ACV – Applied Computer Vision

Bachelor Medientechnik & Creative Computing

Matthias Zeppelzauer
matthias.zeppelzauer@fhstp.ac.at

Djordje Slijepcevic
djordje.slijepcevic@fhstp.ac.at

Roadmap

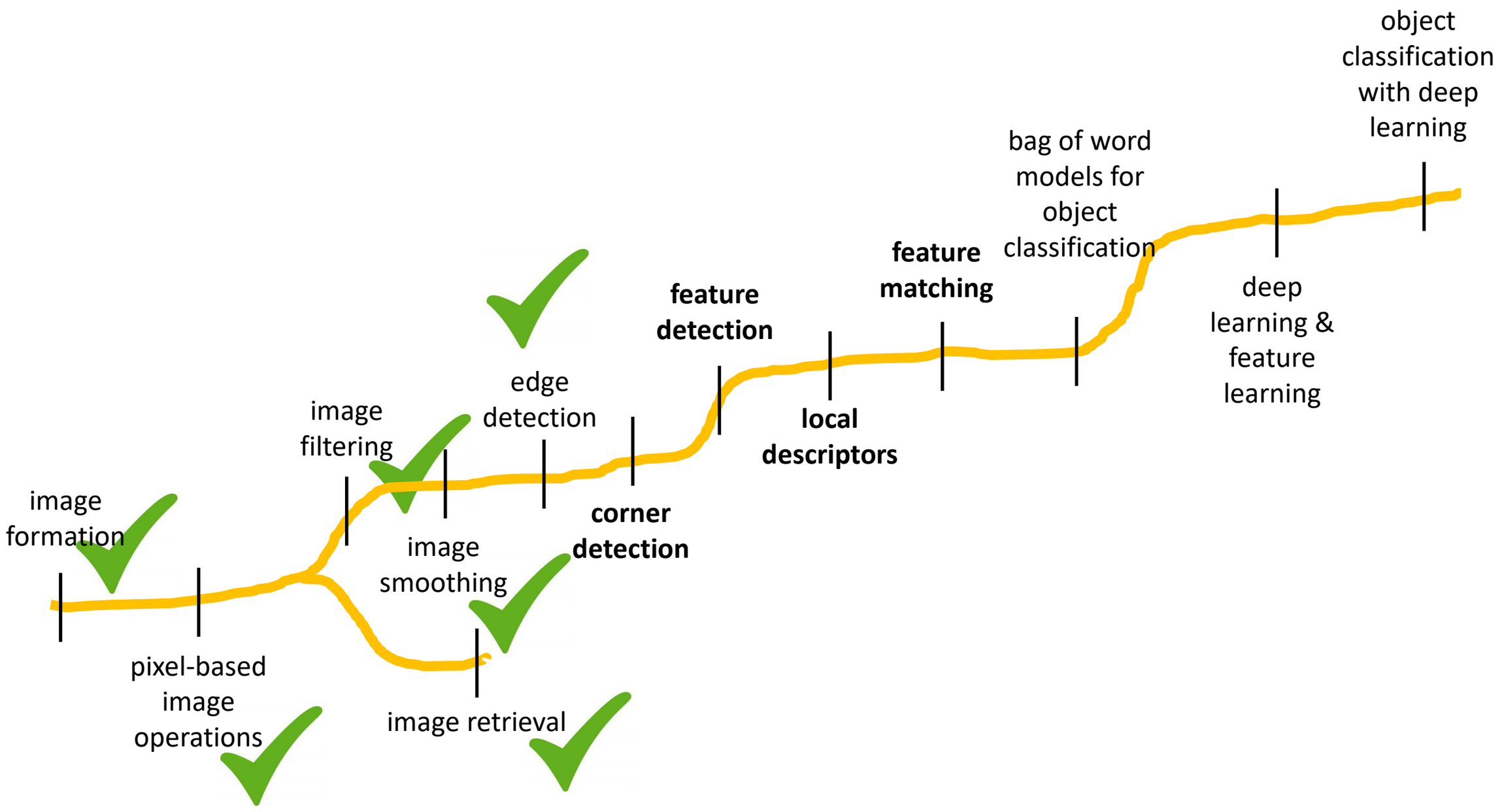


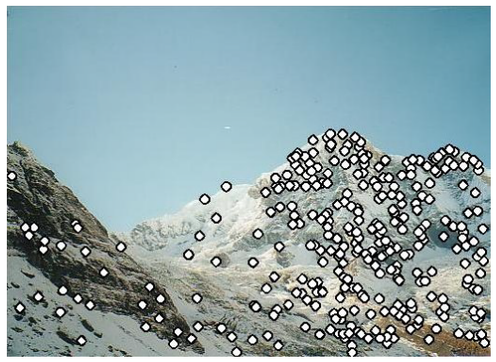
image formation
pixel-based image operations
image filtering
image smoothing
edge detection
corner detection
feature detection
local descriptors
feature matching
bag of word models for object classification
image retrieval
deep learning & feature learning
object classification with deep learning

Outlook

- So far:
  - Pixel-based operations on images
  - Filter (smooting, sharpening)
  - Edge detection
- Today:
  - Corner detection
  - Local features and descriptors
  - Feature matching

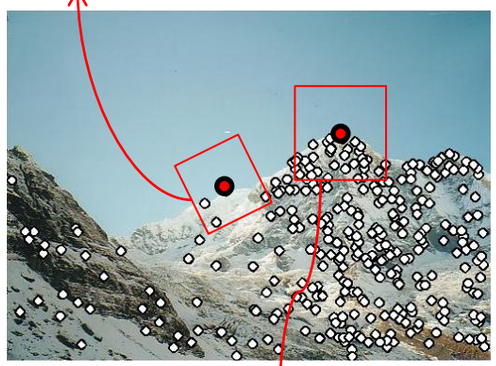
Overview – The Big Picture

- 1. **Detection**: identify the interest points



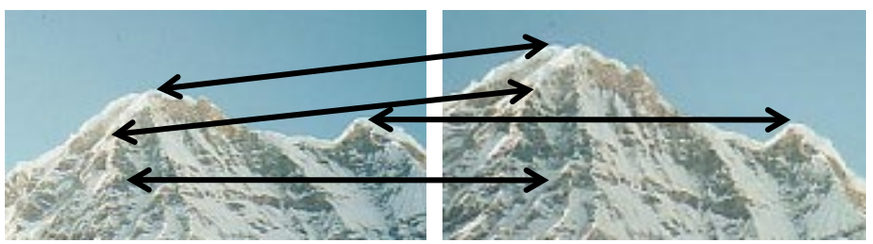
- 2. **Description**: extract a vector (feature descriptor) that represents the surrounding of each feature point

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

- 3. **Matching**: determine correspondence between descriptors in two views



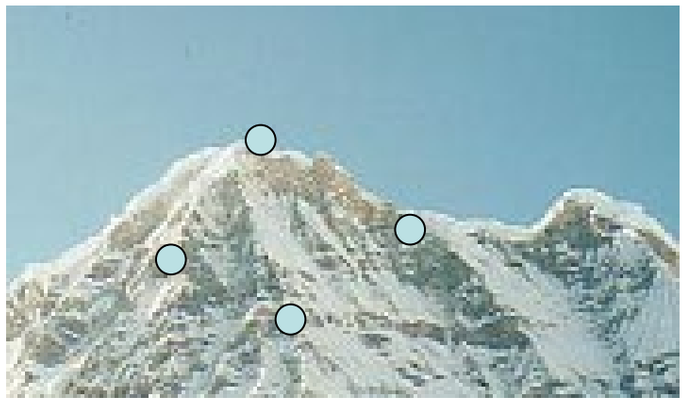
Source: Antonio Torralba

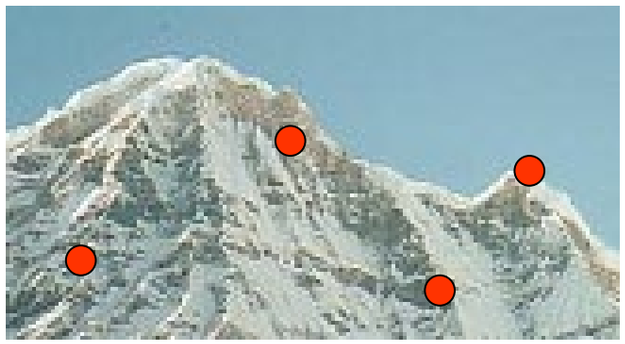
Let’s start with feature detection…

- Note: Local features are often also referred to as:
  - Keypoints
  - Feature points
  - Interest Points
  - Salient Points
  - or simply “features”

Finding reliable local features

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





no chance to match!

We need a repeatable *detector*

Source: Antonio Torralba

What Points would you choose?

- Where would you tell your friend to meet you?

Source: James Hays

What Points would you choose?

- Where would you tell your friend to meet you?

Source: James Hays

Properties of the ideal feature

- Local: features are local, so robust to occlusion and clutter (no prior segmentation necessary)
- Invariant to **scale, rotation, translation**, affine and perspective transforms
- Invariant to **brightness variation** in different images (different illuminations)
- Robust to **noise**, blur, discretization, compression etc.
- Quantity: many features can be generated for even small objects
- Accurate: precise **localization**
- Efficient: close to **real-time** performance

Credit: http://homes.esat.kuleuven.be/~tuytelaa/tutorial-ECCV06.pdf

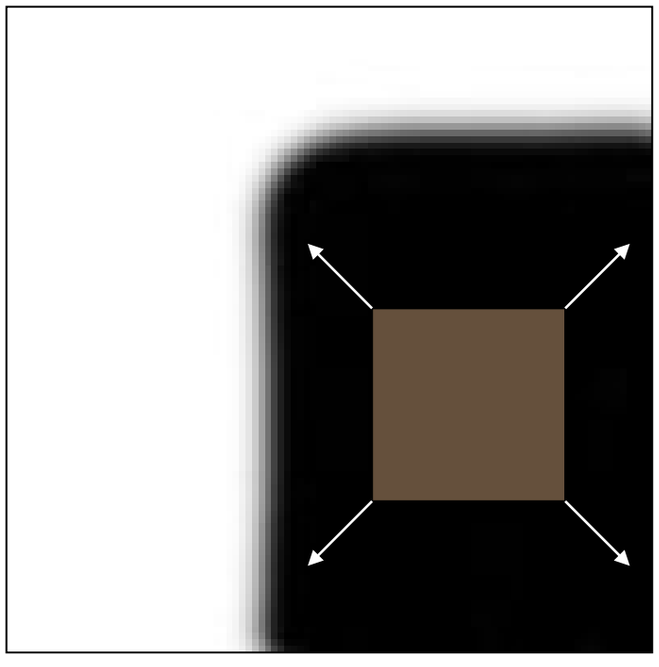
Good Candidates for Local Features are *Corners:*

Are edges good feature locations?

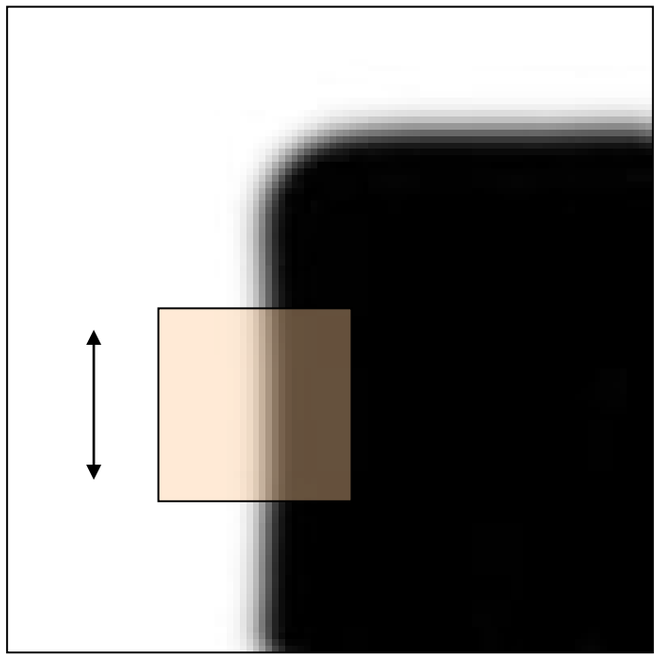
Credit Rick Szeliski, Svetlana Lazebnik, and Kristin Grauman

Why Corners?

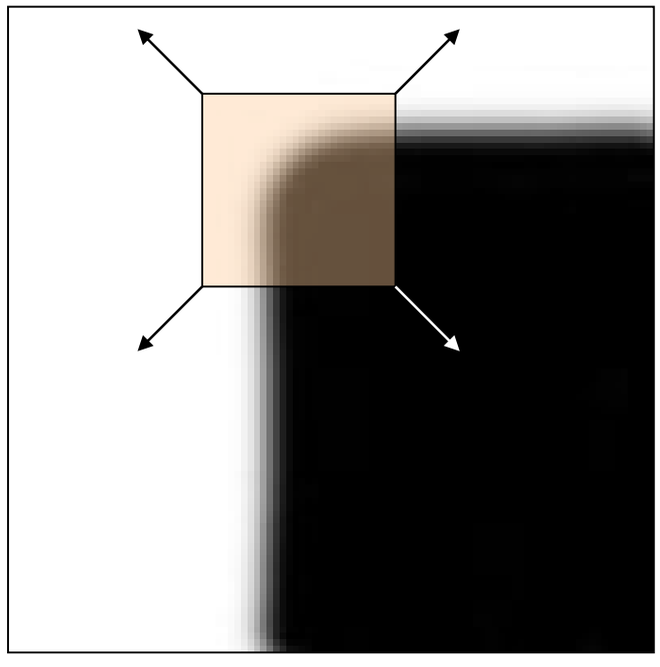
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction

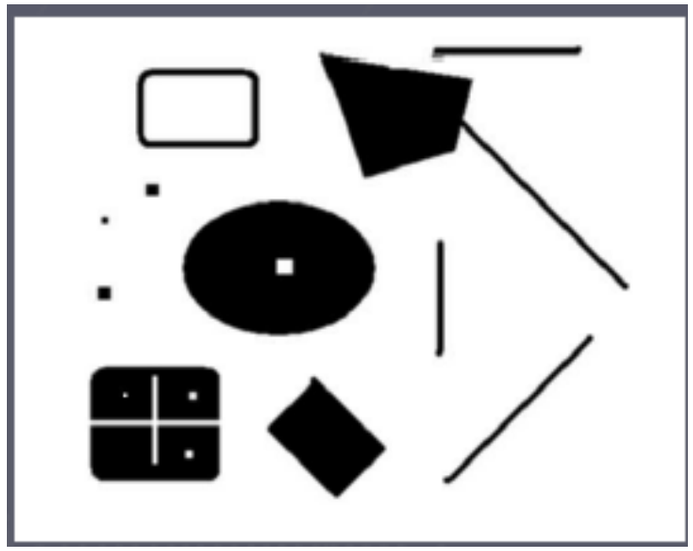


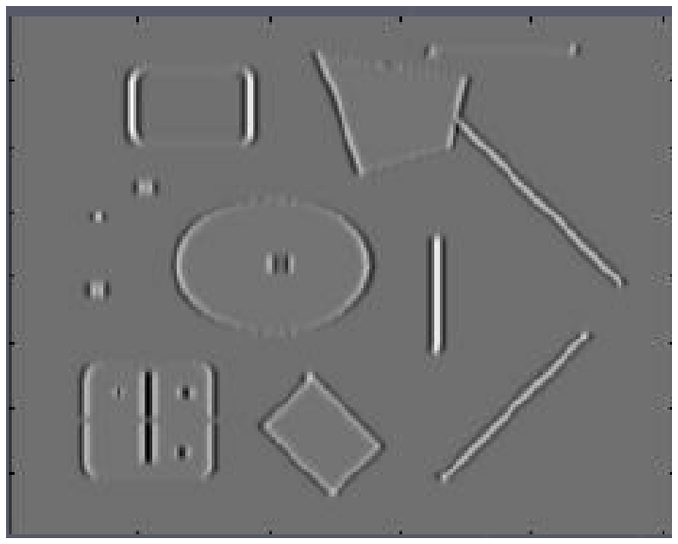
"corner":
significant change in all directions

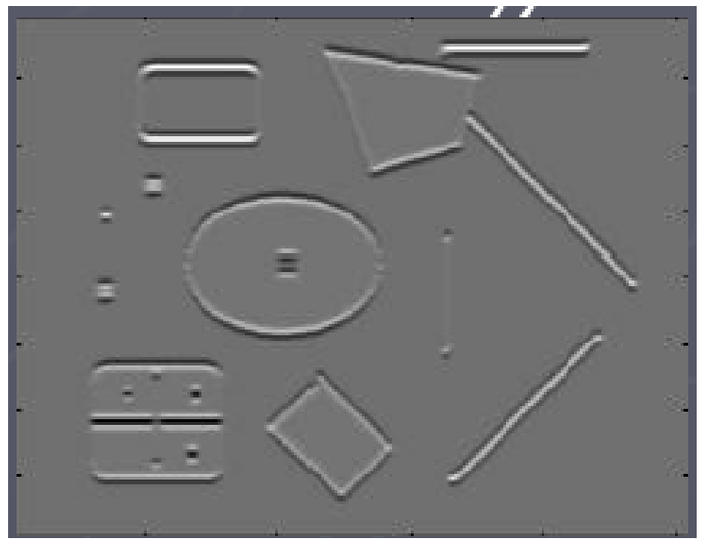
Source: A. Efros

The Harris Corner Detector

- A robust corner detector that is based on the **image gradient**
- Idea: find locations in the image where the gradient is strong in at least two directions
- So back to gradients and partial derivations:







