

# 3D Reconstruction with Computer Vision

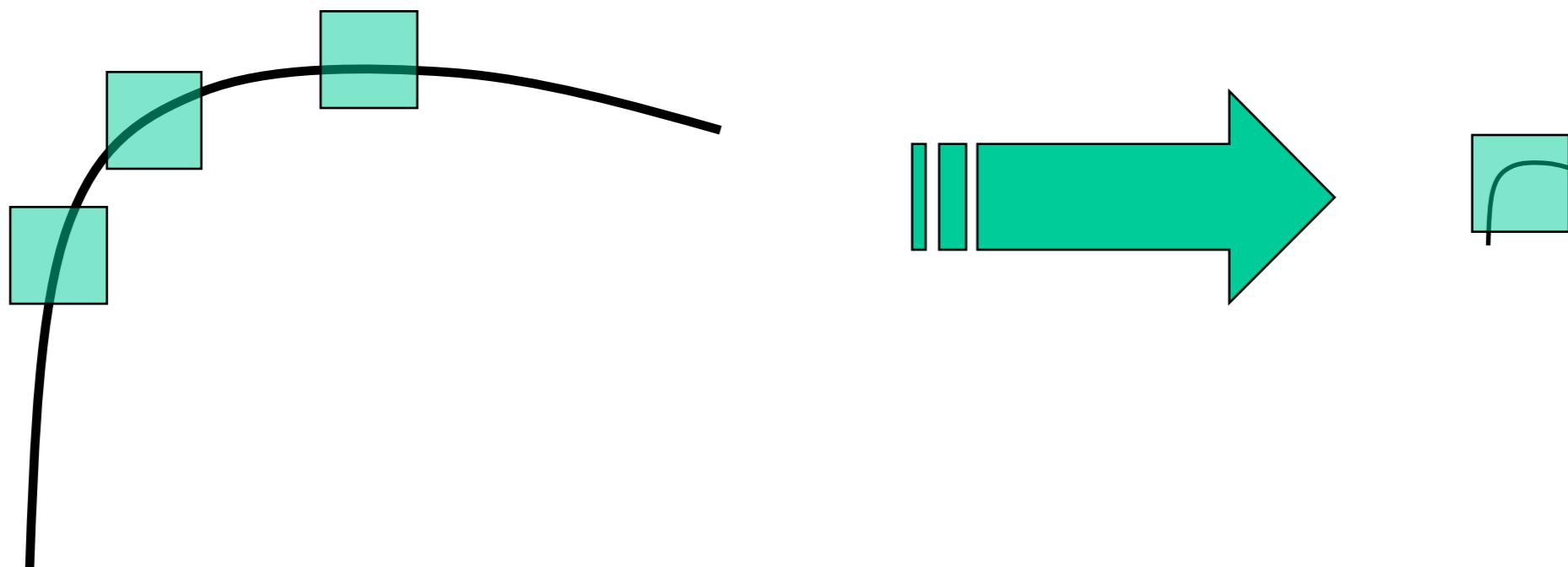
## Meeting 5: Local Features 2



# Properties of the Harris corner detector

Rotation invariant? Yes

Scale invariant? No



All points will be  
classified as **edges**

**Corner !**

# Scale invariant interest points

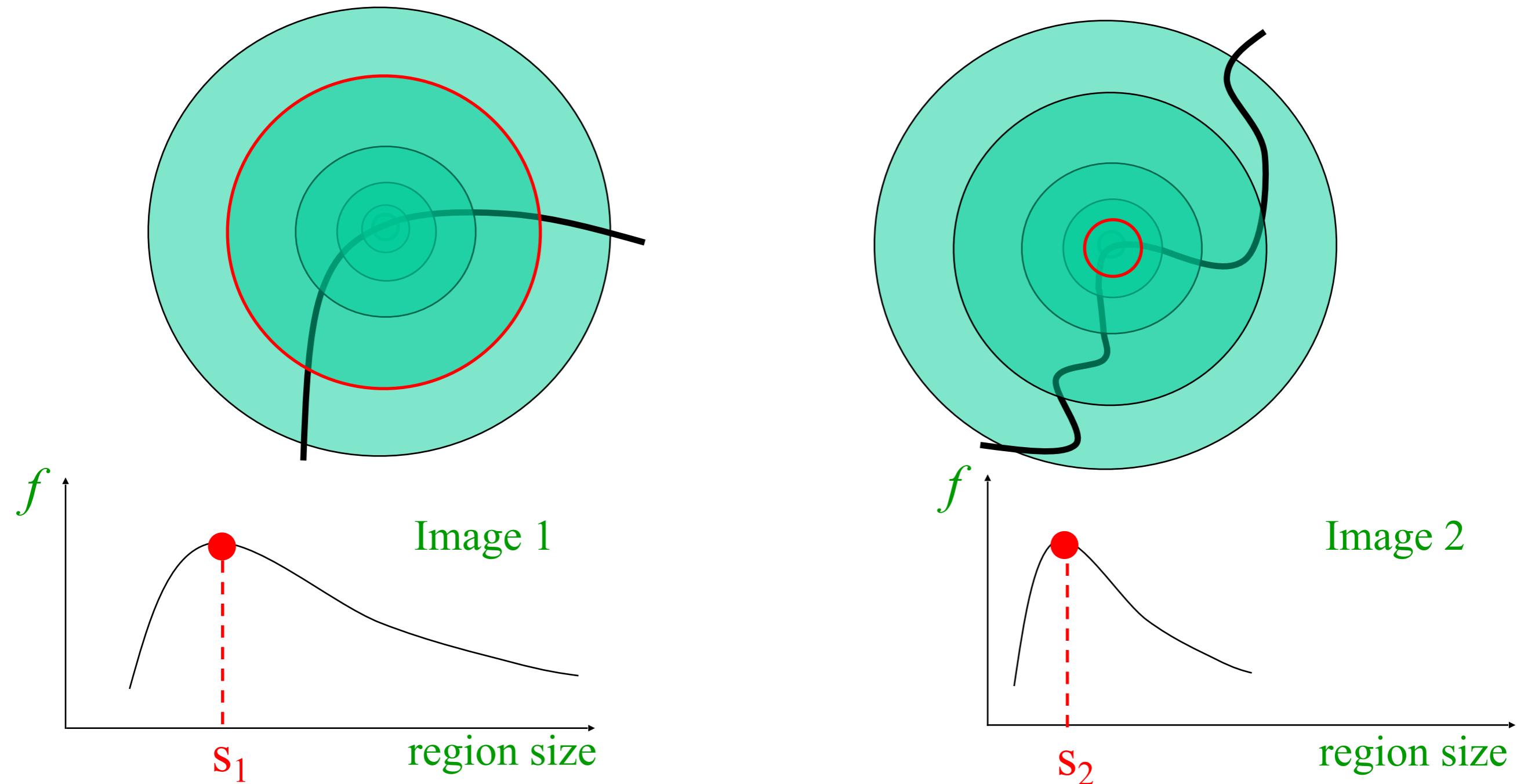
How can we independently select interest points in each image, such that the detections are repeatable across different scales?



# Automatic scale selection

## Intuition:

- Find scale that gives local maxima of some function  $f$  in both position and scale.



# Image matching

---



by [Diva Sian](#)



by [swashford](#)

