustment
23.12.2021	Lecture 14 - Event based vision Exercise session: Final VO Integration

# 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**



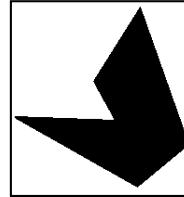
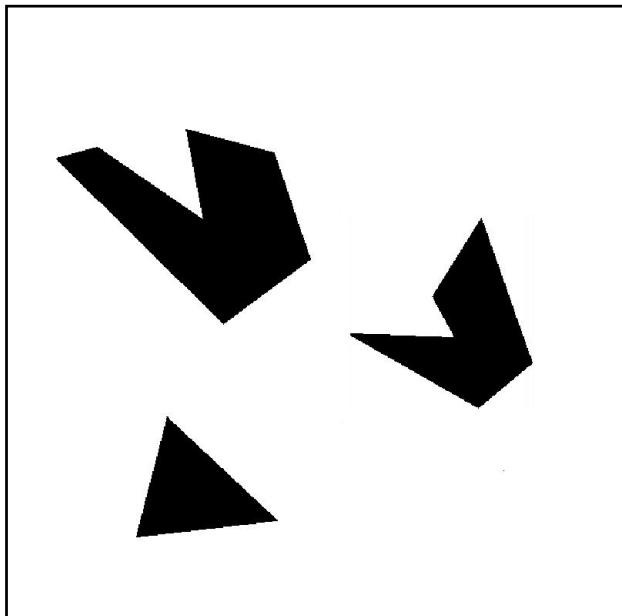
- However, filters can also be used to detect “**features**”  
**Goal: reduce amount of data to process in later stages**, discard redundancy to preserve only what is useful (leads to **lower bandwidth and memory storage**)

- **Edge detection** (we have seen this already; edges can enable line or shape detection)
- **Template matching**
- **Keypoint detection**

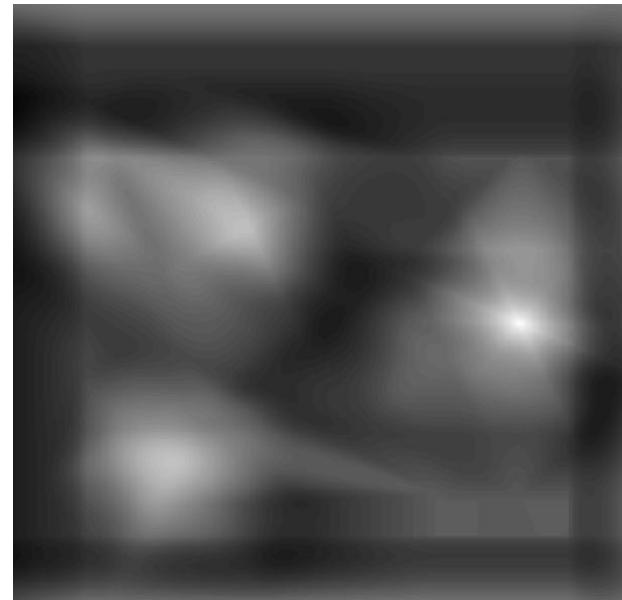


# Filters for Template Matching

- Find locations in an image  $I$  that are similar to a **template**  $H$
- If we look at filters as **templates**, we can use **cross-correlation** (see lecture 4, like convolution but without rotating the filter) to detect these locations



$H$



Correlation map  $I'$ :

$$I'[x, y] = \sum_{u=-k}^k \sum_{v=-k}^k I[x + u, y + v]H[u, v]$$

# Where's Waldo?

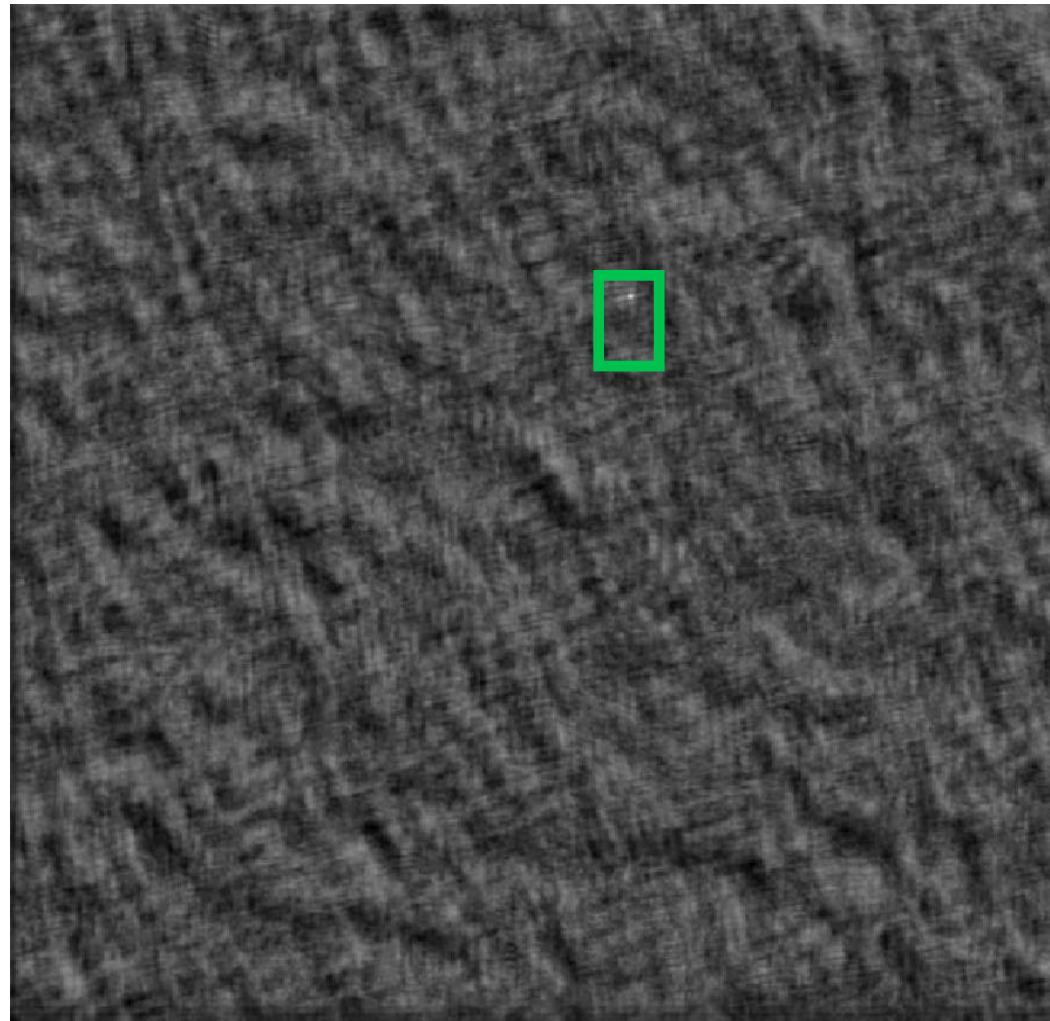


Scene



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?
- What about the pixels in the template's background (object-background problem)?
- Template Matching will only work if **scale**, **orientation**, **illumination**, and, in general, the **appearance of the template** (including anything in background) and the object to detect are **very similar**.



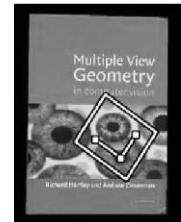
Scene



Template



Scene

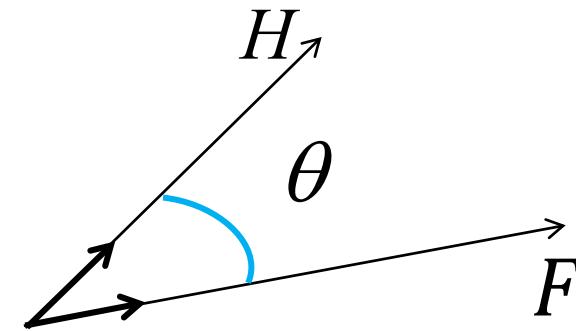


Template

# Correlation as Scalar Product

- Consider two image patches  $H$  and  $F$  of the same size as 1-dimensional vectors with  $n$  entries (where  $n$  is the number of pixels), their cross-correlation can be written as an inner product:

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



- In **Normalized Cross Correlation (NCC)**, we consider the unit vectors of  $H$  and  $F$ , hence we measure their similarity based on the angle  $\theta$ .

$$\cos \theta = \frac{\langle H, F \rangle}{\|H\| \|F\|}$$

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

# Similarity Measures

- **Normalized Cross Correlation (NCC)**: takes values between -1 and +1 (+1 if  $H$  and  $F$  are 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)**

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

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

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

# 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:

$$\mu_H = \frac{1}{n} \sum_{u=-k}^k \sum_{v=-k}^k H(u, v) \quad \mu_F = \frac{1}{n} \sum_{u=-k}^k \sum_{v=-k}^k F(u, v) \quad n \text{ is the number of pixels of H or F}$$

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

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

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

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

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

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

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

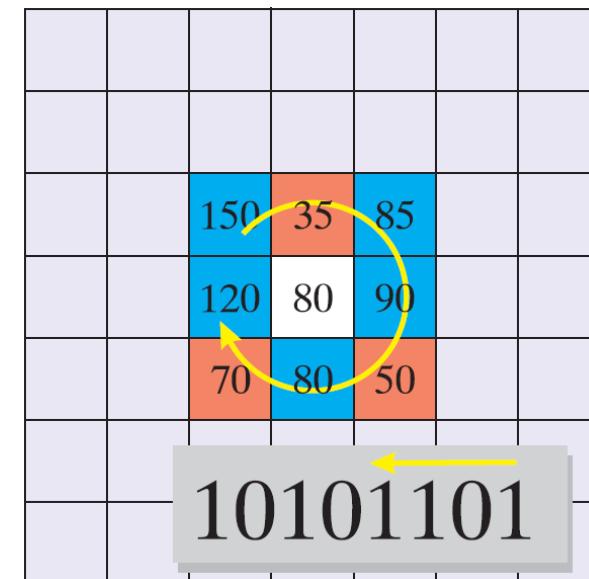
Are these invariant to affine  
illumination changes?