Notation:

$$I_x \Leftrightarrow \frac{\partial I}{\partial x}$$

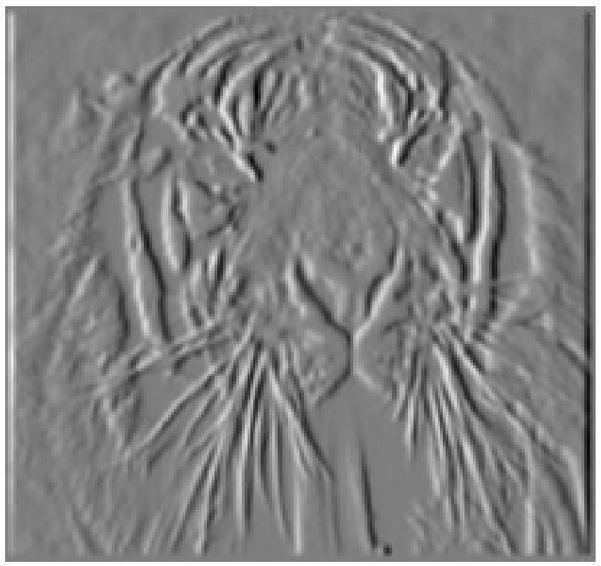
$$I_y \Leftrightarrow \frac{\partial I}{\partial y}$$

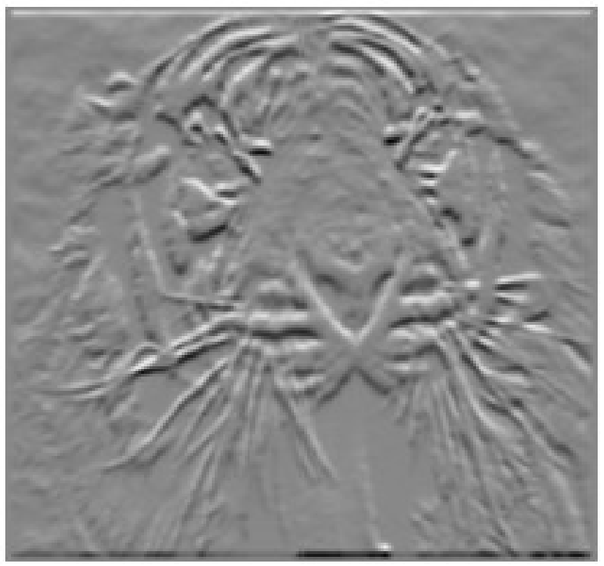
C.Harris and M.Stephens. "A Combined Corner and Edge Detector." *Proceedings of the 4th Alvey Vision Conference*: pages 147--151.

Recap: how to compute the derivation in X- or Y-direction?

$$\frac{\partial f(x,y)}{\partial x}$$

-1	1





$$\frac{\partial f(x,y)}{\partial y}$$

-1
1

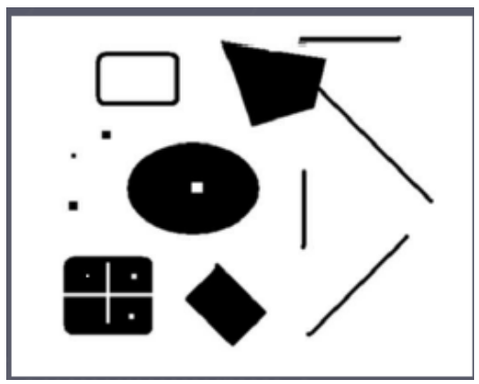
Recap: how to compute the derivation in X- or Y-direction?

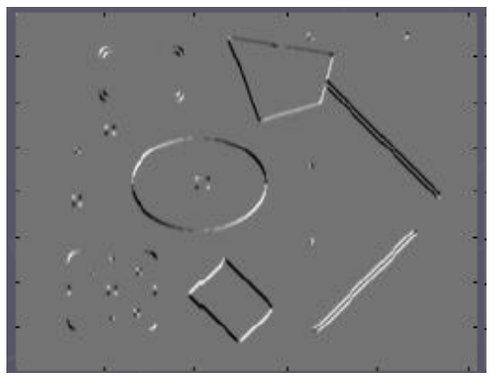
- The gradient map is then simply the product of horizontal derivative and vertical derivative of the image:

$$\left| \frac{\partial f(x,y)}{\partial x} * \frac{\partial f(x,y)}{\partial y} \right|$$

- That’s all we need to detect Harris corners…

Harris Corner Detection



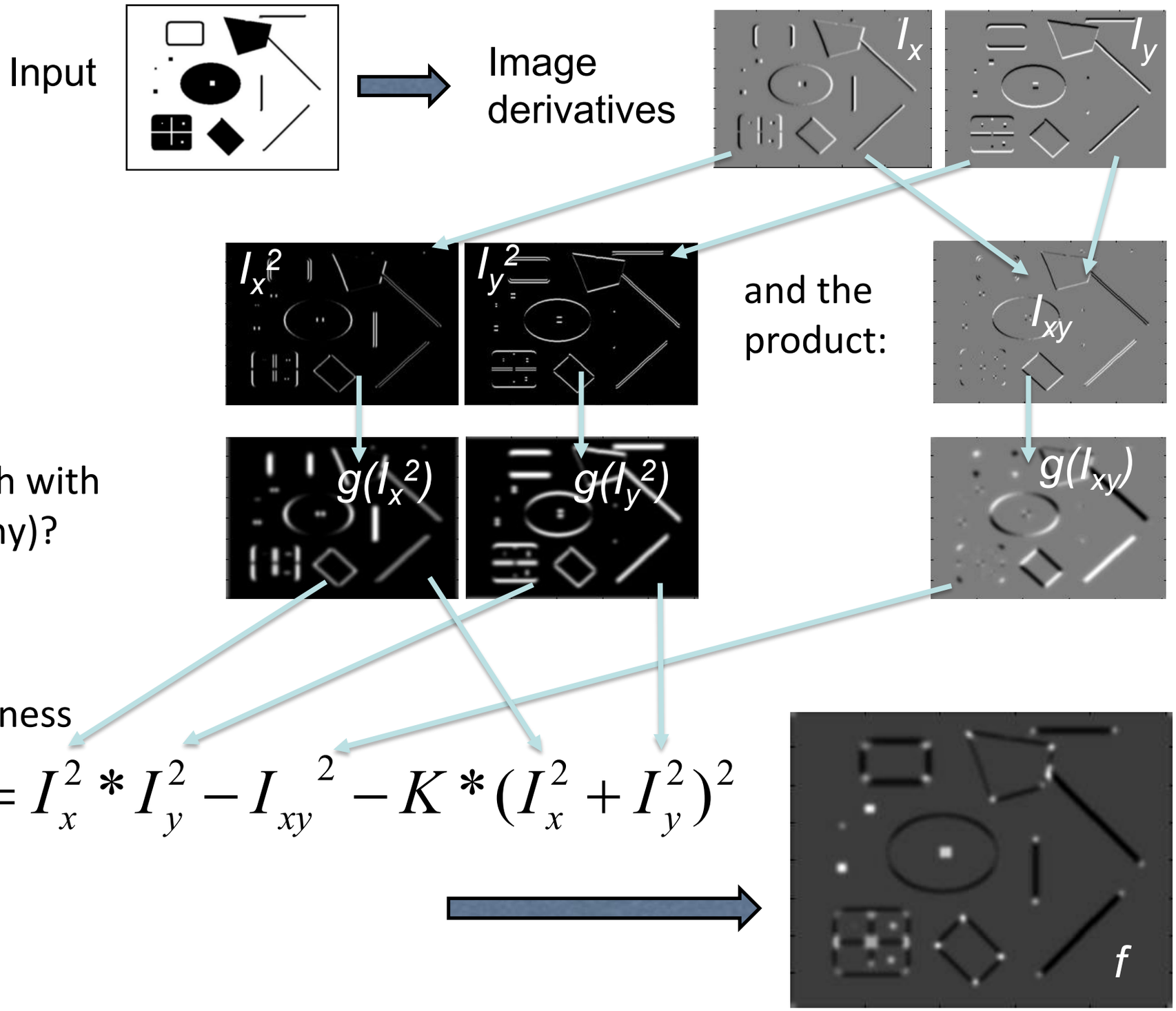



Notation:

$$I_x \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_y \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

Harris Corner Detection

- Computation:
  - 1. Compute image derivatives
  - 2. Compute square of derivatives
  - 3. Optionally smooth with Gaussian filter g (why)?
  - 4. Compute „cornerness function“



$$R = I_x^2 * I_y^2 - I_{xy}^{\ 2} - K * (I_x^2 + I_y^2)^2$$

Harris corner detector

1) Compute *cornerness* score *R* (=an image)

2) Find peaks in R, i.e. points which are surrounded by smaller values and take the points of local maxima as corners

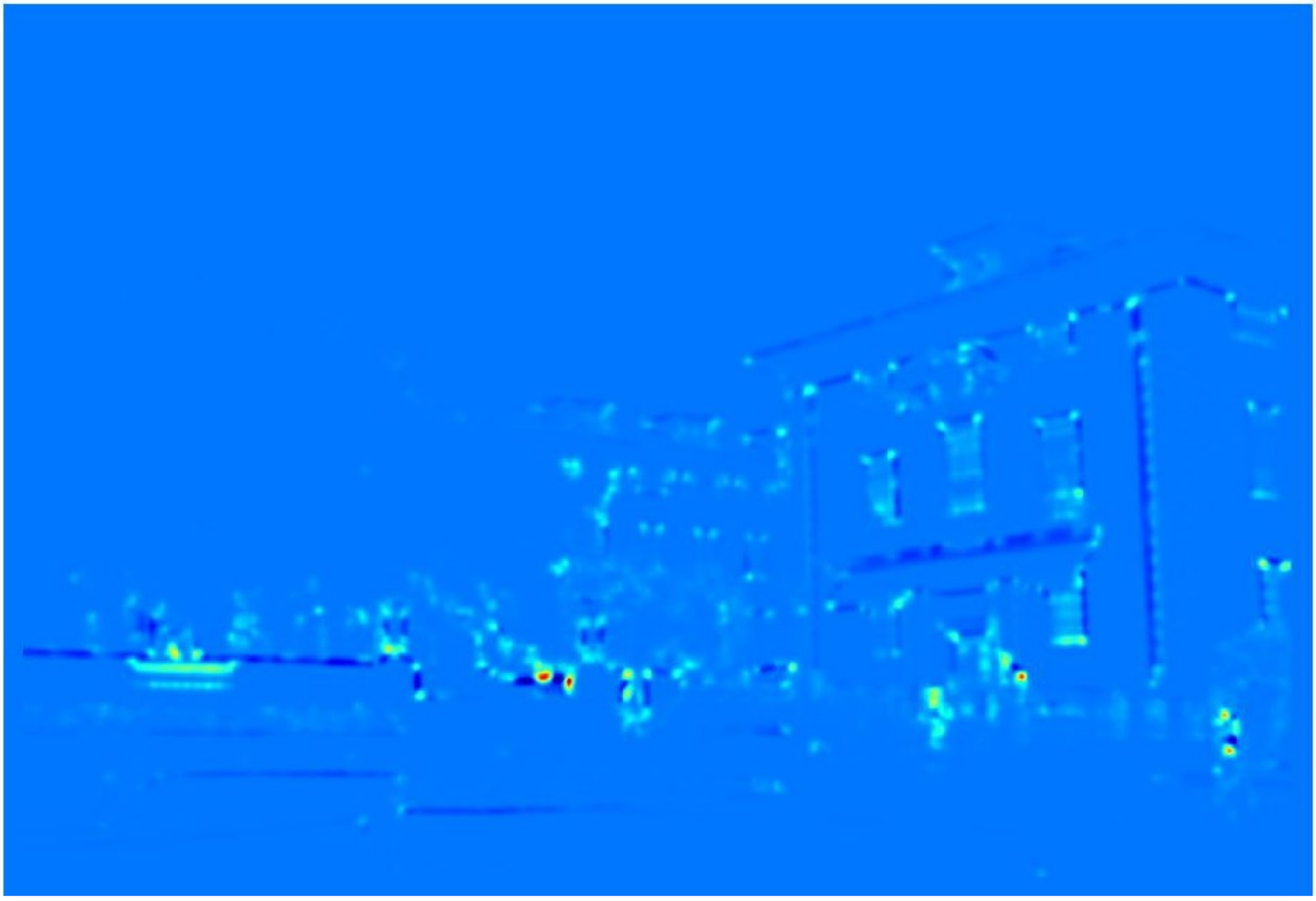
3) Or find the N strongest points by selecting all points *p* by sorting all intensities and selecting the first N entries.

Example of Harris application

Kristen Grauman

Example of Harris application

- Compute corner response at every pixel.



Kristen Grauman

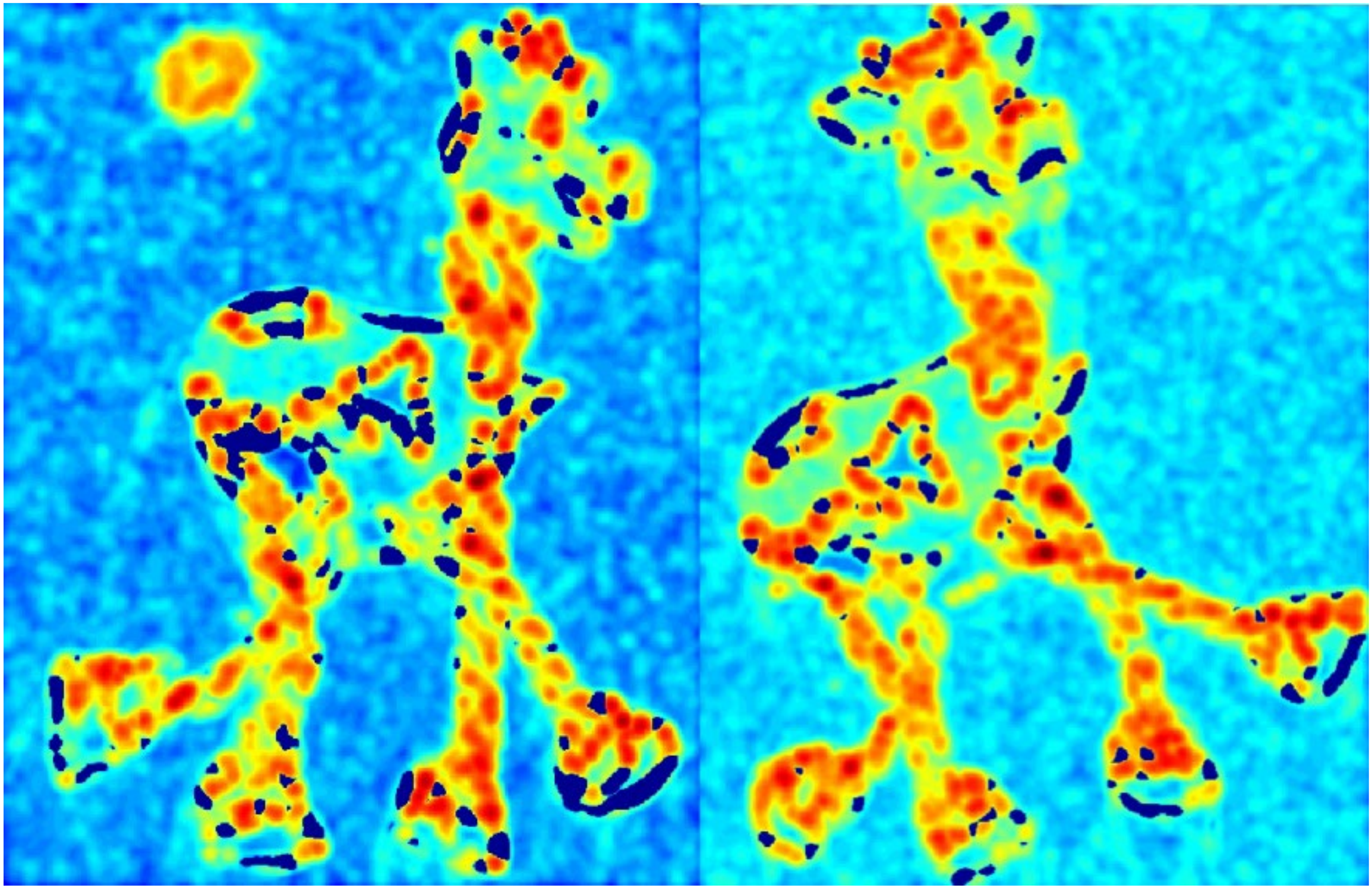
Example of Harris application

Kristen Grauman

Harris detector example – robustness?