# 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  $w \times w$  patch, 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 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**

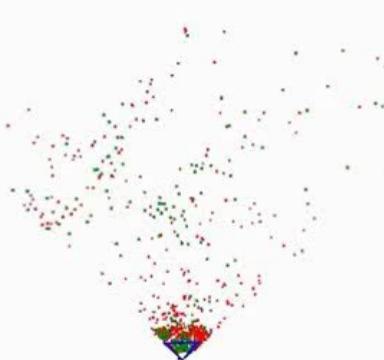


# Outline

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

# Keypoint extraction and matching - Example

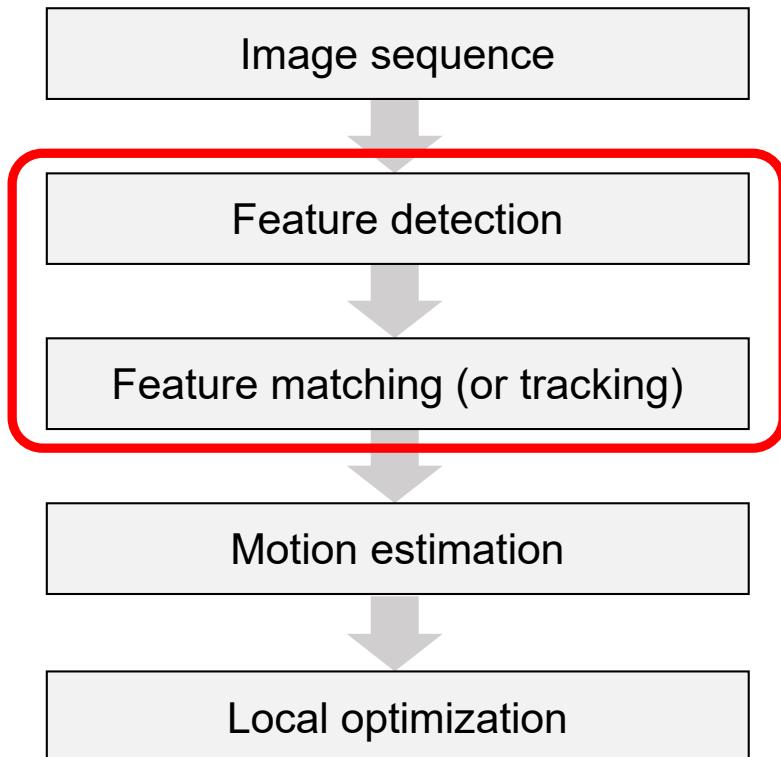
**SVO with a single camera** on Euroc dataset



Video from “Forster, Pizzoli, Scaramuzza, *SVO: Semi-Direct Visual Odometry*”. IEEE Transactions on Robotics, 2017. [PDF](#). [Video](#)

# Why do we need keypoints?

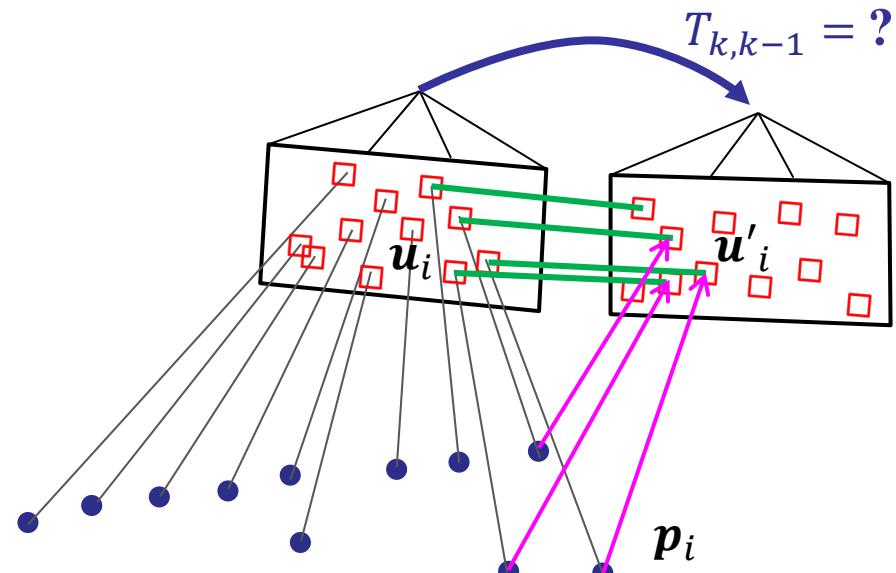
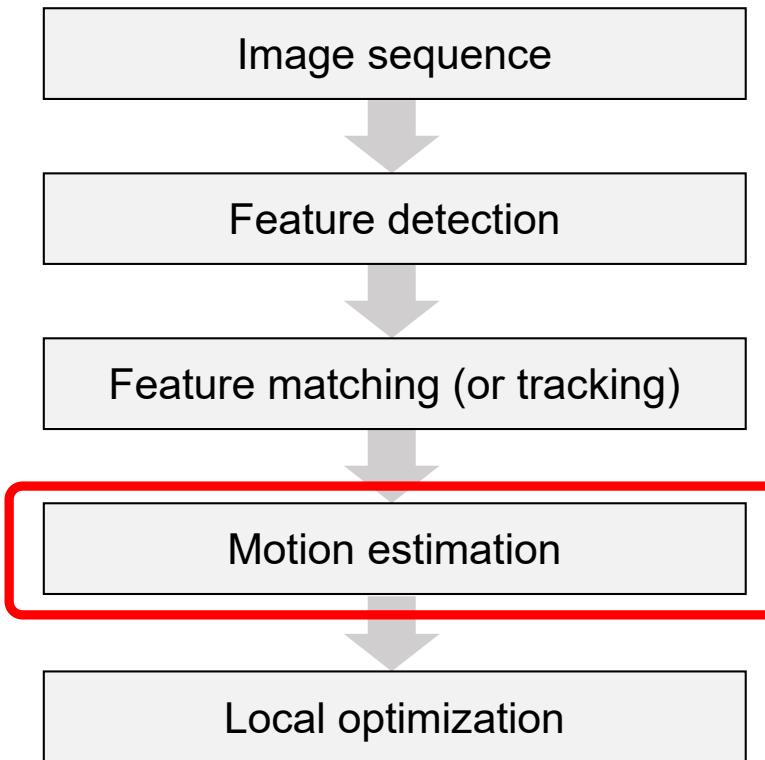
- Recall the Visual-Odometry flow chart:



Features tracked over multiple recent frames  
overlaid on the last frame

# Why do we need keypoints?

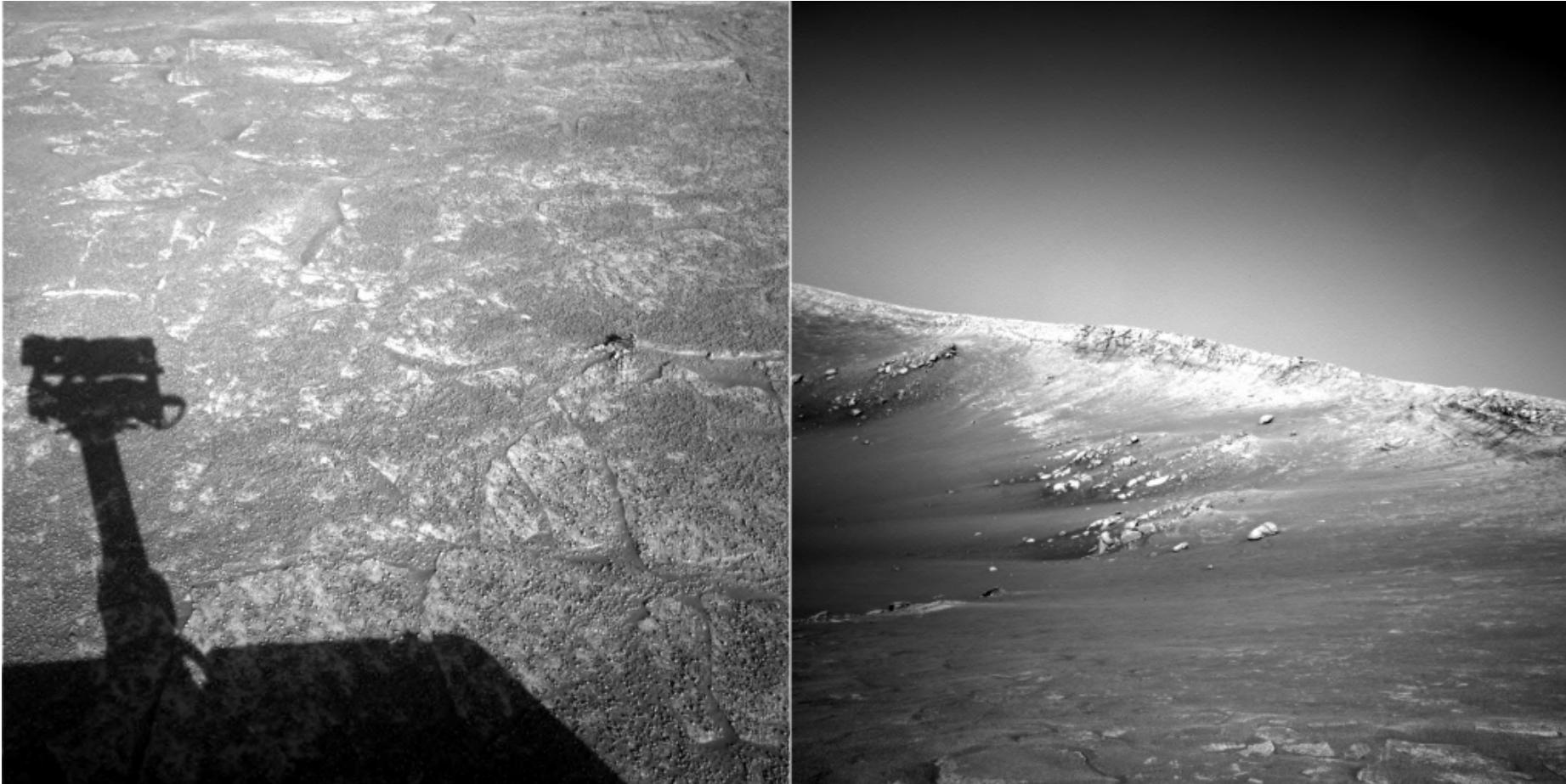
- Keypoint extraction is the key ingredient of motion estimation!



# 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 **Feature Matching problem**: finding similar keypoints between two images of the same scene taken under different conditions

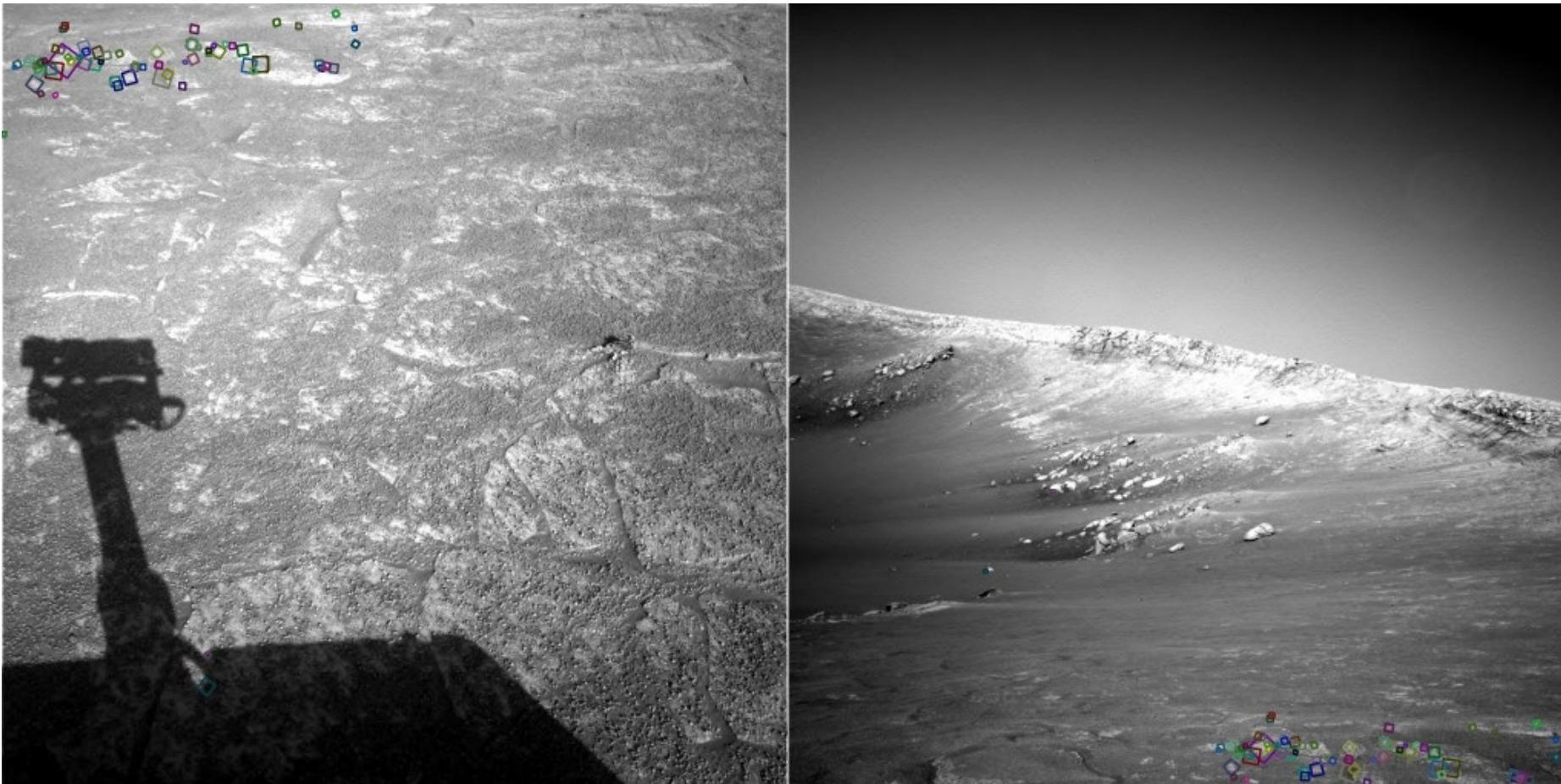
# Image matching: why is it challenging?