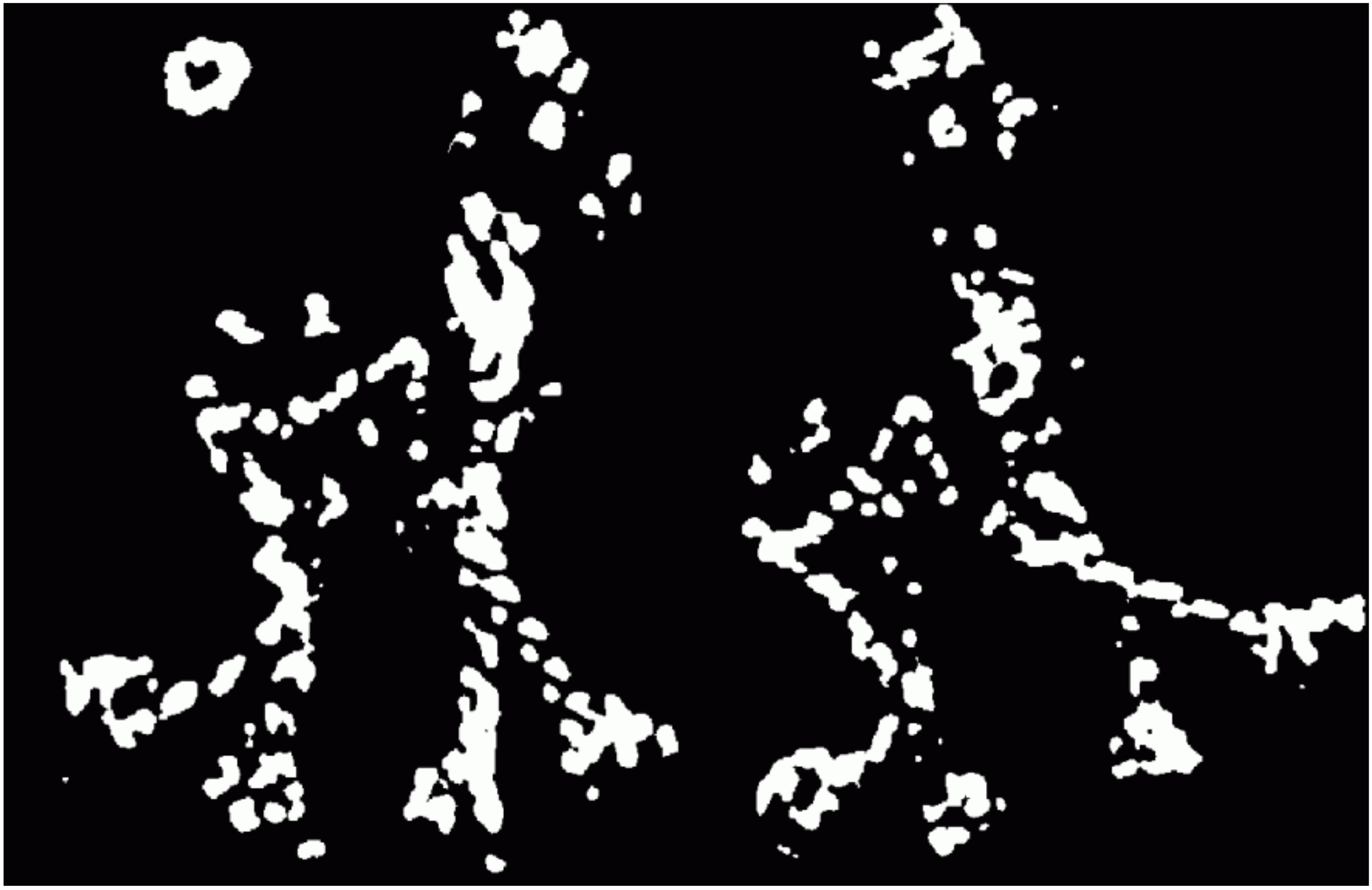
Kristen Grauman

Cornerness function R (red high, blue low)

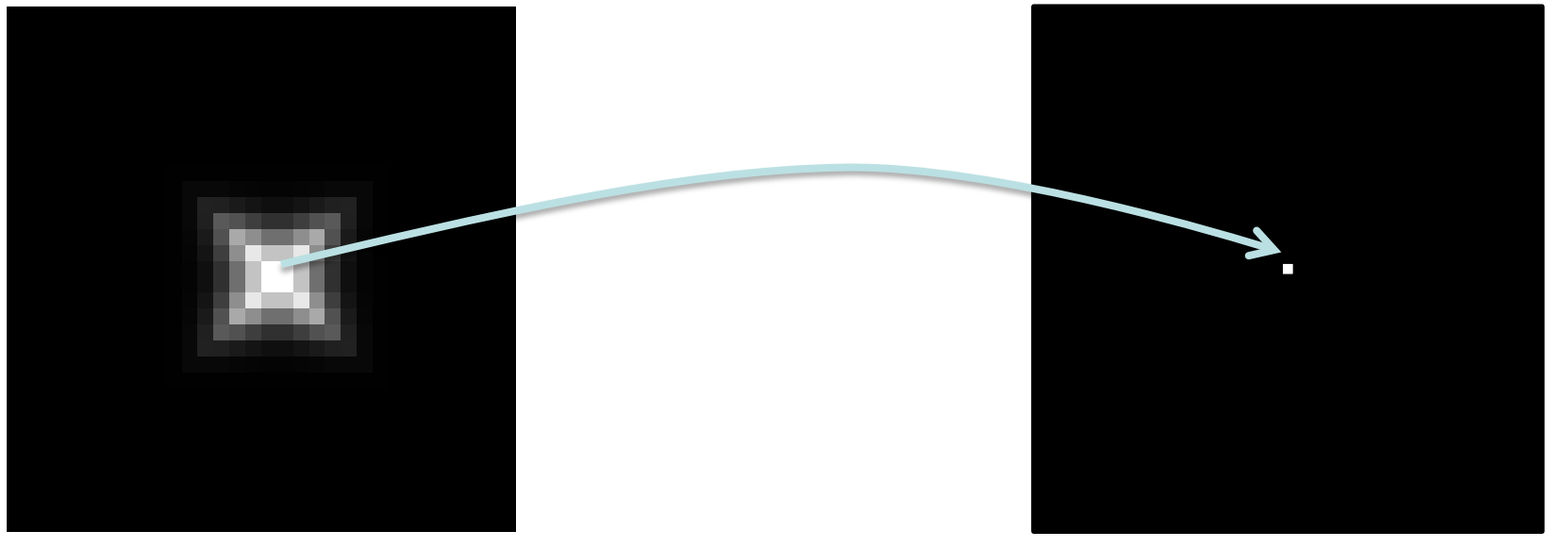


Kristen Grauman

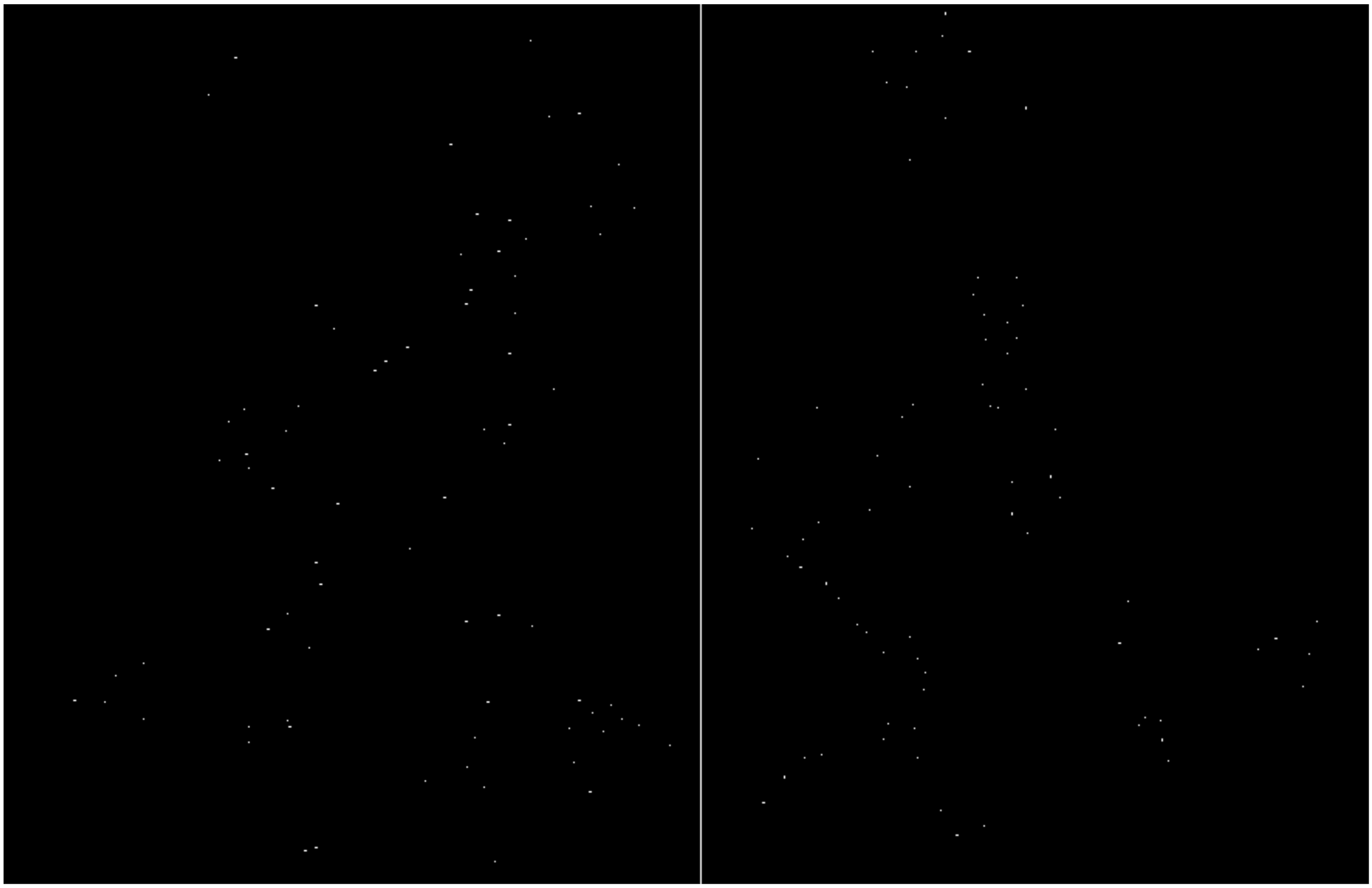
Threshold (f > value)



Search for maxima in Cornerness Function



Find local maxima of f

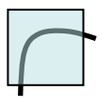


Harris features (in red)

Harris Corner Detector - Summary

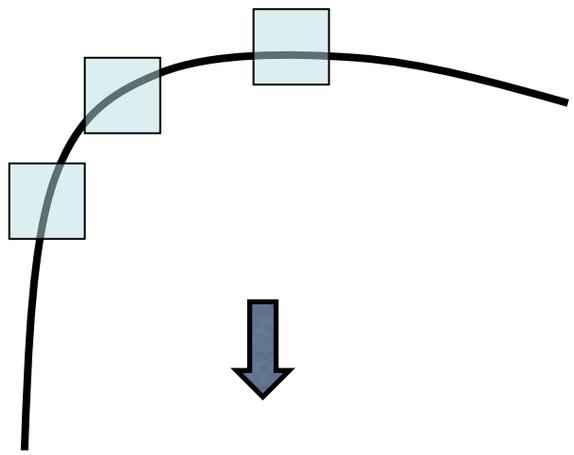
- Harris corner detector is a frequently and popular detector
- Easy to computer (from image derivatives only)
- Harris corners are rotation invariant and translation invariant
- Harris corner detector is, however, **not scale invariant**:

Scale 1:



Corner !

Scale 2:



The detector sees only edges but no corners!

Kristen Grauman

Corners vs. Local Features

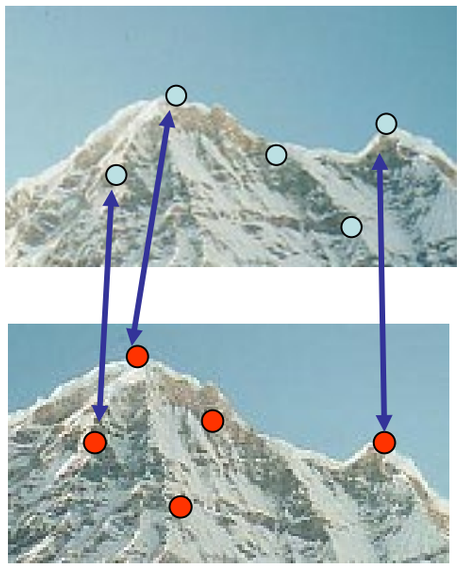
- A “local feature” is more than a “corner”
- Corner: has an x and y coordinate, i.e. is a point in the image
- Local feature
  - has also an x and y coordinate, but also has:
    - **scale** and
    - **orientation**
- Scale: the size of the structure that generates the corner (e.g. a corner in a window has a smaller scale than a corner produced by the roof of a building
- Orientation: imagine a corner related to the rim (Felge) of a car wheel. The same corner in different images of the wheel might be differently rotated. Orientation helps to compensate this rotation.

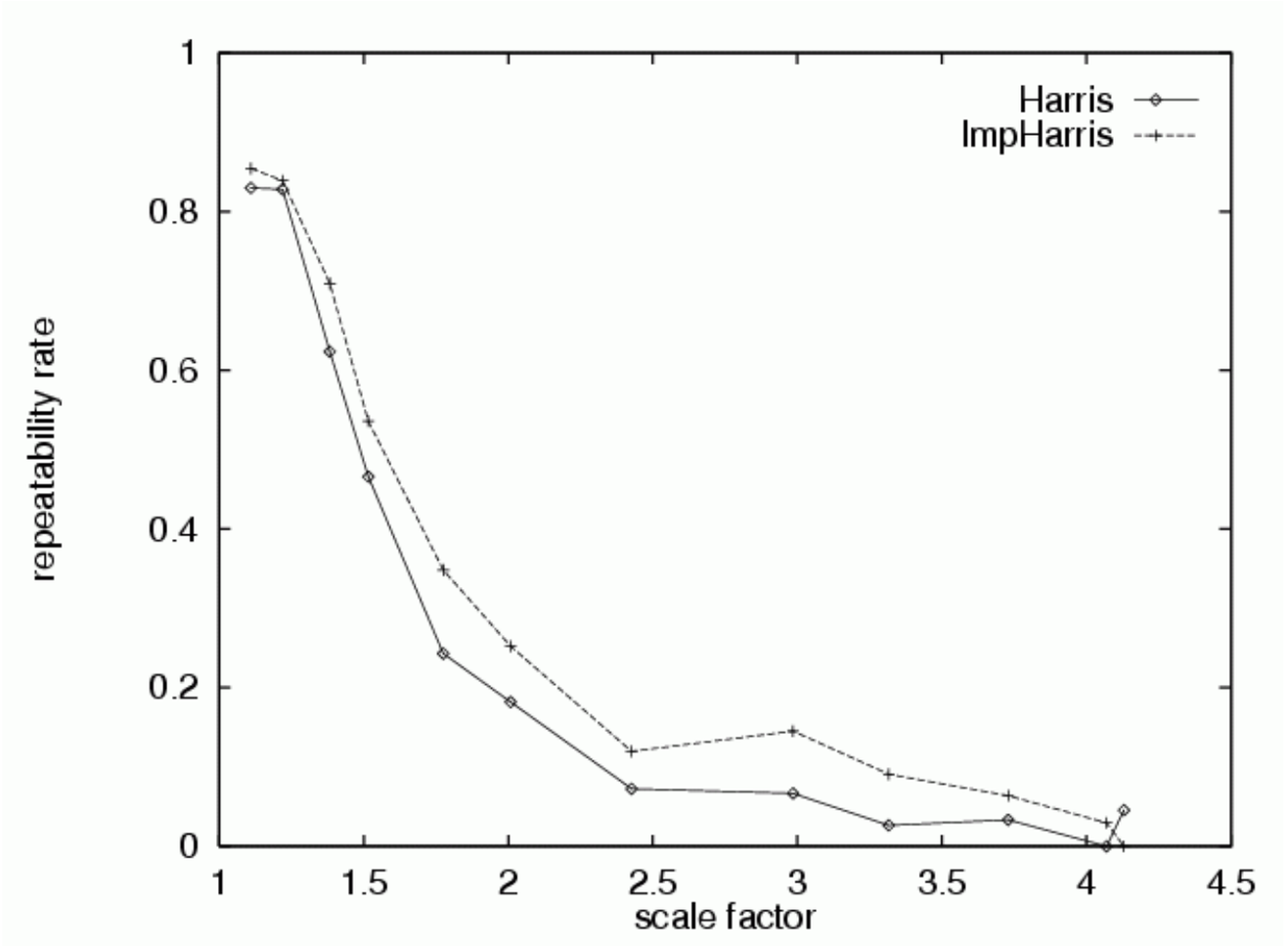
Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability rate:

$$\frac{\#\text{ correspondences}}{\#\text{ possible correspondences}}$$





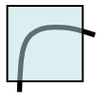
C. Schmid et.al. “Evaluation of Interest Point Detectors”. IJCV 2000; Slide Credit: Antonio Torralba

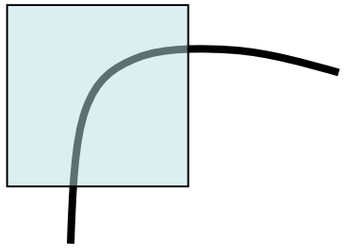
How to get a scale invariant detector?

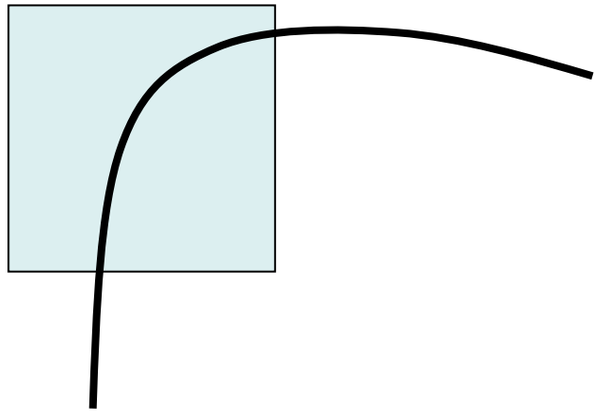
- To match (or stitch) these images we need a feature detector whose detections are *repeatable* across different scales

How to get a scale invariant detector?

- Apply corner detection with different window sizes

Scale 1: 

Scale 2: 

Scale 3: 

- Apply same corner detection to different sizes of images! → „image pyramid“

Multi-scale corner detection

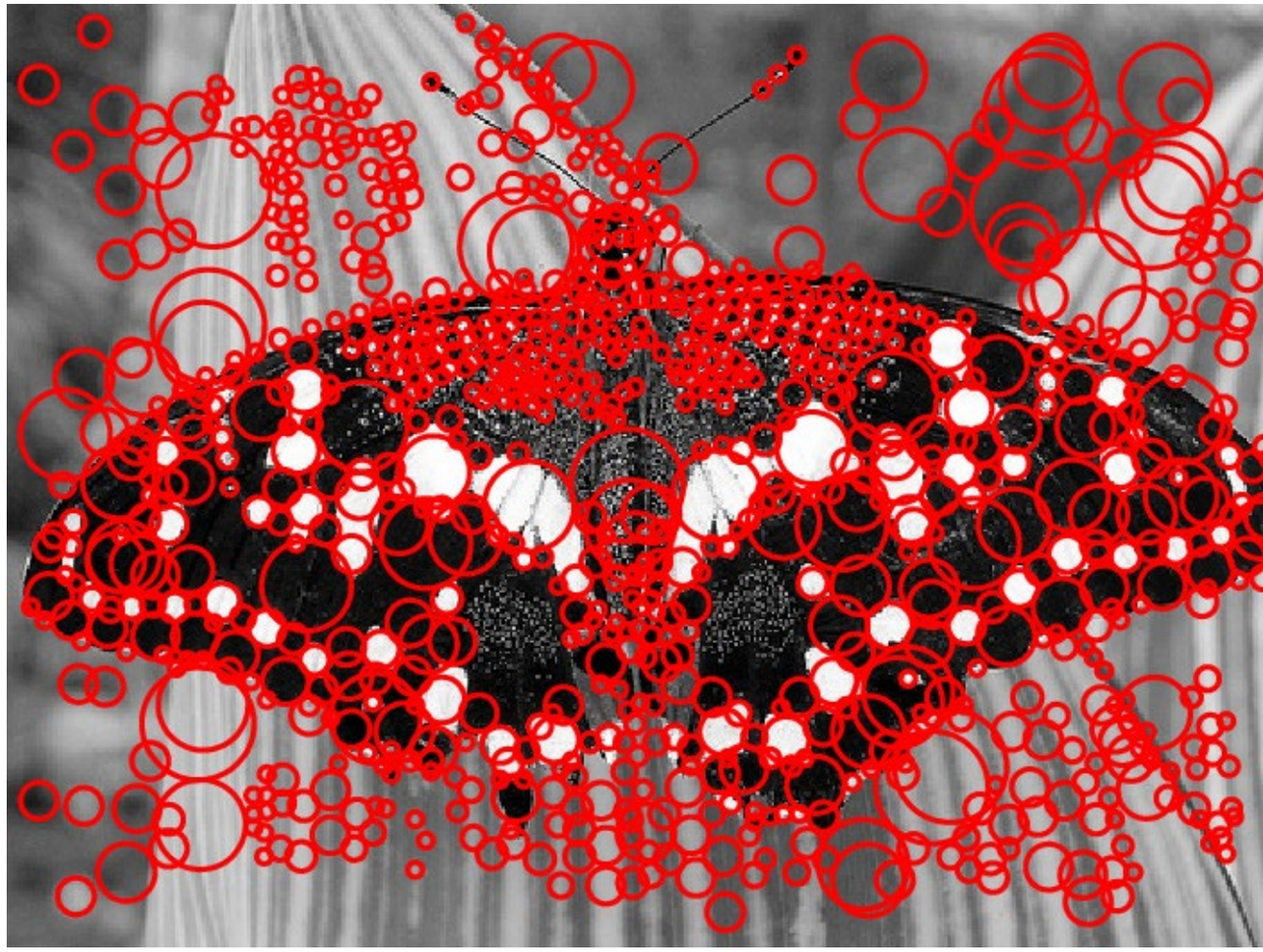


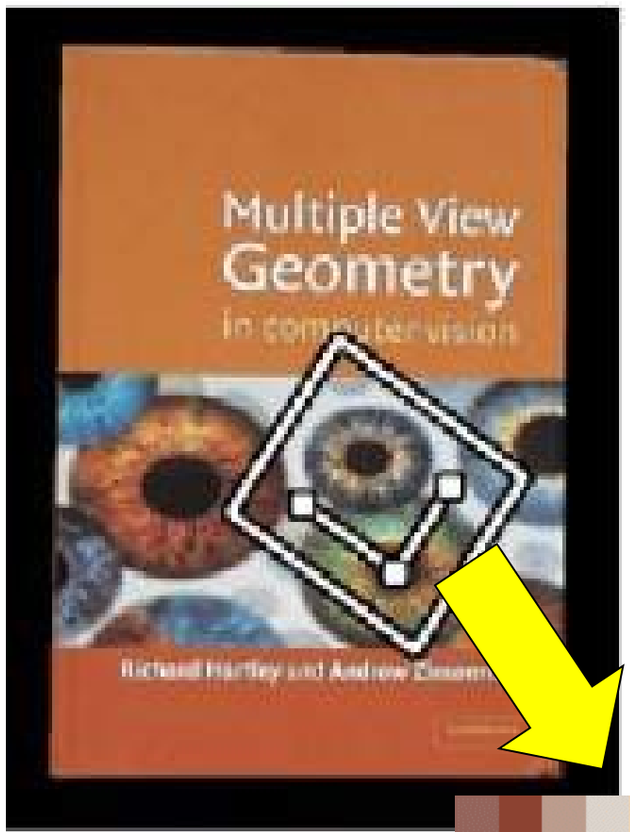
Image credit: Lana Lazebnik

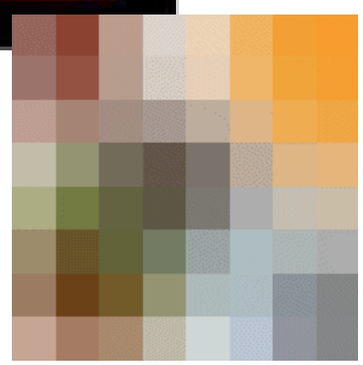
Orientation Normalization

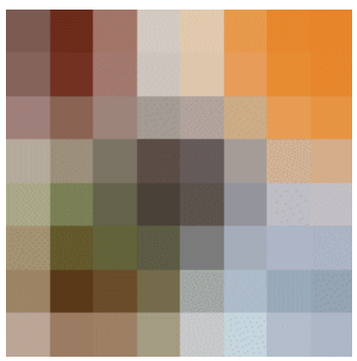
- Now every feature point has a **scale** (a “size”)
- But what about the **orientation** of the point?

Orientation Normalization

- Now every feature point has a **scale** (a “size”)
- But what about the **orientation** of the point?

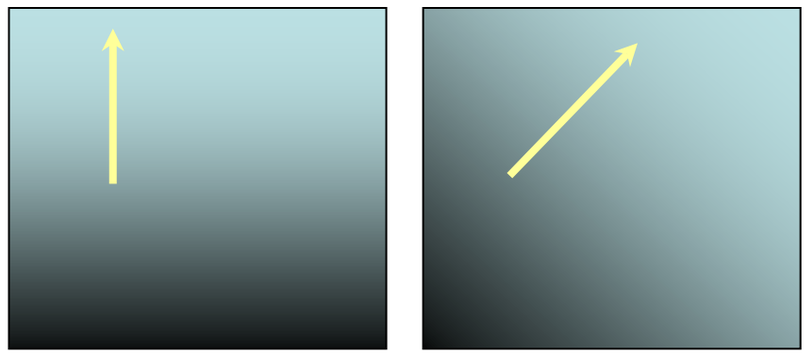






Find local orientation of feature point:

- Use the dominant direction of the gradient for the entire image patch



- Rotate feature point according to this angle
- Now all feature points can be compared in a rotation normalized way. In other words: differently rotated local features can be compared directly

Credit: Travor Darrel

Local features: Each corner gets a scale and an orientation

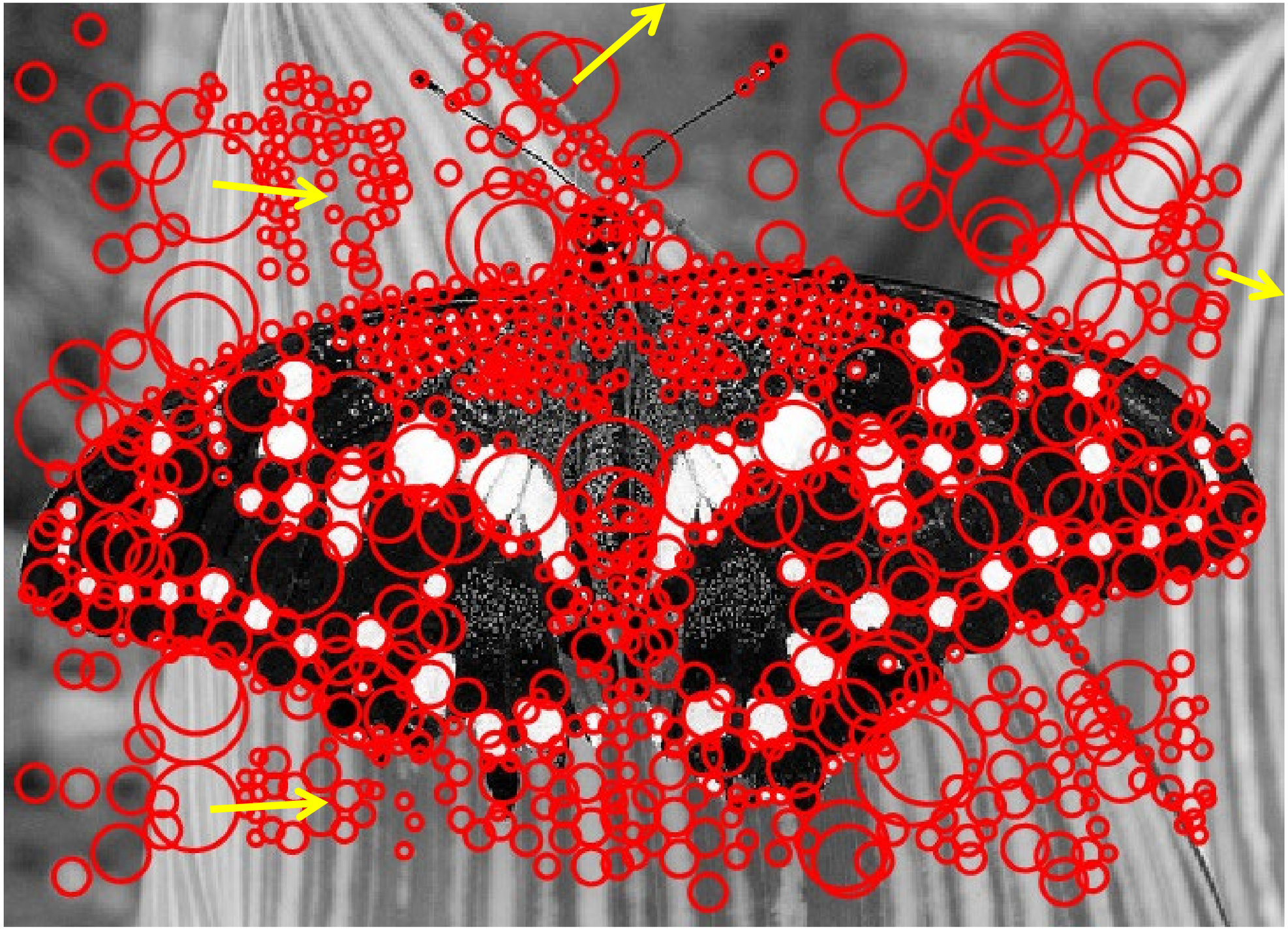
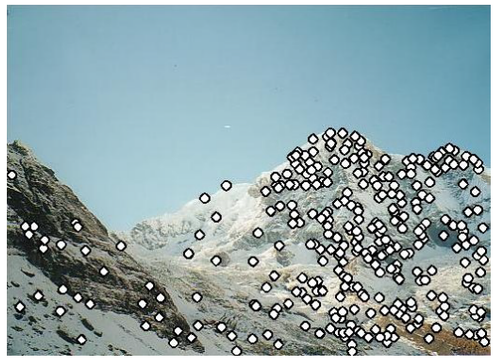


Image credit: Lana Lazebnik

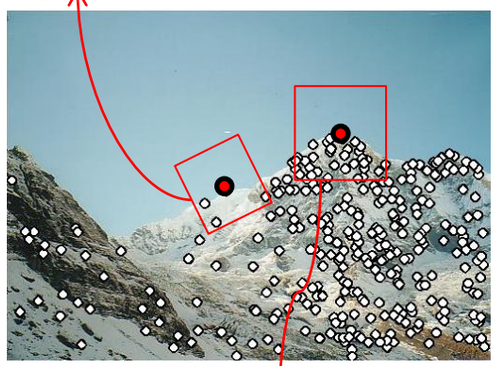
Overview – The Big Picture

- 1. **Detection**: identify the interest points



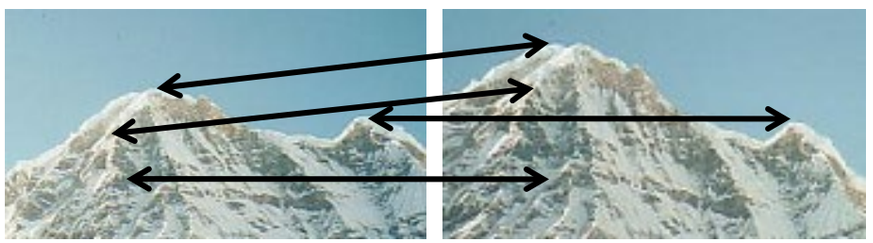
- 2. **Description**: extract a vector (feature descriptor) that represents the surrounding of each feature point

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

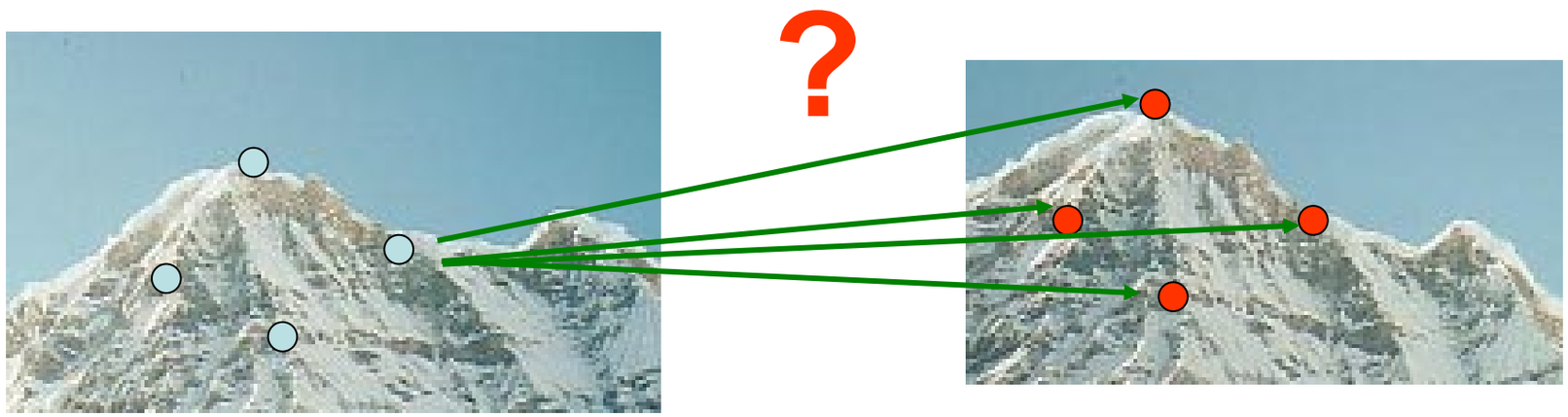
- 3. **Matching**: determine correspondence between descriptors in two views