NASA Mars Rover images

# Image matching: why is it challenging?

- Answer below



NASA Mars Rover images with SIFT feature matches

# Example: panorama stitching



## How does it work?

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

M. Brown and D. G. Lowe. Recognising Panoramas, International Conference on Computer Vision (ICCV), 2003. [PDF](#).

# Local features and alignment

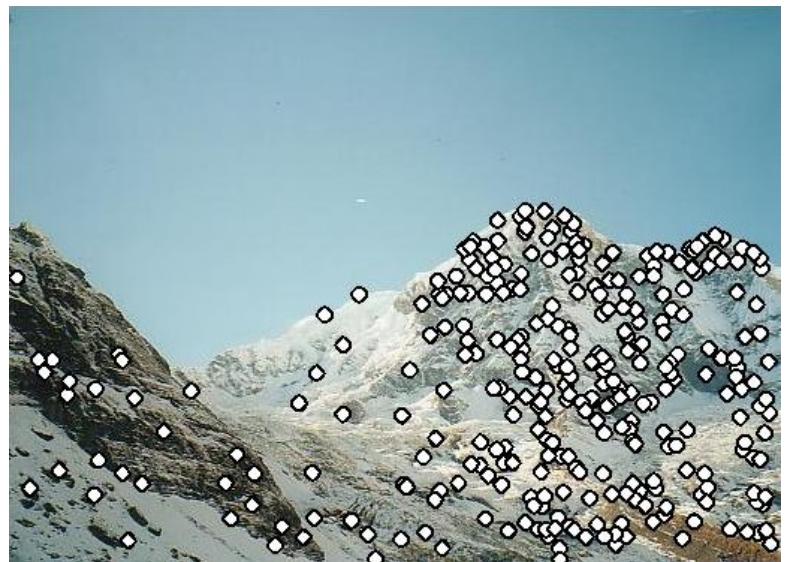
- We need to align two images
- How would you do it?



# Local features and alignment

Idea:

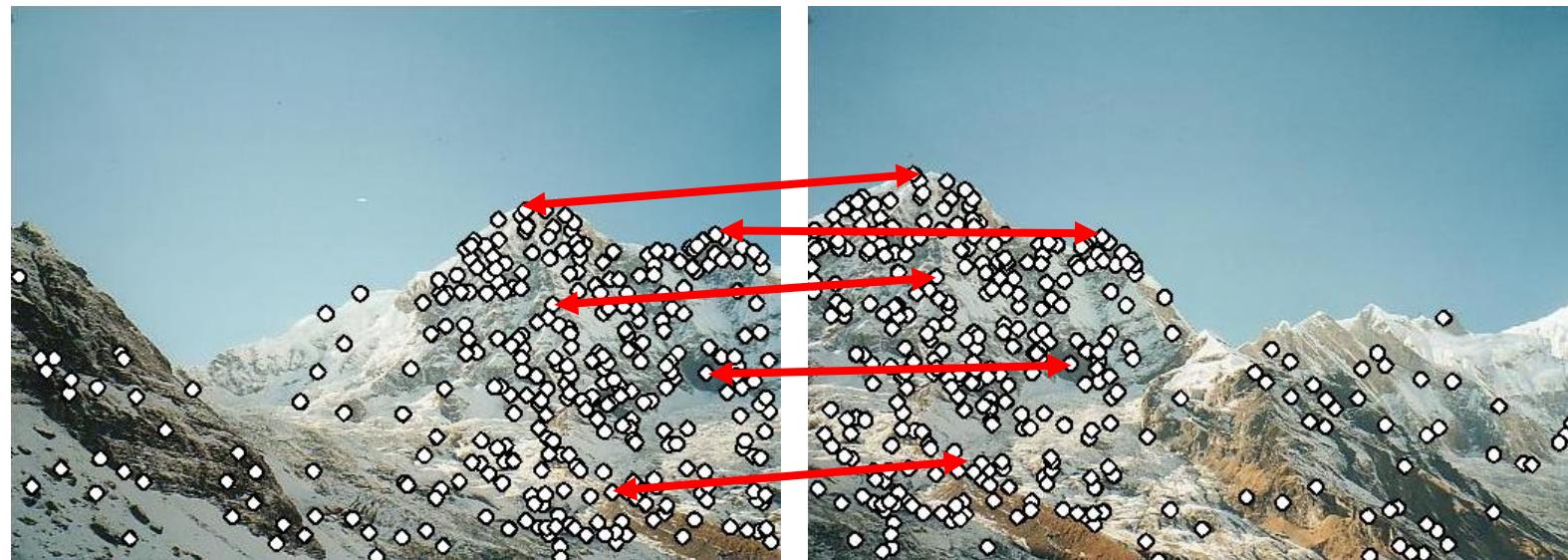
- Detect point features in both images



# Local features and alignment

Idea:

- Detect point features in both images
- Find corresponding pairs



# Local features and alignment

Idea:

- Detect point features in both images
- Find corresponding pairs
- Use these pairs to align the images: **what image transformation would you use?**



# Matching with Features

**Problem 1:** How to detect the same points **independently** in both images?



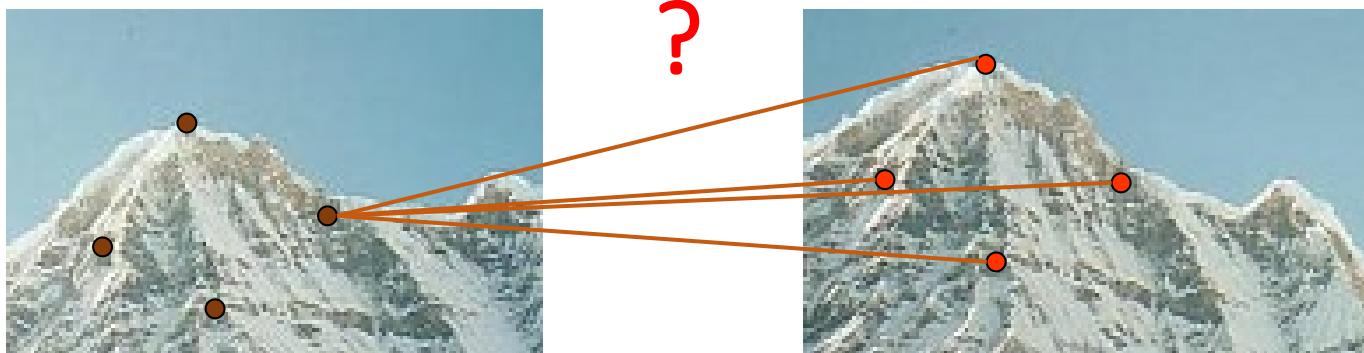
no chance to match!

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, how to **match** its **corresponding point** in the other image

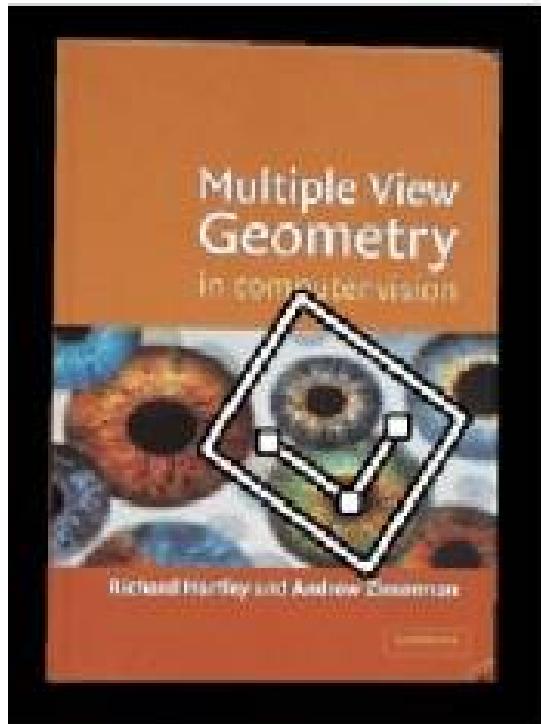


We need a **distinctive** feature descriptor. A descriptor is a “description” of the pixel information around a feature (e.g., patch intensity values, gradient values, etc.). Distinctive means that the descriptor uniquely identifies a feature from other features without ambiguity. This property is called **Distinctiveness** of a feature **descriptor**.