Source: Antonio Torralba

Major Challenge:

- For each point correctly recognize the corresponding one



We need a reliable and distinctive *descriptor*

Source: Antonio Torralba

Example

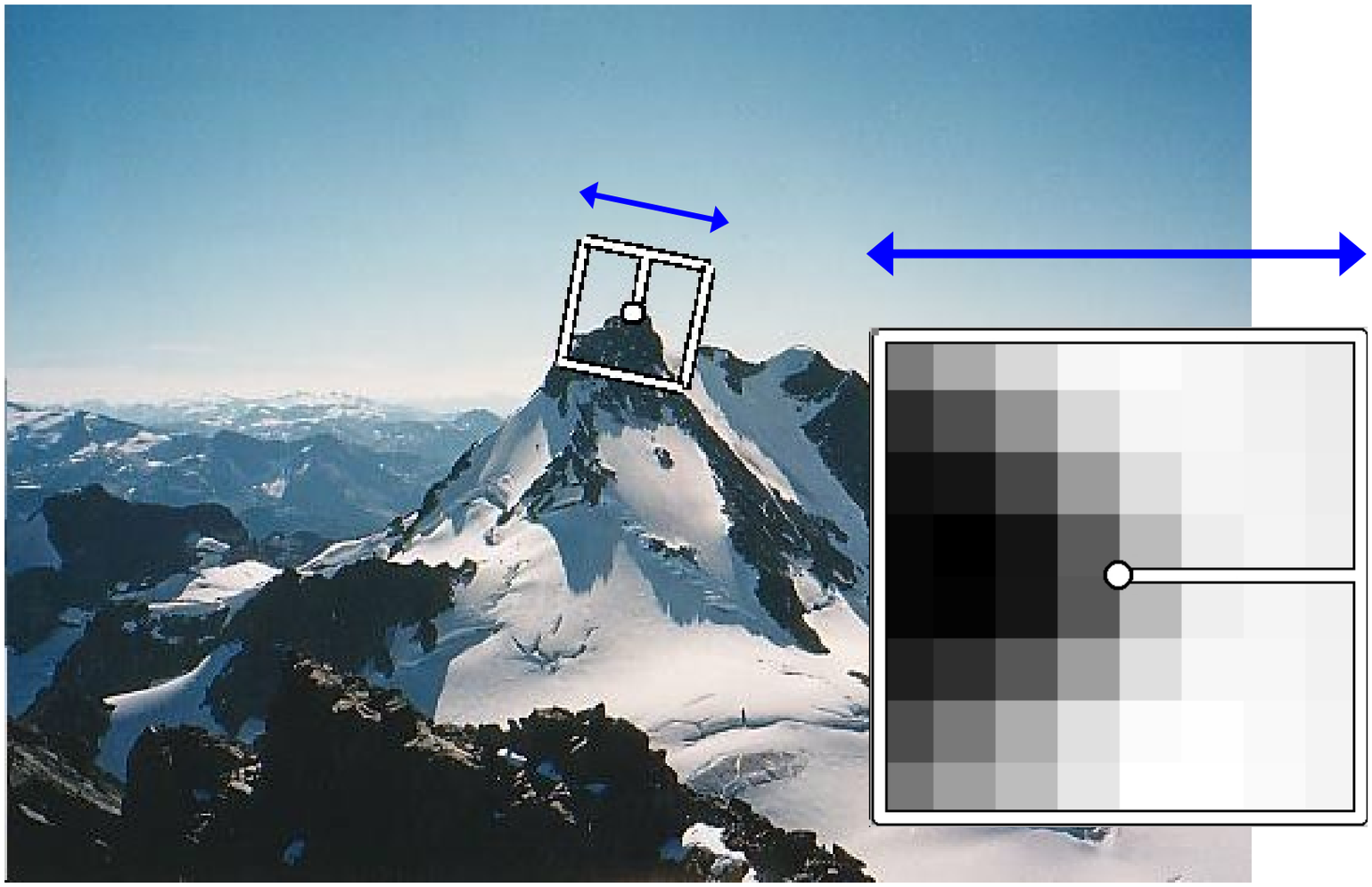


Image from Matthew Brown

Detections at multiple scales and with orientation

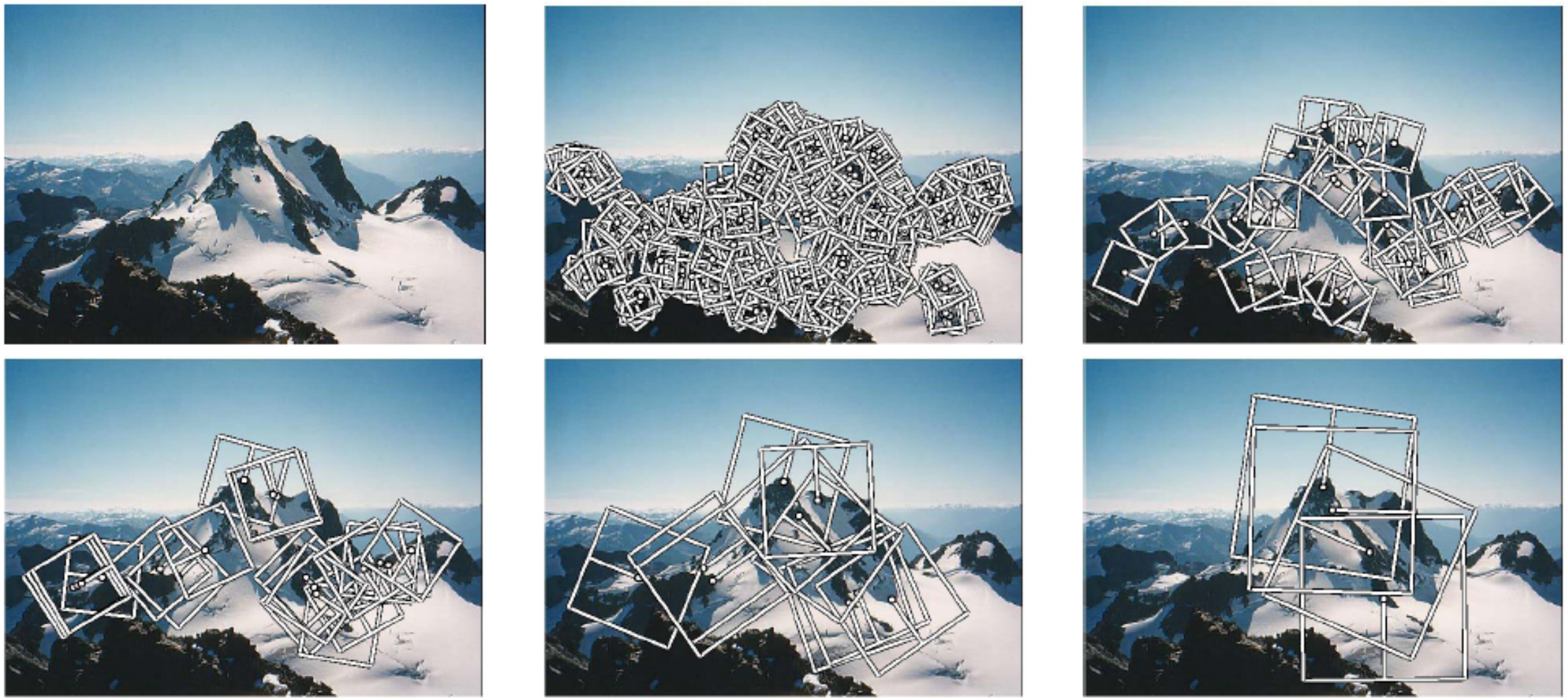
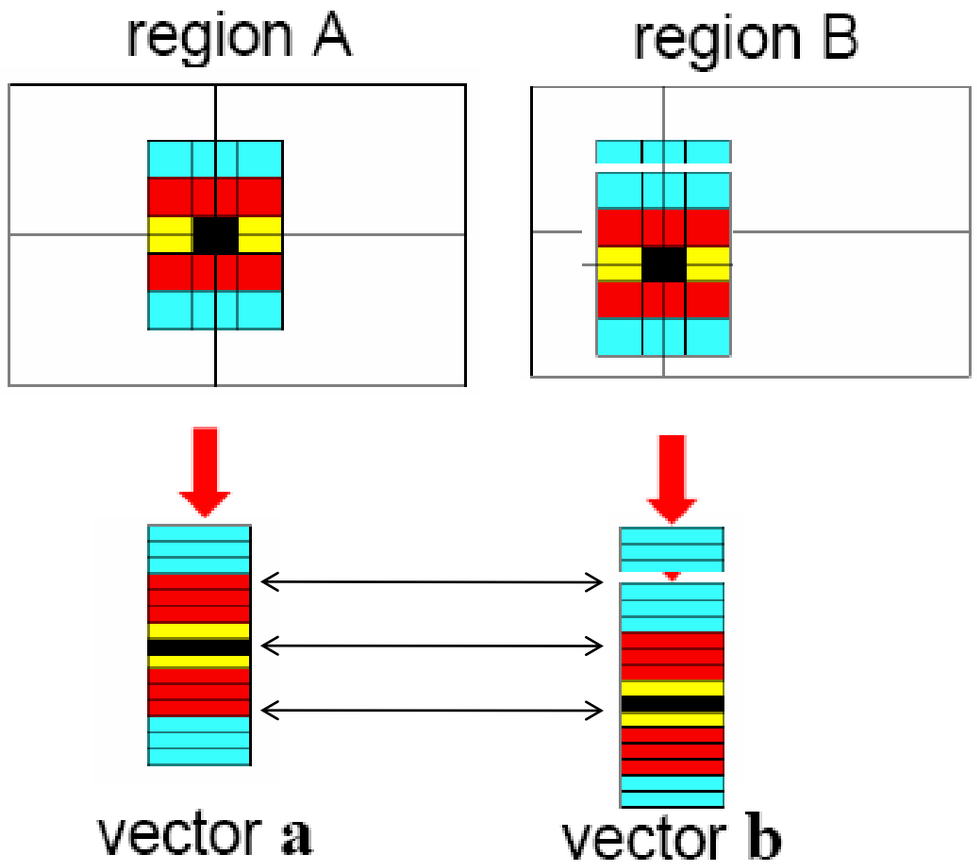


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Image credit: S. Szeliski

How to describe the surrounding of a feature point?



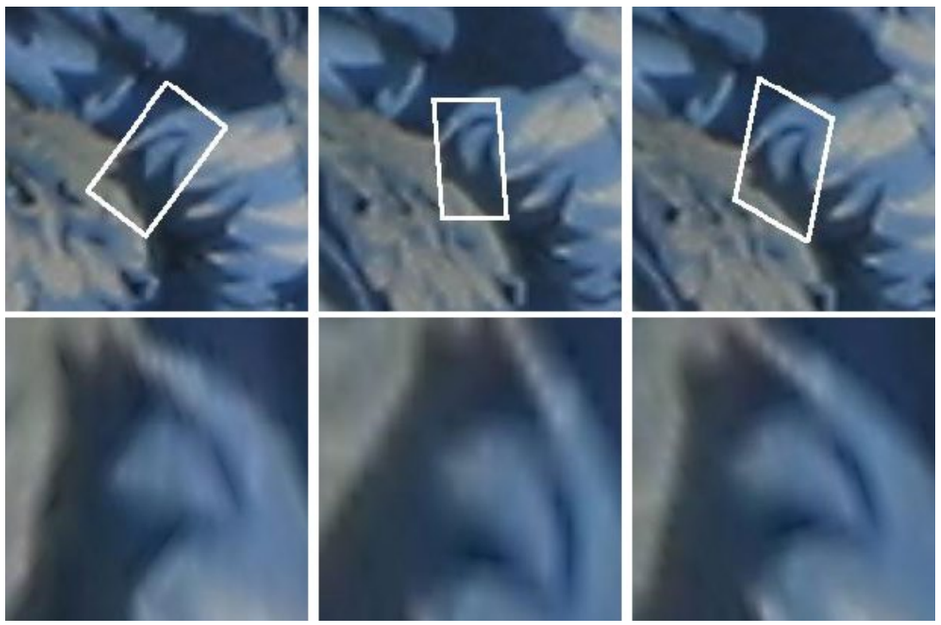
The simplest way to describe the neighborhood around an interest point is to write down the list of intensities to form a feature vector.

But this is very sensitive to even small shifts, rotations.

Credit: Travor Darrel

Feature descriptors

- Disadvantage of patches as descriptors: Small shifts can affect matching score a lot

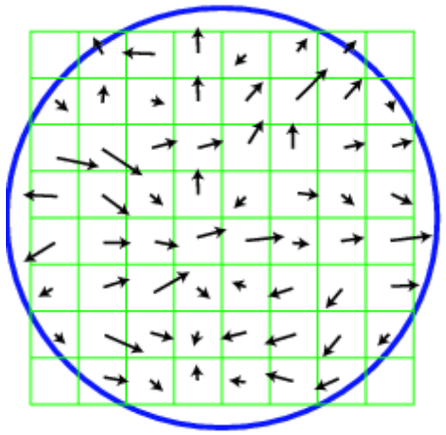


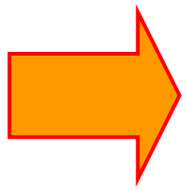
- List of pixels is a bad idea..

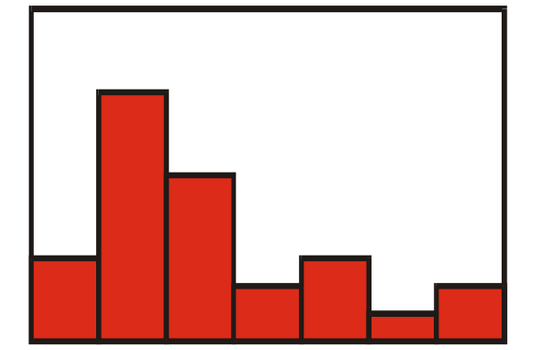
Source: Lana Lazebnik

Feature descriptors

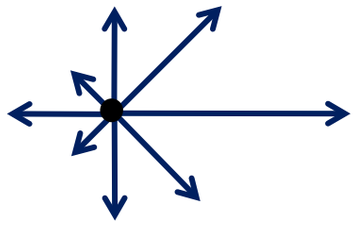
- Better: not pure pixels but image gradients (edge directions)
  - Advantage: more robust to illumination variations
- Aggregate over several pixels using histogram
  - To compensate small shifts
- Histograms of gradients







Histogram

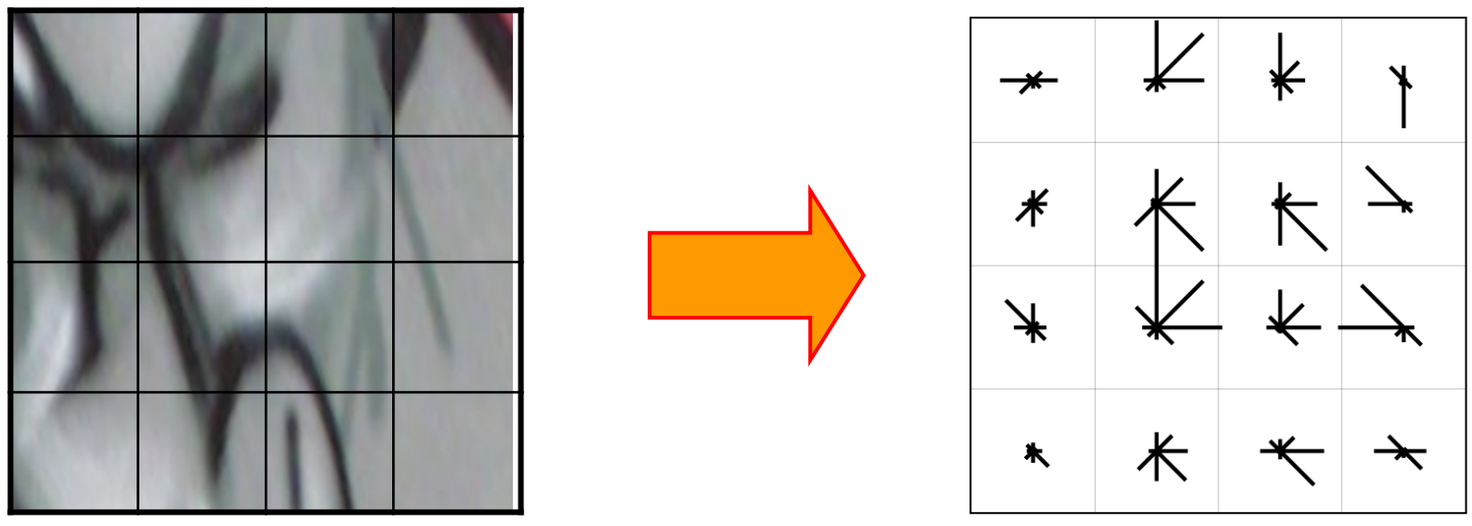


Gradient Lengths

Source: Lana Lazebnik

Feature descriptors: SIFT

- **S**cale **I**nvariant **F**eature **T**ransform
- Descriptor computation:
  - Divide patch into 4x4 sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch. Includes also magnitude!
- Resulting descriptor: 4x4x8 = 128 dimensions

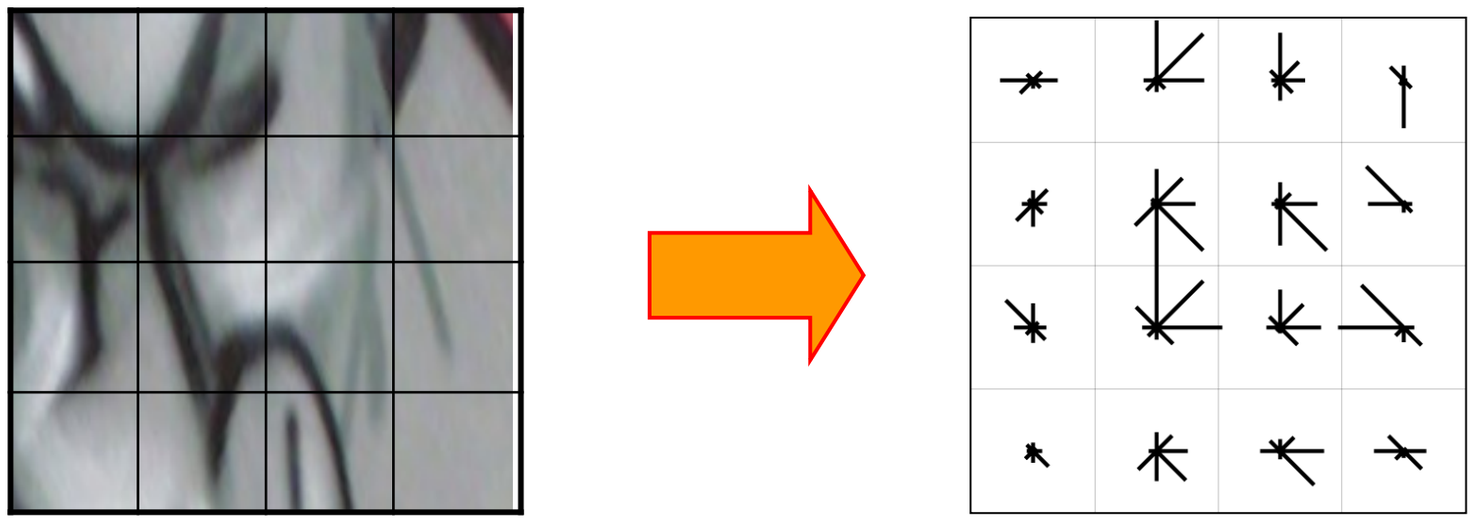


David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Source: Lana Lazebnik

Feature descriptors: SIFT

- **S**cale **I**nvariant **F**eature **T**ransform
- Descriptor computation (cont.):
  - Normalize vector to length of 1 (→ robustness to lightning variations)
  - Remove entries < 0.2 (very small gradients – probably noise)
  - Renormalize vector again to length of 1

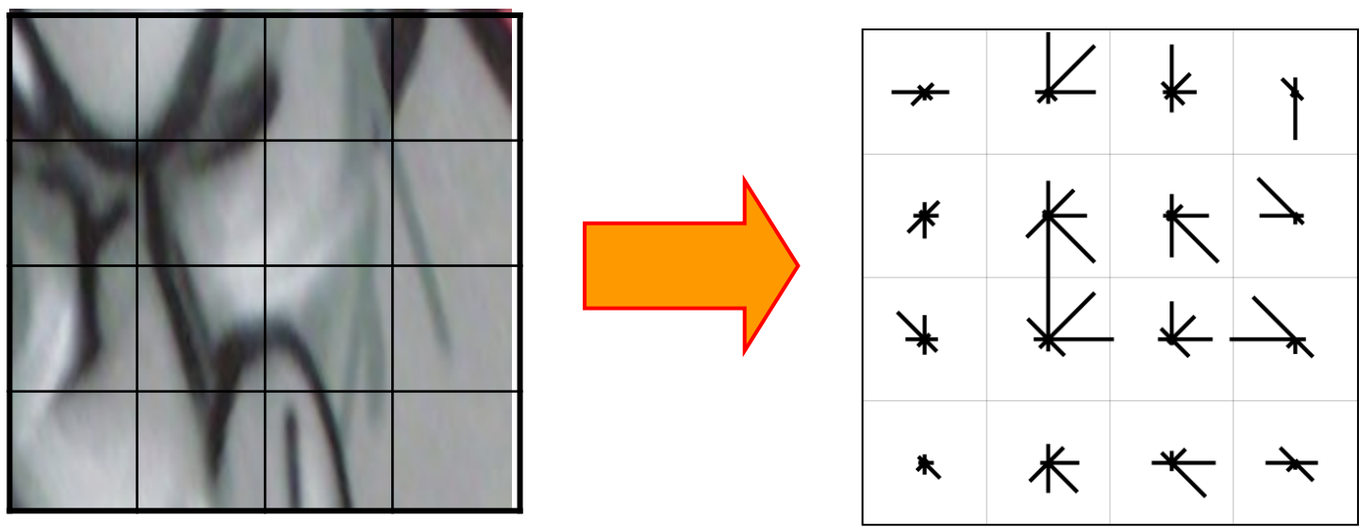


David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Source: Lana Lazebnik

Feature descriptors: SIFT

- Why subpatches?
- What invariances do we get?
  - Illumination?
  - Translation?
  - Scale?
  - Rotation?



Credit: Travor Darrel

Feature descriptors: SIFT

- Extraordinarily robust descriptor
  - Can handle changes in viewpoint: Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination: sometimes even day vs. night (below)
  - Fast and efficient
  - Lots of code available (however patented!)
  - Today many different types of descriptors exist: **ORB**, GLOH, HOG, ...

Roadmap

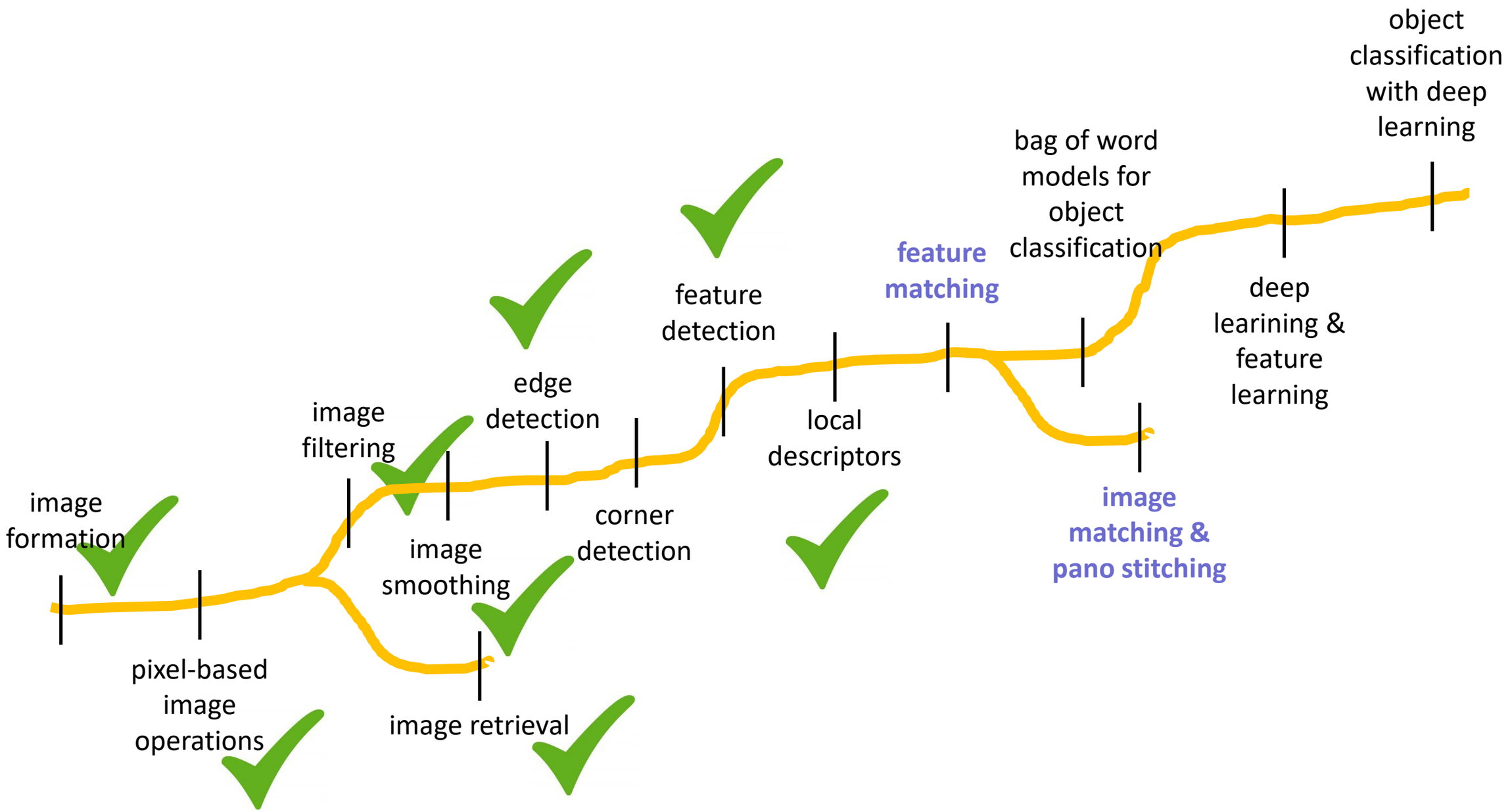
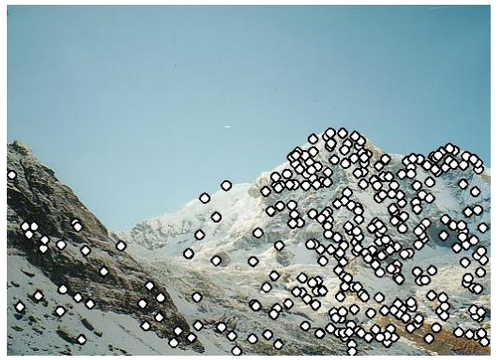


image formation
pixel-based image operations
image filtering
image smoothing
edge detection
corner detection
feature detection
image retrieval
local descriptors
feature matching
bag of word models for object classification
image matching & pano stitching
deep learining & feature learning
object classification with deep learning

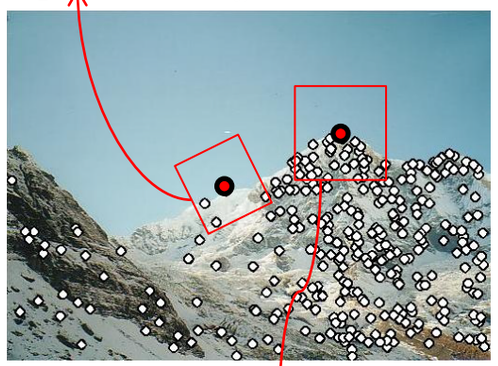
Overview – The Big Picture

- 1. **Detection**: identify the interest points



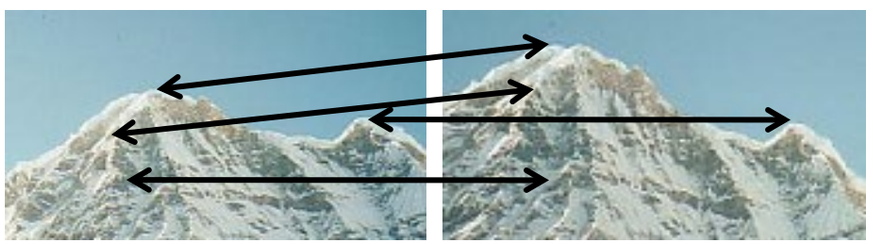
- 2. **Description**: extract a vector (feature descriptor) that represents the surrounding of each feature point

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

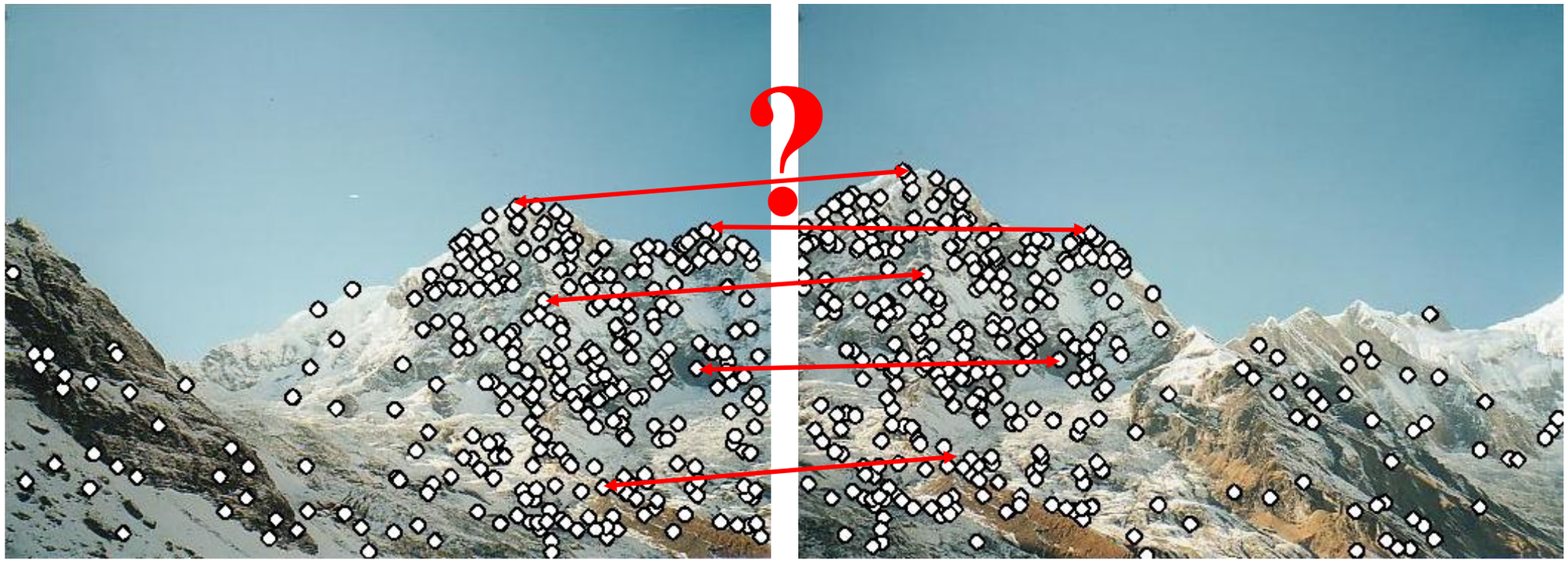
- 3. **Matching**: determine correspondence between descriptors in two views



Source: Antonio Torralba

Finding corresponding points

- We know now how to detect and describe local feature points. How can we use them now to match images?



Feature matching

Given a feature in image I_1, how to find the best match in image I_2?