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

# Geometric changes

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



# Photometric Changes (i.e., Illumination changes)

- **Small illumination changes** are modelled with an **affine transformation** (so called *affine illumination changes*):

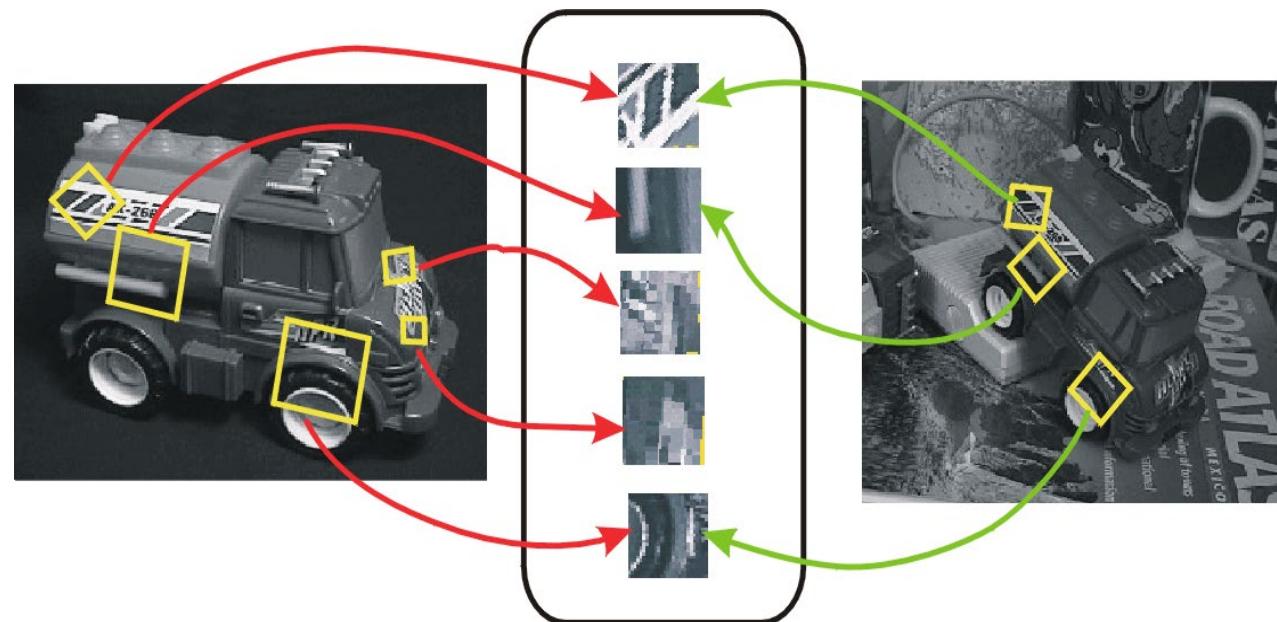
$$I'(x, y) = \alpha I(x, y) + \beta$$



# Local Invariant Features

The key to feature detection and matching is to find **repeatable features** and **distinctive descriptors** that are **invariant** to 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 images below with some patches. Notice how some patches can be localized or matched with higher accuracy than others

Image 1

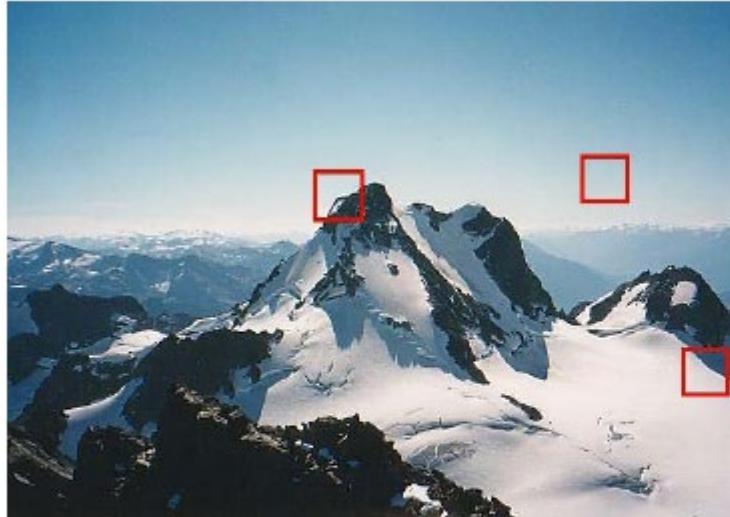
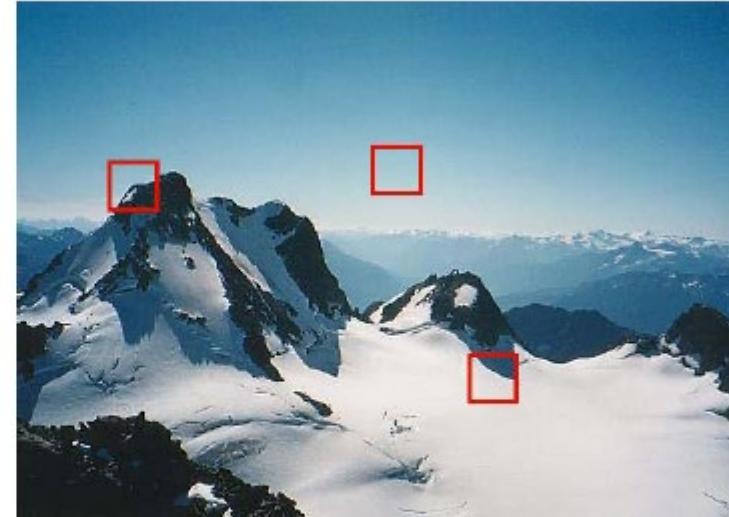


Image 2

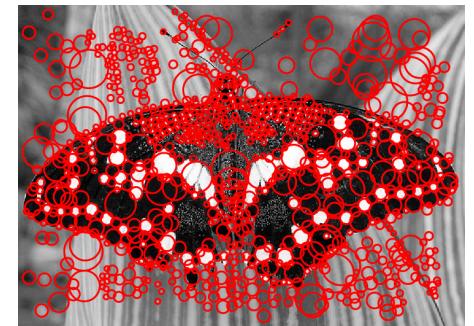


# Point Features: Corners vs Blob detectors

- A **corner** is defined as the intersection of two or more edges
  - Corners 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 corners** → blobs are better for place recognition
  - E.g., *MSER, LOG, DOG (SIFT), SURF, CenSurE, etc.*

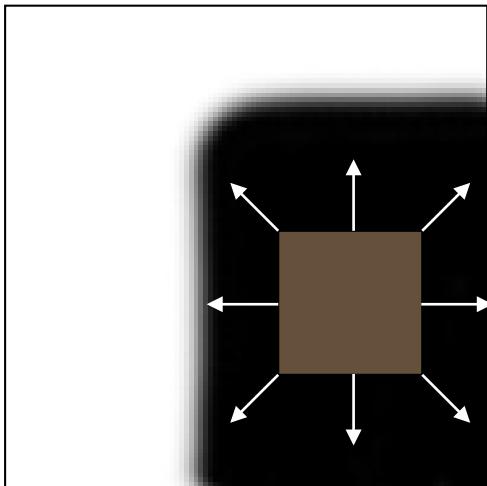


# Corner Detection

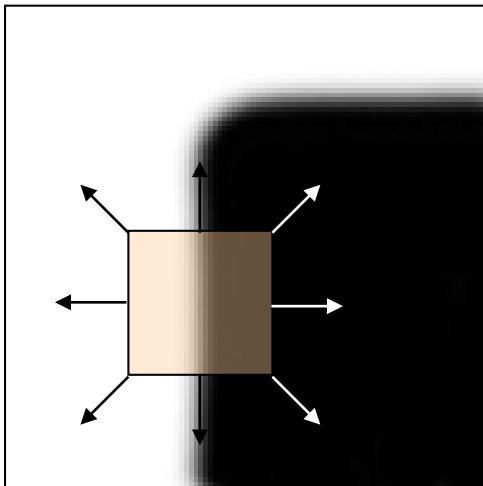
- Key observation: in the region around a corner, image gradient has **two or more** dominant directions
- Corners are **repeatable** and **distinctive**

# The Moravec Corner detector (1980)

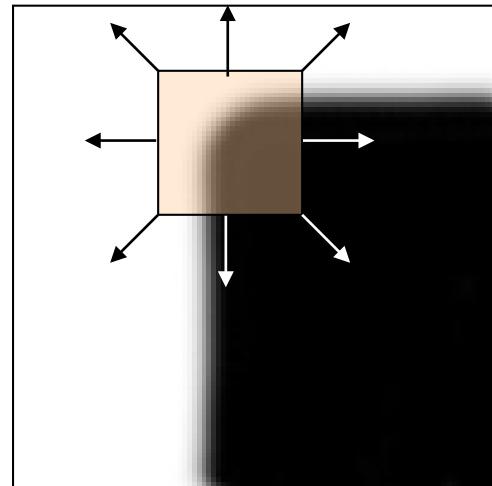
- How do we identify corners? Look at a region of pixels through a small window
- Shifting a window in **any direction** should give raise to **large intensity changes** (e.g., in SSD)



“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)

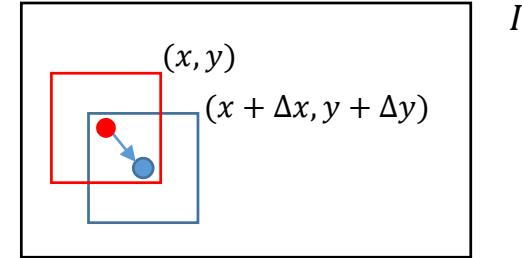


“corner”:  
significant change in all directions  
(i.e.,  $SSD \gg 0$  in all directions)

# 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  $\Omega$ . The Sum of Squared Differences between them is:

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

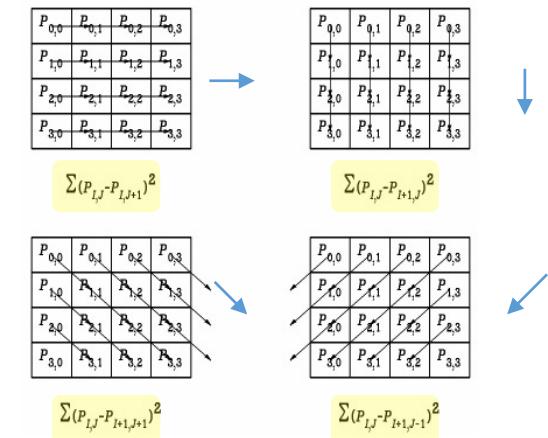


*“Sums of squares of differences of pixels adjacent in each of four directions (horizontal, vertical and two diagonals) over each window are calculated, and the window's interest measure is the minimum of these four sums. Features are chosen where the interest measure has local maxima.”* [Moravec'80,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

The disadvantage of the Moravec corner detector is that we need to compute four SSDs, one for each shifted version of the patch (1 pixel right, down, down right, and down left).

Can we make it more efficient?

Can we do it without shifting the patch at all?



# 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.



# 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  $\Omega$ . The Sum of Squared Differences between them is:

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

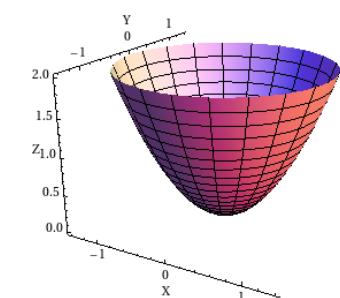
- Let  $I_x = \frac{\partial I(x, y)}{\partial x}$  and  $I_y = \frac{\partial I(x, y)}{\partial y}$ . Approximating with a 1<sup>st</sup> order Taylor expansion:

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

$$\Rightarrow SSD(\Delta x, \Delta y) \approx \sum_{x,y \in \Omega} (I_x(x, y)\Delta x + I_y(x, y)\Delta y)^2$$

- This is a **quadratic function** in two variables  $(\Delta x, \Delta y)$  (i.e., a paraboloid).

How can the shape of this paraboloid reveal whether the patch is corner, an edge or a constant region?