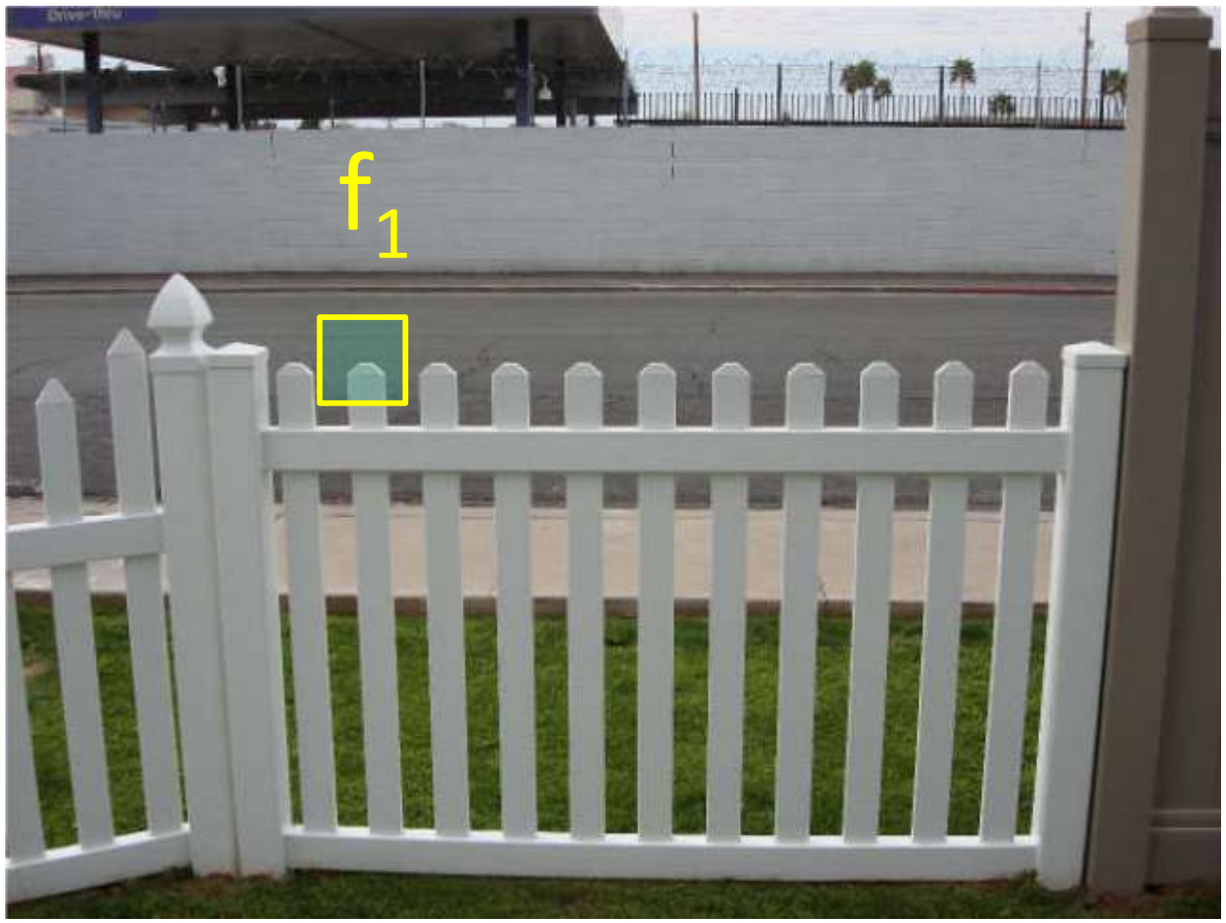
1. Define distance function that compares two **descriptors**
2. Test all the features in I_2, find the one with **minimum distance**

Credit: S. Szeliski

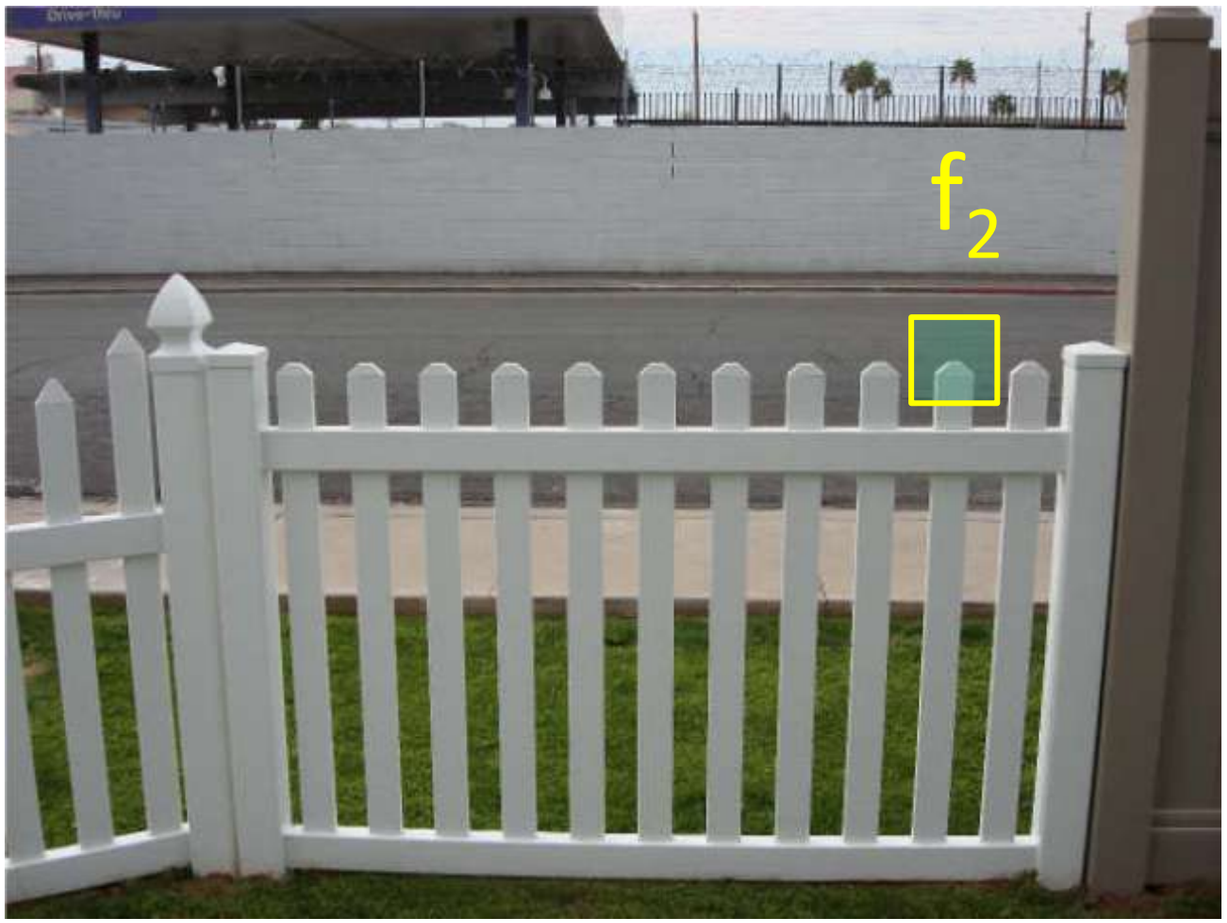
Feature distance

How to define the difference between two features f_1, f_2?

- Simple approach is SSD(f1, f2)
  - **Sum of square differences** between entries of the two descriptors
  - Can give good scores to very ambiguous (bad) matches



I_1



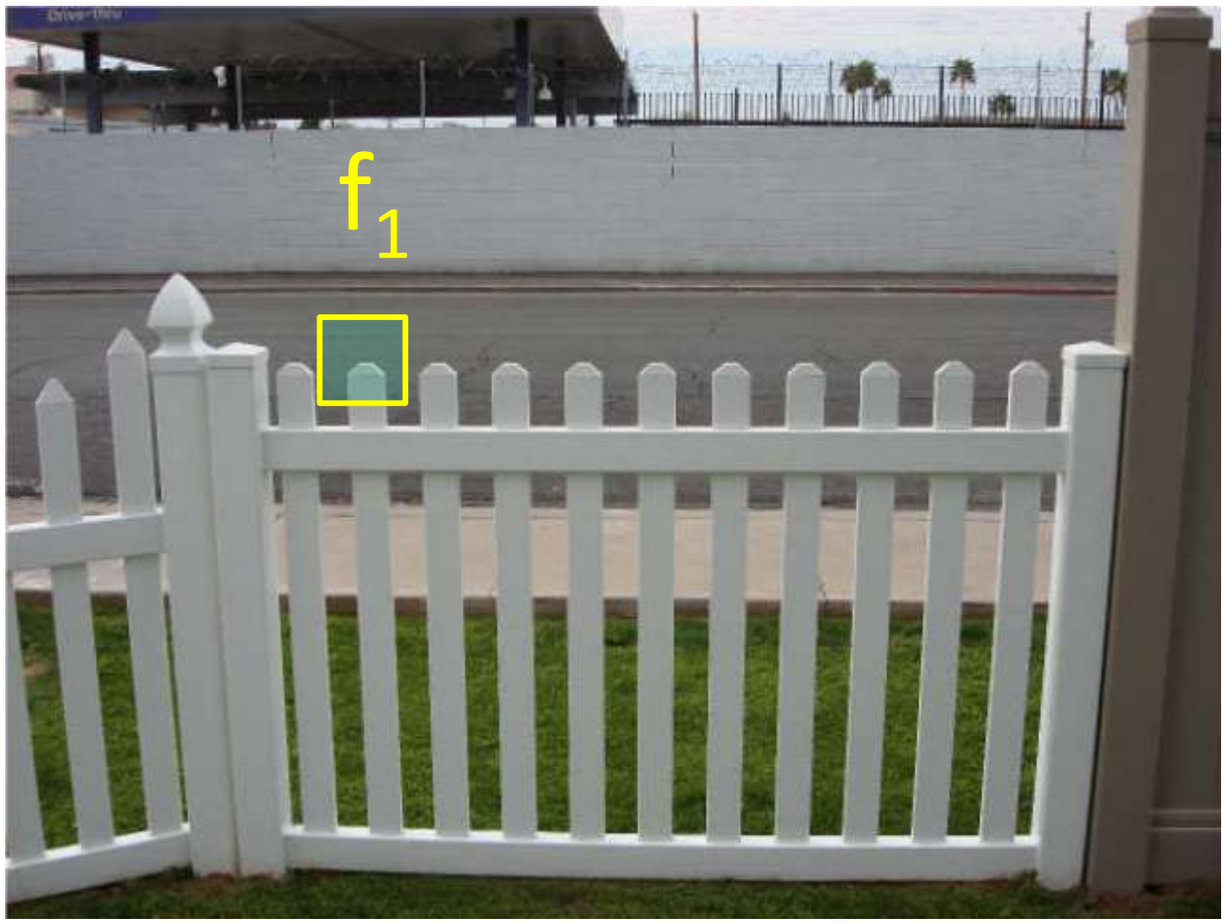
I_2

Credit: S. Szeliski

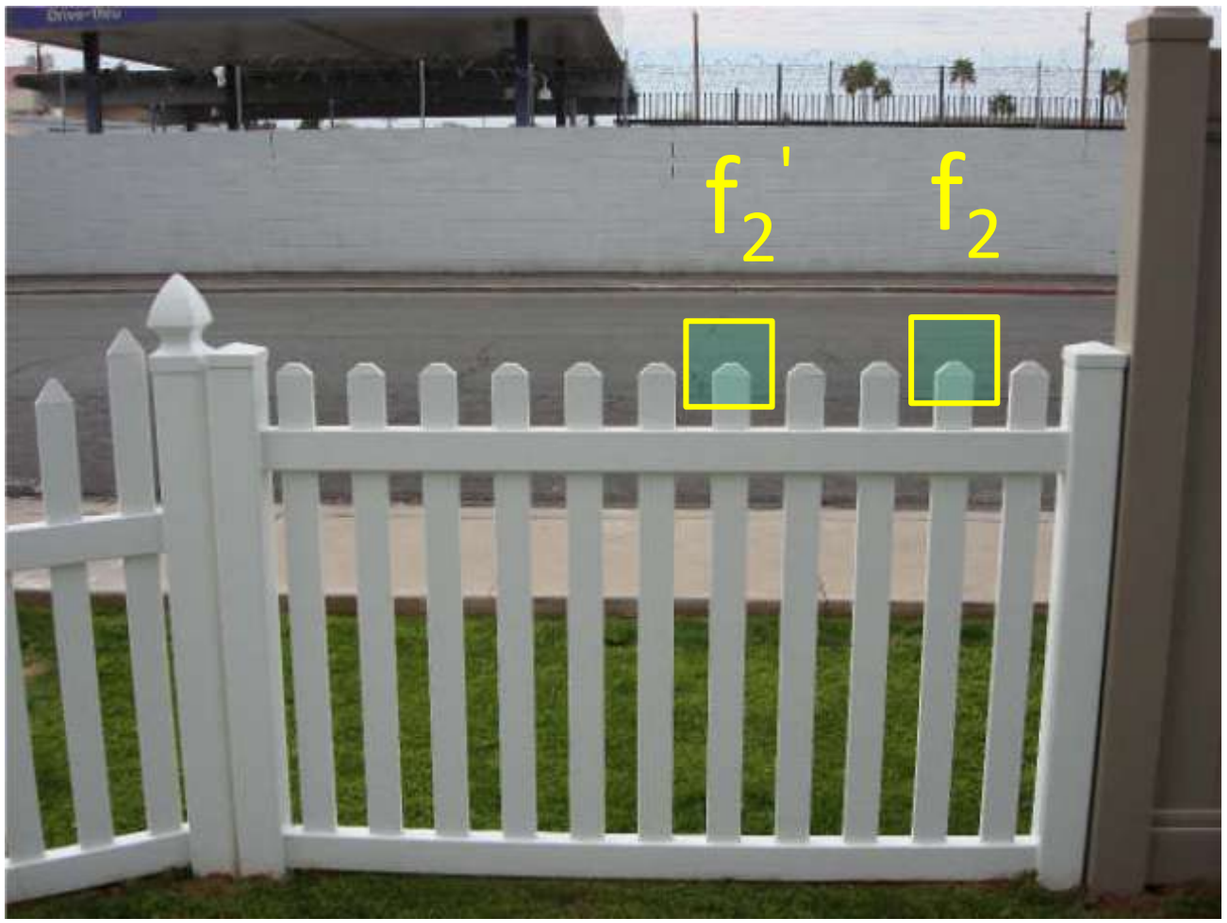
Feature distance

How to define the difference between two features f_1, f_2?

- Simple approach is SSD(f1, f2)
  - **Sum of square differences** between entries of the two descriptors
  - Can give good scores to very ambiguous (bad) matches



I_1



I_2

Credit: S. Szeliski

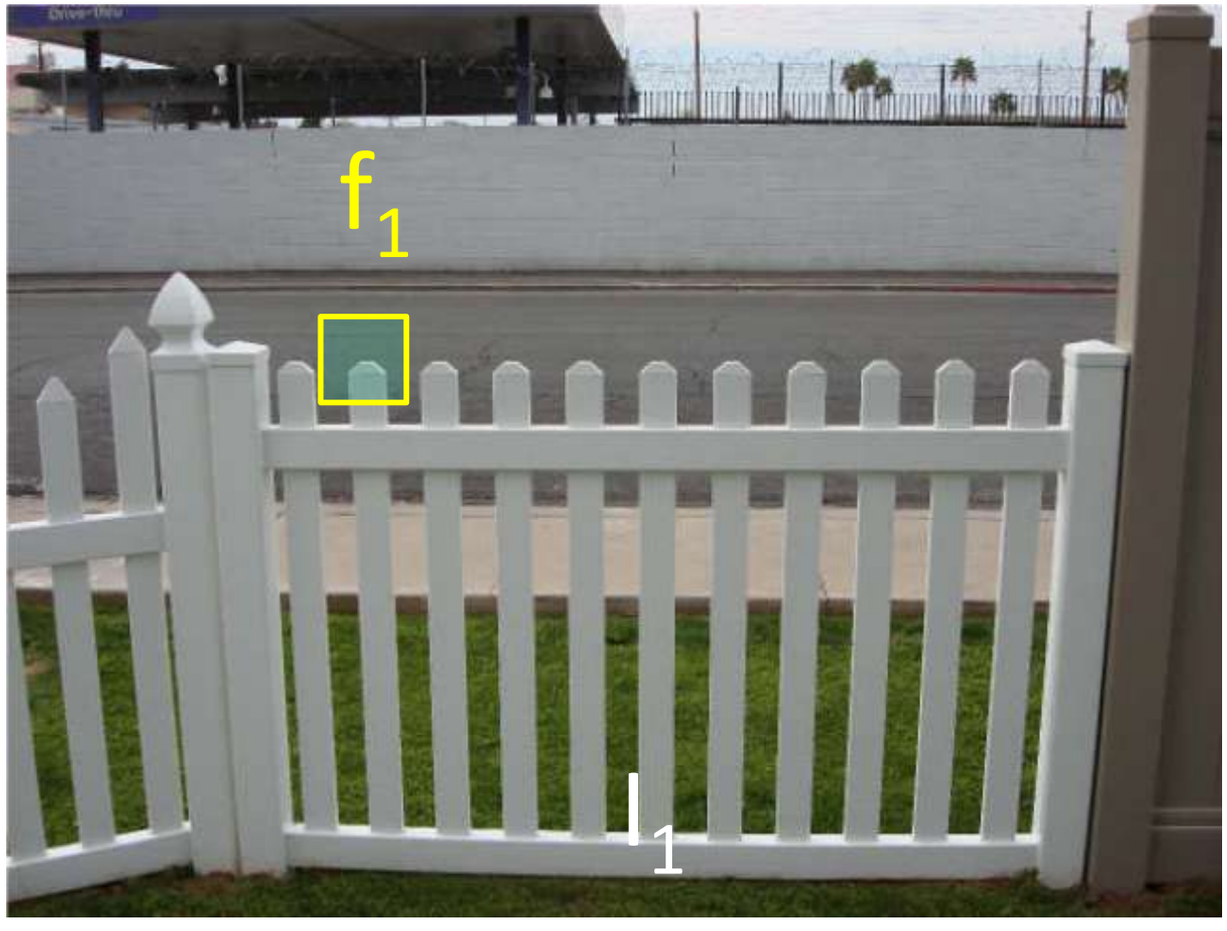
Dual threshold method:

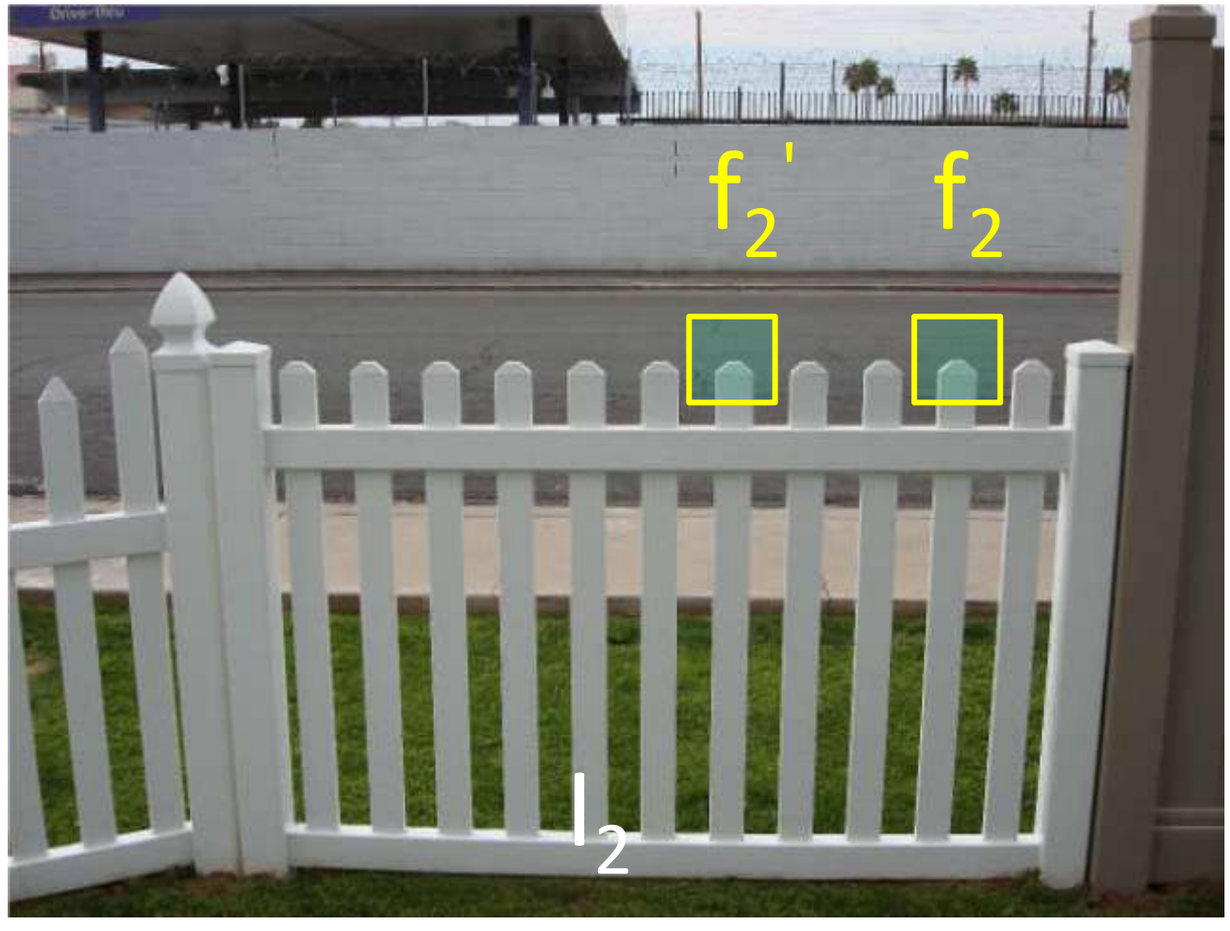
- Find the *two* nearest neighbors (Euclidean distance)
- Get their distances (d1 and d2)
  - What is the possible value range of the distances?
- Threshold the “**ratio** of nearest to second nearest descriptor”: d1/d2

Lowe IJCV 2004

Dual Threshold method

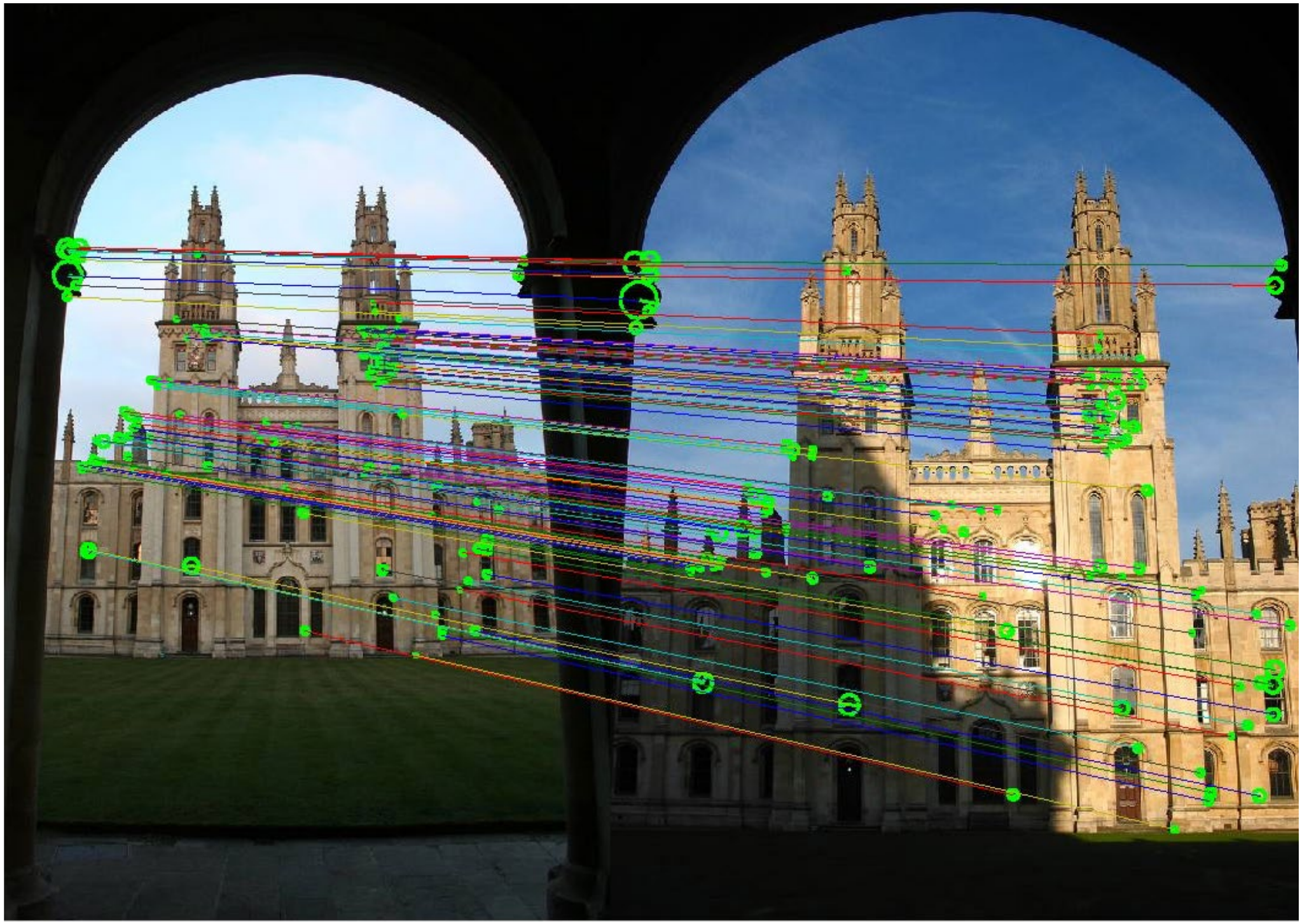
- Ratio distance = L2(f1, f2) / L2(f1, f2’)
  - f_2 is best L2 match to f1 in I_2
  - f_2’ is 2nd best L2 match to f_1 in I_2
  - gives values close to 1 for ambiguous matches (e.g. 0.11/0.12) and small values for unique matches (e.g. 0.1/0.8)



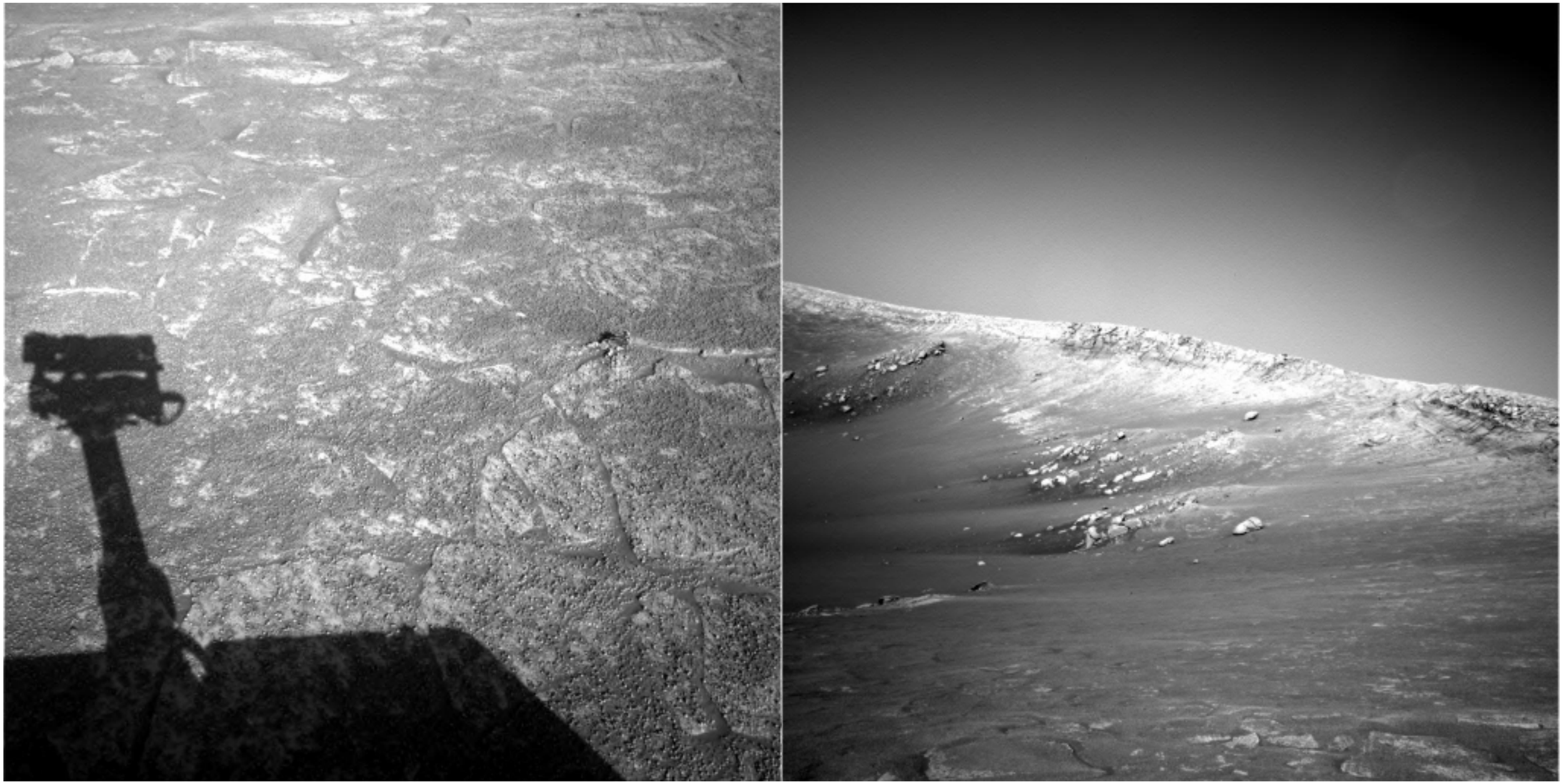


Credit: S. Szeliski

Matching Result

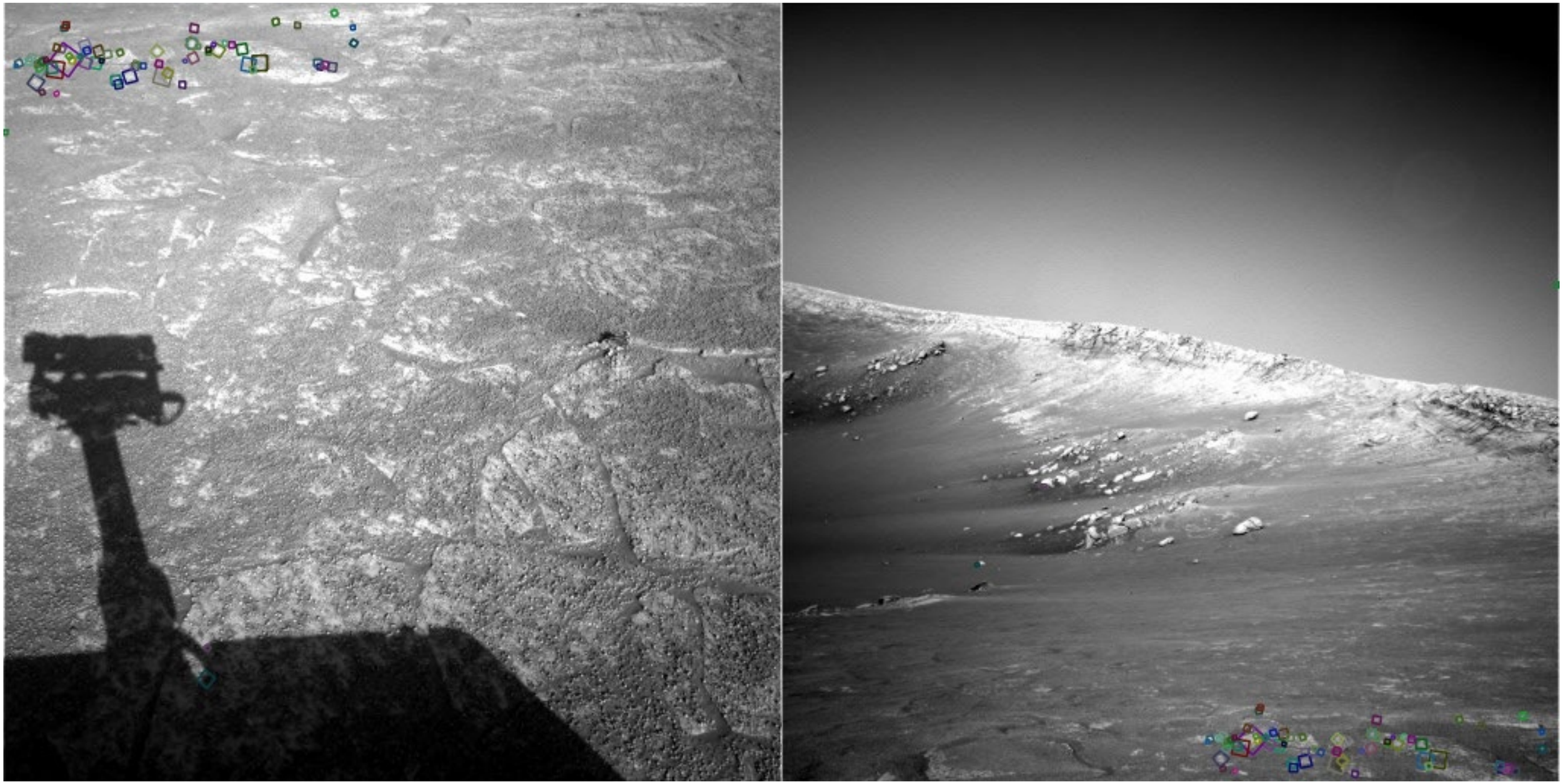


Example – a hard case..



NASA Mars Rover images

Example – a hard case..



NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Summary on Local Features

- Local features are **distinct and salient points** in an image, e.g.
  - Corners: Harris Corner Detector
  - Blobs: Laplacian of Gaussian Detector
- Feature descriptors represent the **neighborhood** of a local feature
  - Most: invariant to scale, rotation, translation, illumination
  - Some: invariant also to affine transforms
- Matching feature points = matching their descriptors
  - Distance measure: sum of squared differences (SSD) or L2 (Euclidean distance)
  - **Dual threshold approach works best in practice**
- **Note: every feature detector can be combined with every feature descriptor!**