# How do we implement this?

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

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

- This can be written in a matrix form as:

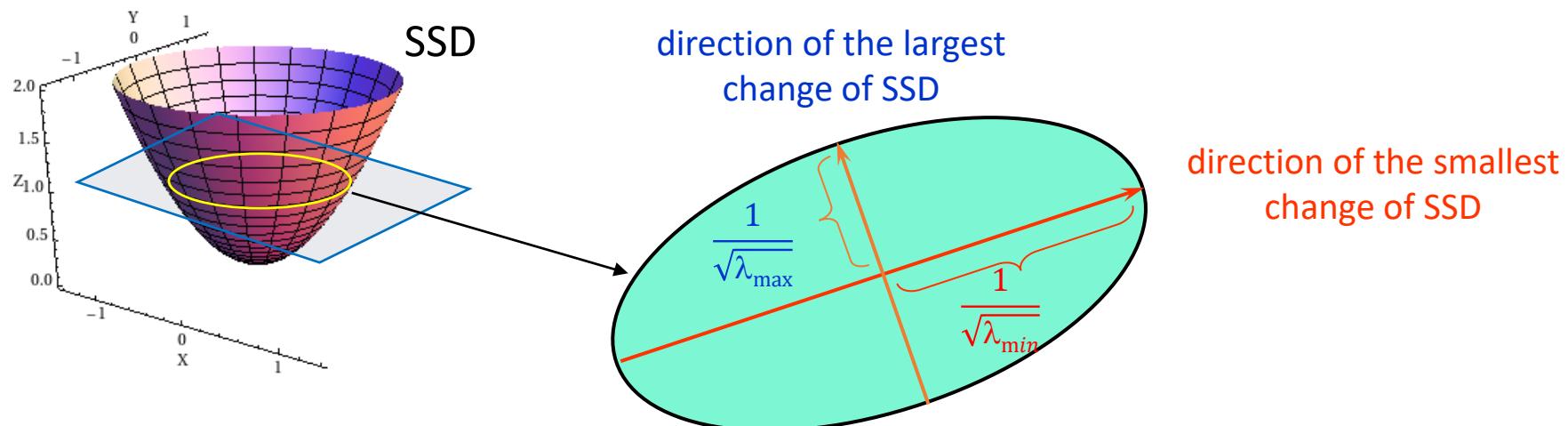
$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in \Omega} [\Delta x \quad \Delta y] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = [\Delta x \quad \Delta y] \underbrace{\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}}_M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$



M  
2nd moment matrix

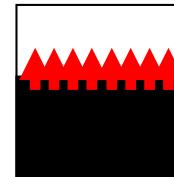
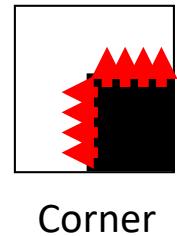
# What does this matrix reveal?

- 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  $[\Delta x \quad \Delta y] M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \text{const}$  as an ellipse with **axis lengths** determined by the **eigenvalues** and the **orientation** 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**
- In presence of noise, we can conclude that if one **eigenvalue is much larger than the other** then it is an **edge**. If they are **both small**, then it is a **flat region**.
- Now, let's consider an axis-aligned corner:



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

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

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 directions of largest and smallest SSD change are at 45 degrees with  $x$  and  $y$  axes
- We can also conclude that if **both eigenvalues are much larger than 0**, then **this is a corner**

# Review: How to compute $\lambda_1, \lambda_2, R$ from $M$ Eigenvalue/eigenvector

- You can easily prove that  $\lambda_1, \lambda_2$  are the **eigenvalues** of  $M$ .
- The **eigenvectors** and **eigenvalues** of a square matrix  $A$  are the vectors  $x$  and scalars  $\lambda$  that satisfy:

$$Ax = \lambda x$$

- The scalar  $\lambda$  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:  $\det \begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix} = 0$

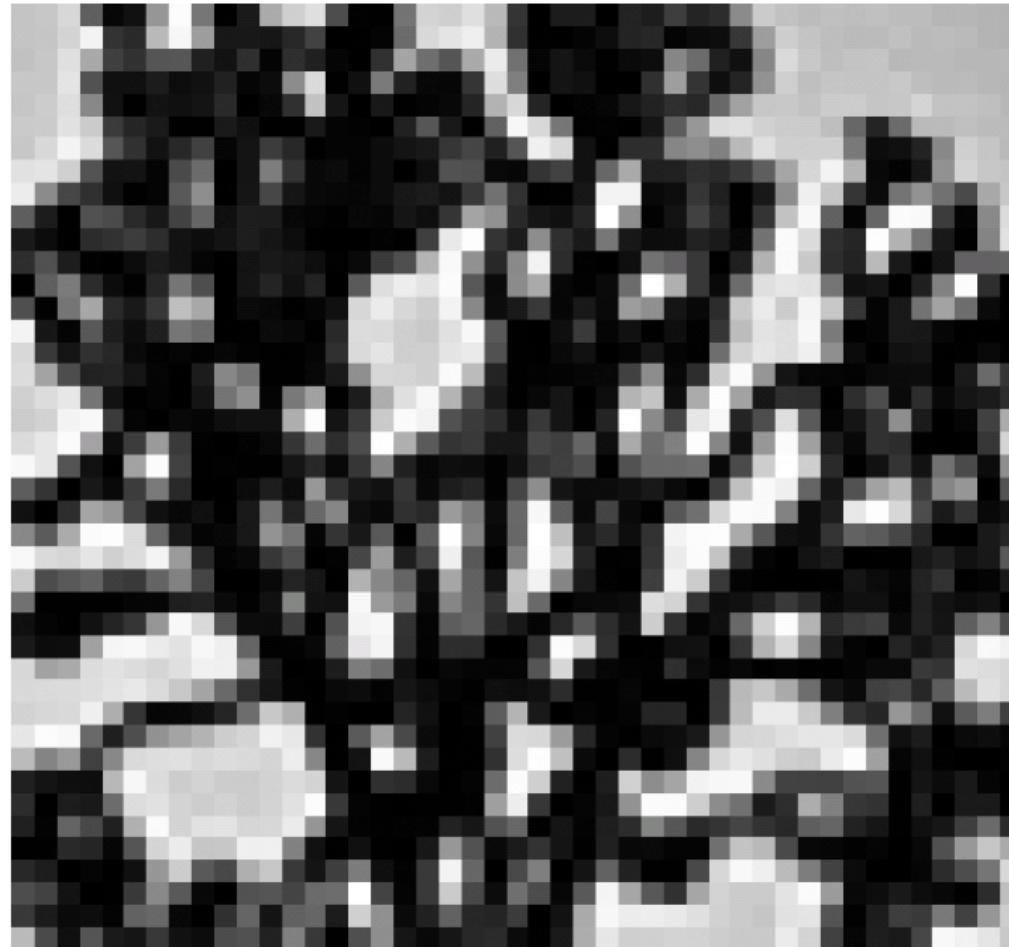
- The solution is:

$$\lambda_{1,2} = \frac{1}{2} \left[ (m_{11} + m_{22}) \pm \sqrt{4m_{12}m_{21} + (m_{11} - m_{22})^2} \right]$$

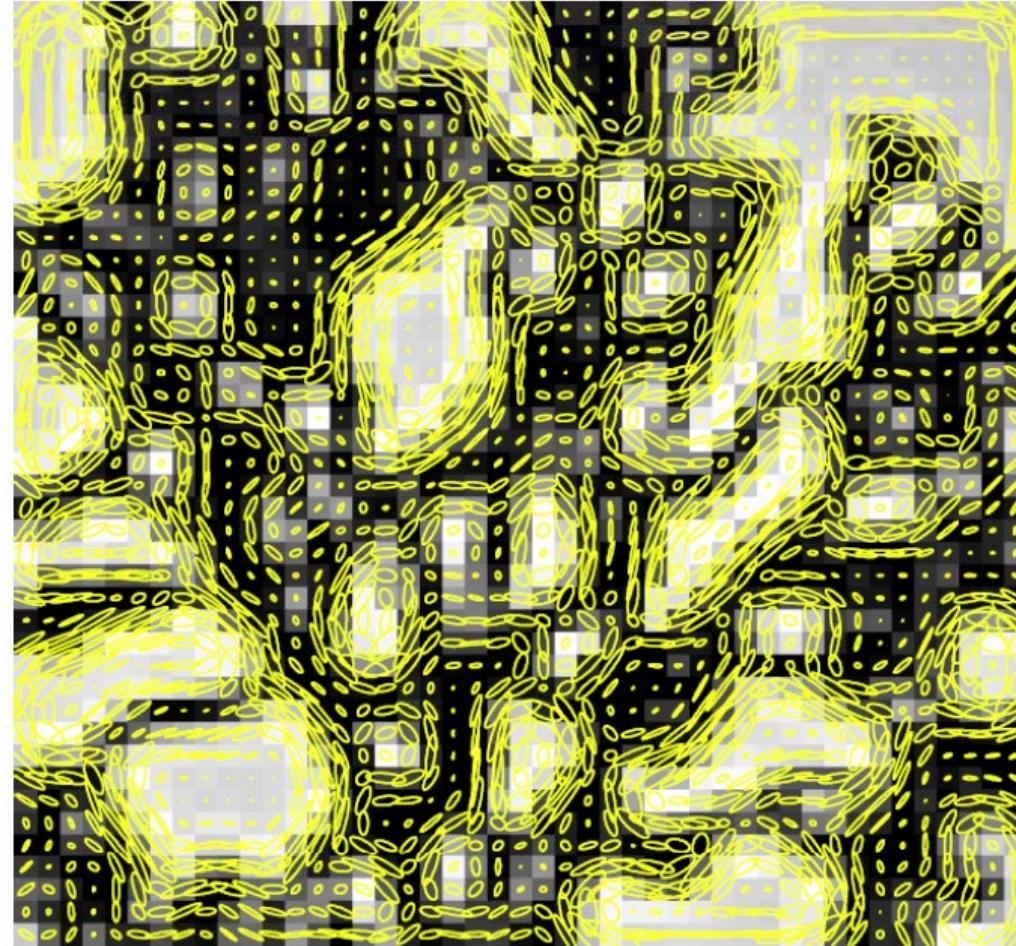
- Once you know  $\lambda$ , you find the two eigenvectors  $x$  (i.e., the two columns of  $R$ ) by solving:

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

# Visualization of 2<sup>nd</sup> moment matrices



# Visualization of 2<sup>nd</sup> moment matrices

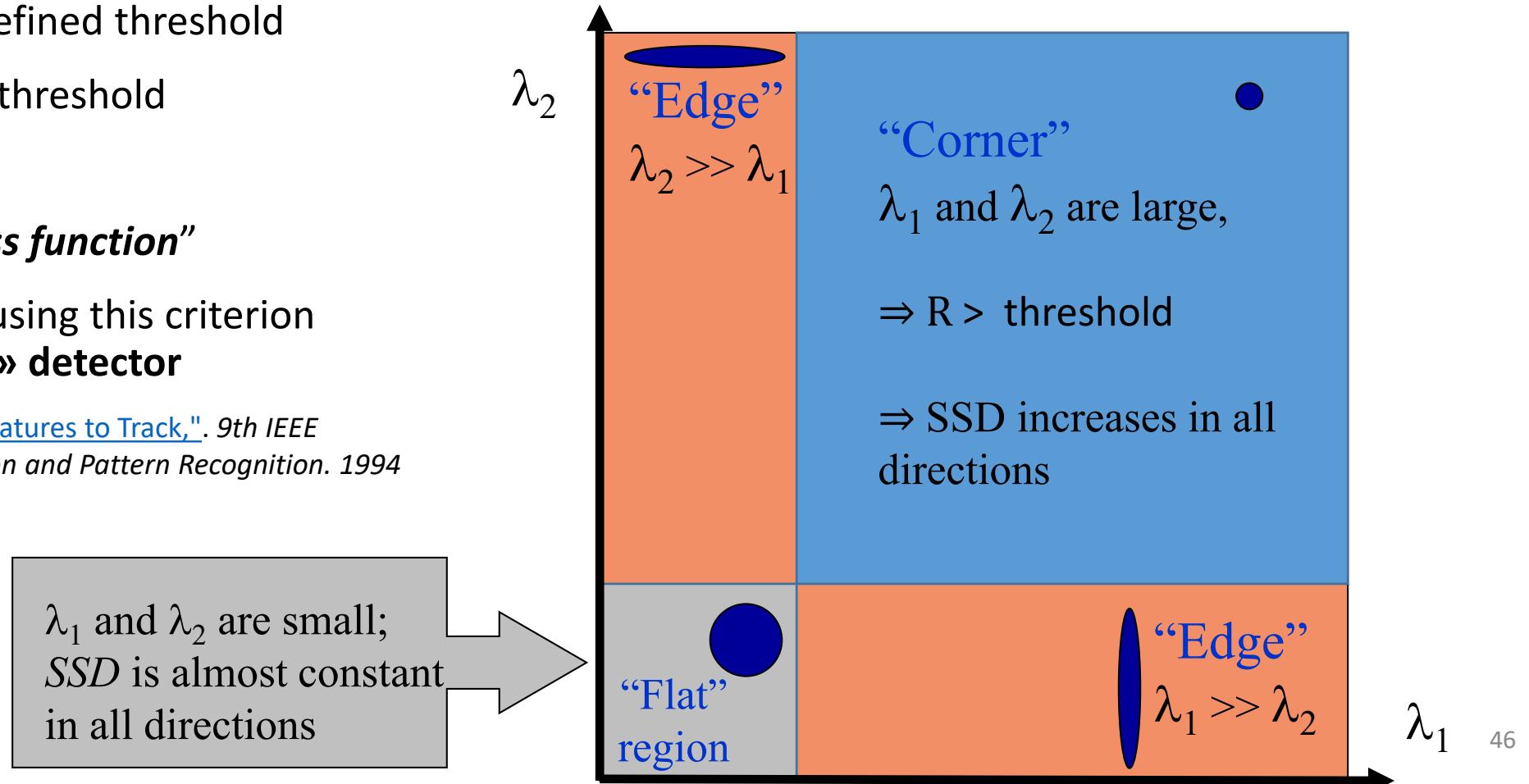


NB: here the ellipses here are plotted proportionally to the eigenvalues and not as iso-SSD ellipses as explained before. So small ellipses here denote a flat region, and big ones, a corner.

# 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  
 $\Rightarrow R = \min(\lambda_1, \lambda_2) > \text{threshold}$
- $R$  is called “***cornerness function***”
- The corner detector using this criterion is called «**Shi-Tomasi**» detector

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



# Interpreting the eigenvalues

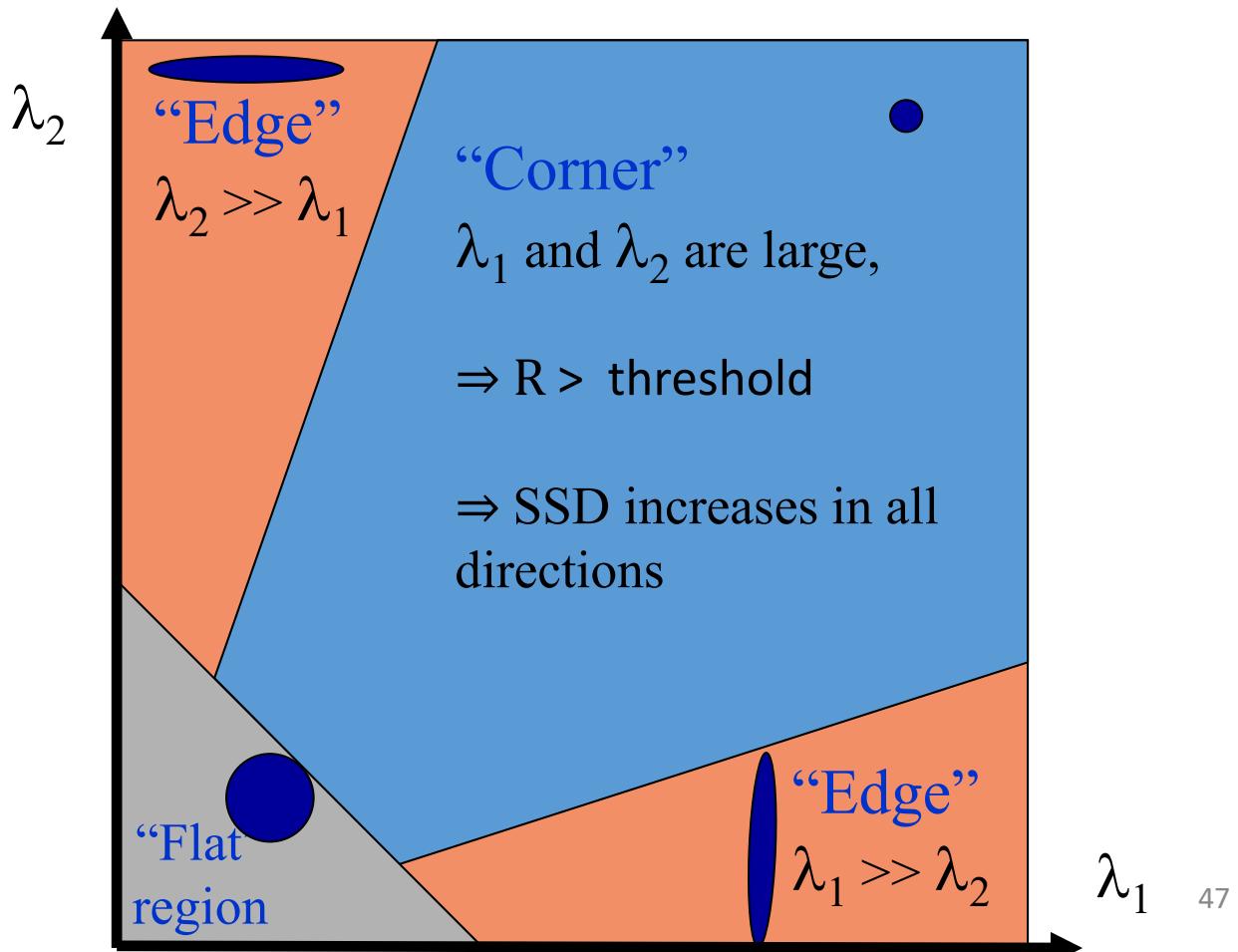
- Computation of  $\lambda_1$  and  $\lambda_2$  is expensive  $\Rightarrow$  Harris & Stephens suggested using a **different cornerness function**:

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

$k$  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"](#),  
*Proceedings of the 4th Alvey Vision Conference*, 1988.

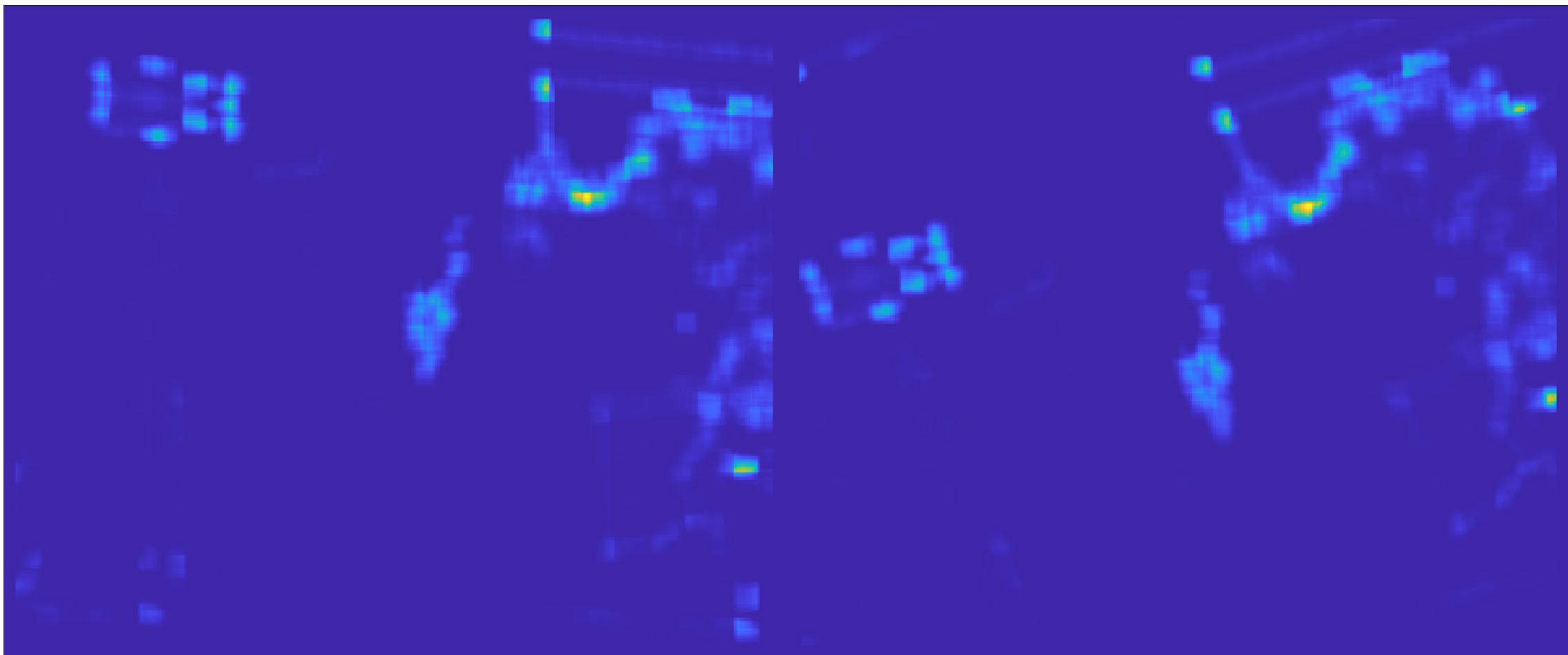


# Harris Detector: Workflow



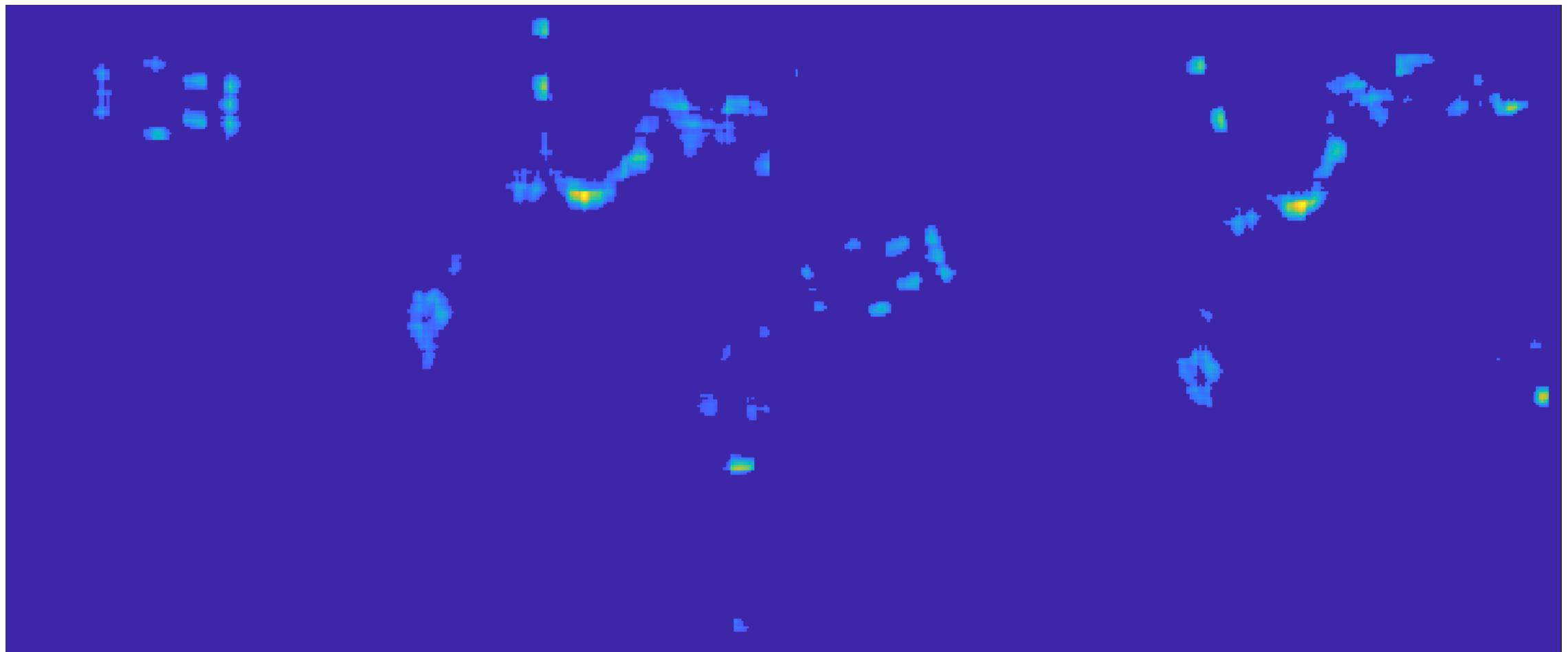
# Harris Detector: Workflow

- Compute corner response  $R$



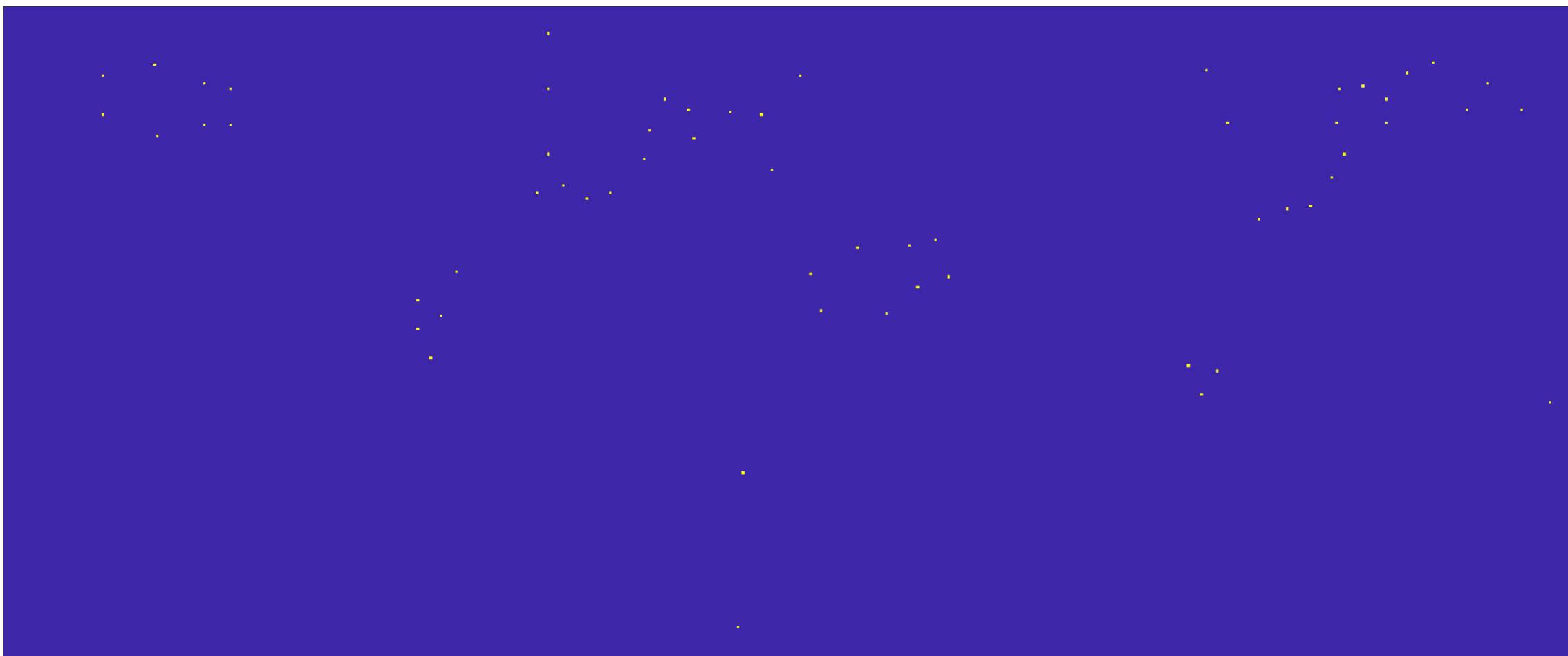
# Harris Detector: Workflow

- **Thresholding:** Find points with large corner response:  $R > \text{threshold}$



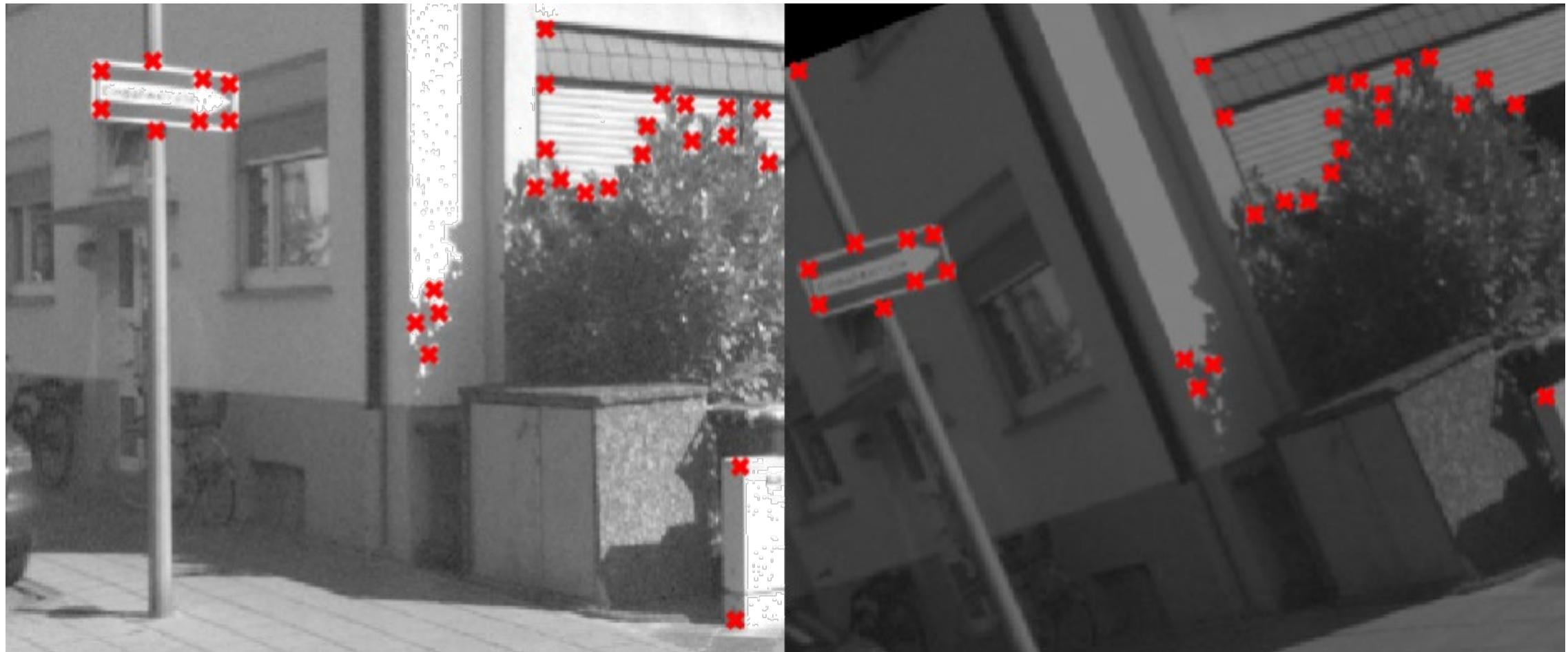
# Harris Detector: Workflow

- **Non-Maxima Suppression:** detect local maxima of thresholded  $R$



# Harris Detector: Workflow

What parameters can we tune to detect more or fewer corners?



# 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  $I_x^2, I_y^2, I_x I_y$  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 ( $R >$  threshold)
6. Take the points of local maxima of  $R$

From now on, whenever we talk about the 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 detector, despite being a bit more expensive, yet has a small advantage... see next slide

# Harris vs. Shi-Tomasi

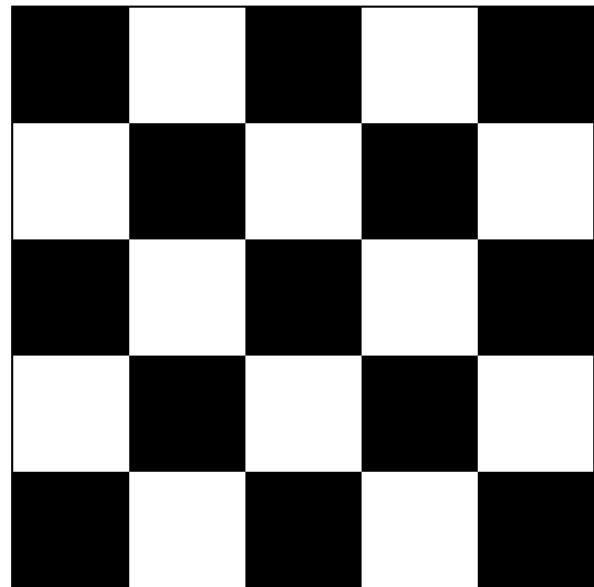
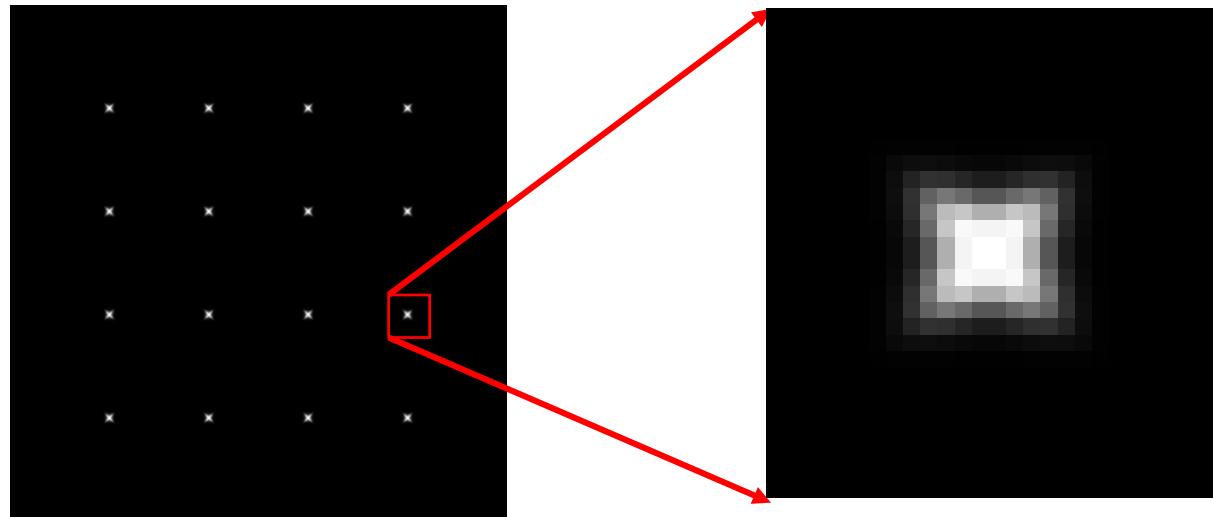
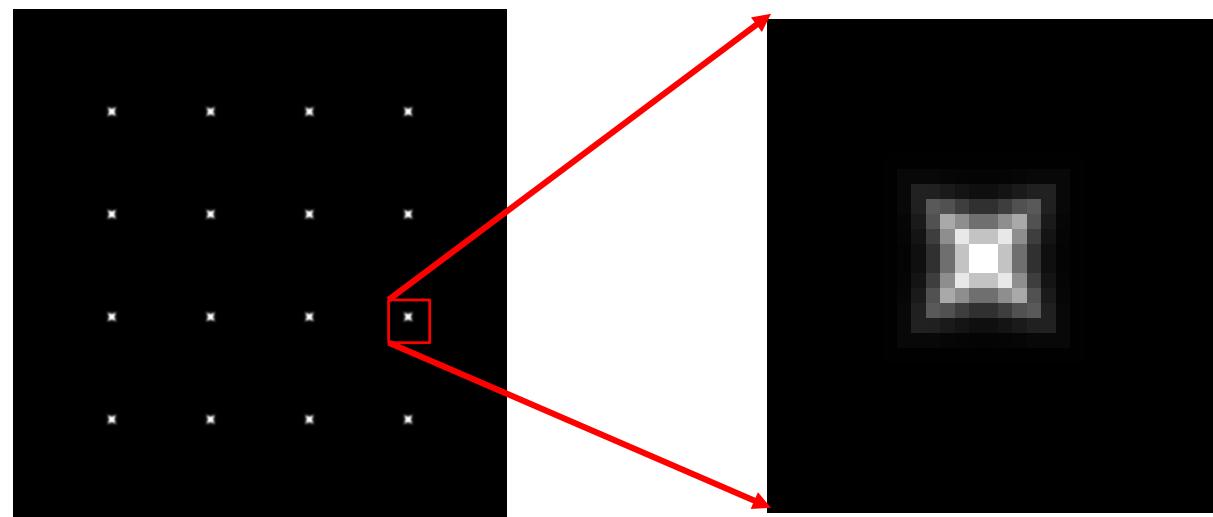


Image  $I$



Harris'cornerness  
response



Shi-Tomasi's  
cornerness response

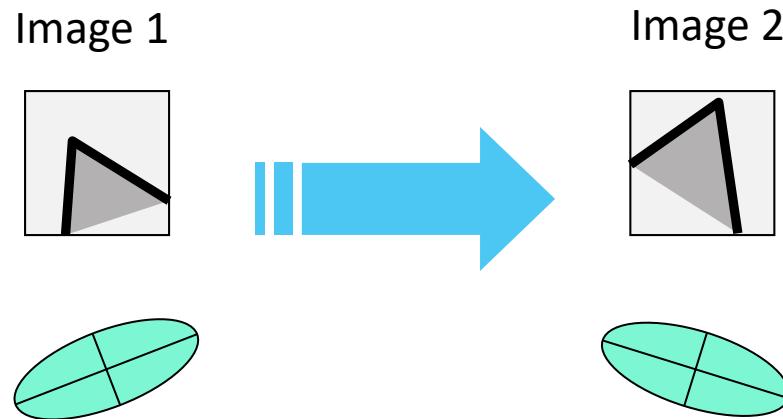
# Harris Detector: Some Properties

## Repeatability:

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

# Harris Detector: Some Properties

- The Harris detector is **rotation invariant**



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

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

# Harris Detector: Some Properties

- The Harris detector is **not scale invariant**

Image 1

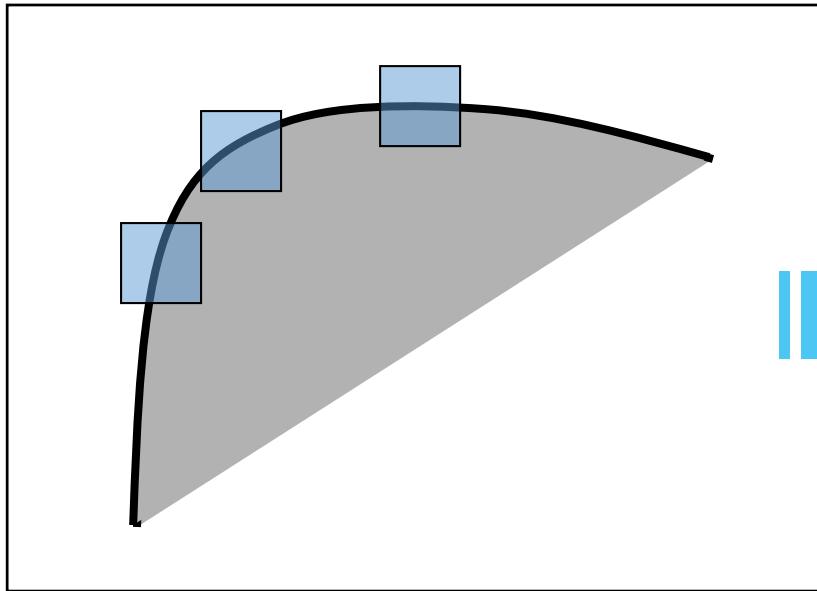
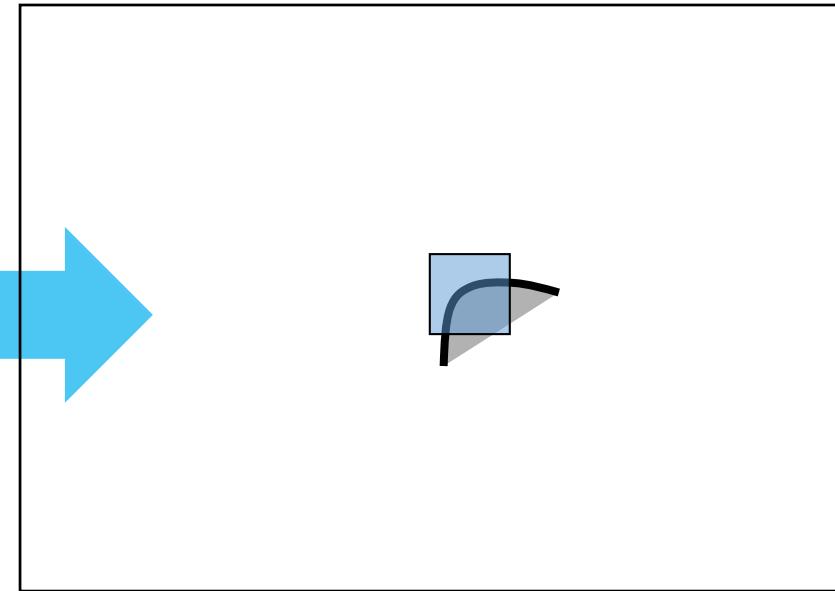


Image 2



All points will be classified as **edges**

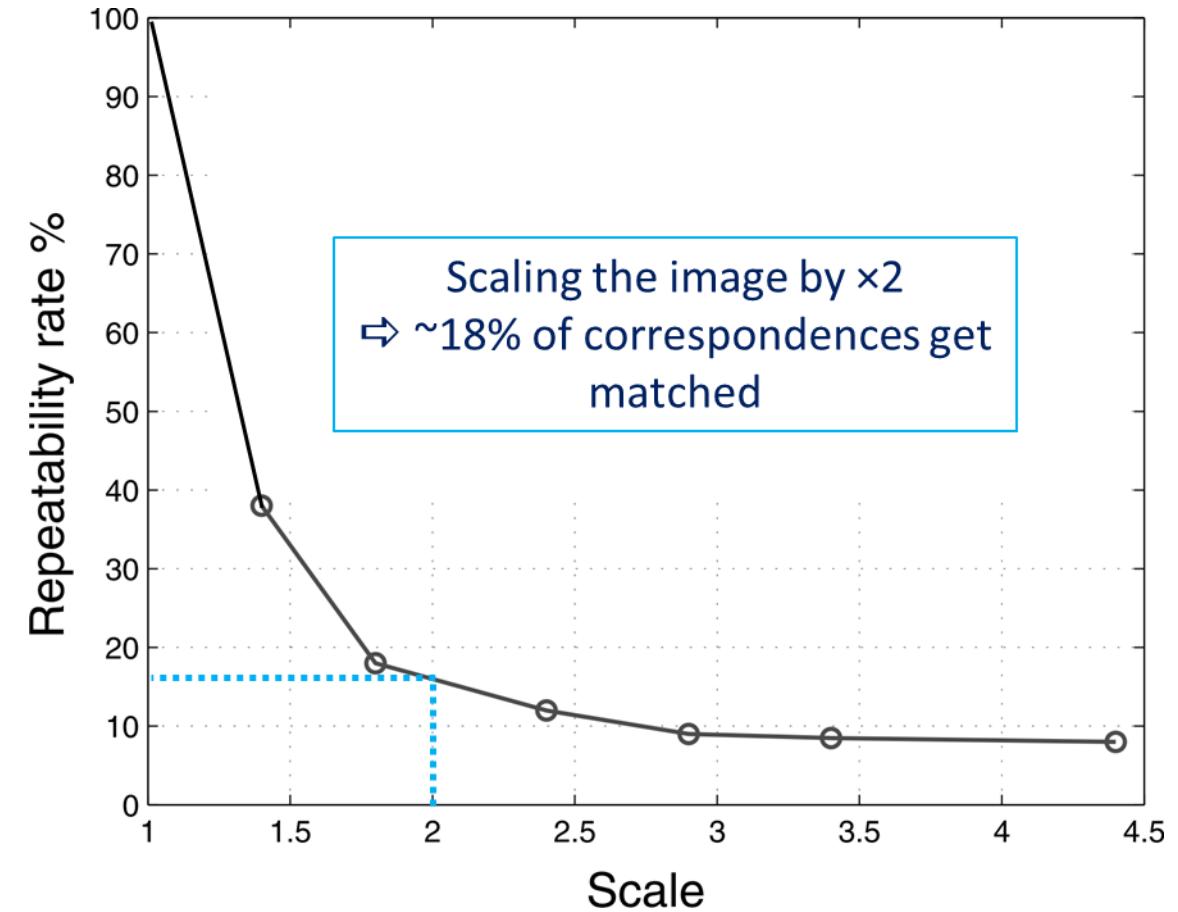
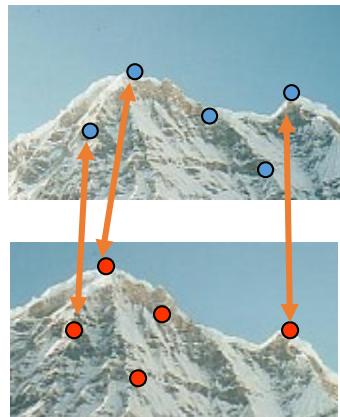
**Corner!**

# Harris Detector: Some Properties

- **Repeatability** of the Harris detector for different scale changes

**Repeatability=**

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



# Harris Detector: Some Properties

- Is it invariant to:
  - Affine illumination changes?
    - yes, why?
  - 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
      - Invariance to rotation, scale, illumination changes

# Readings

- Ch. 7.1 and Ch. 9.6 of Szeliski book, 2<sup>nd</sup> Edition
- Chapter 4 of Autonomous Mobile Robots book: [link](#)
- 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 measures: 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? How is their localization accuracy?
- 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 a paraboloid and 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 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?