# Harder case

---



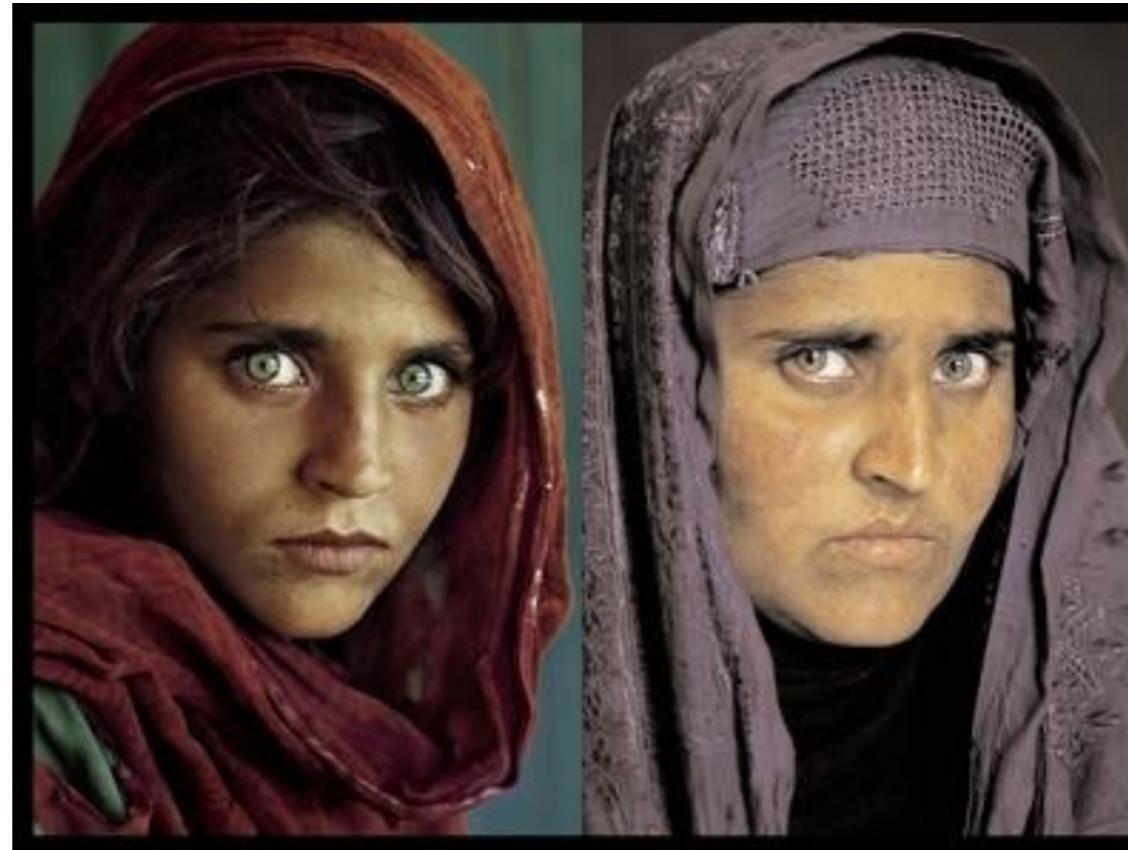
by [Diva Sian](#)



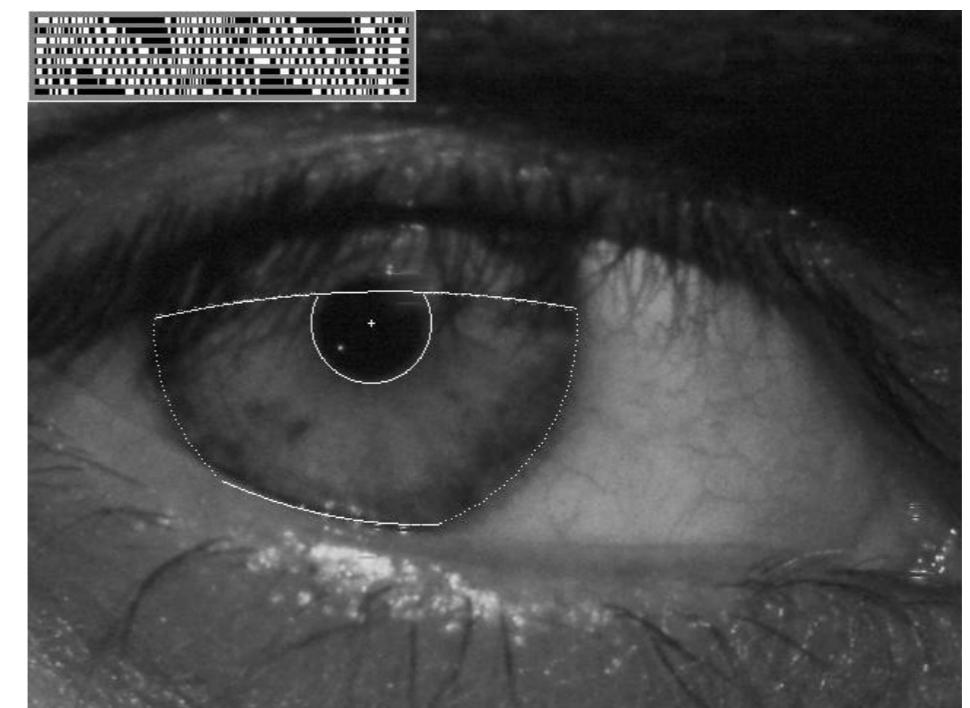
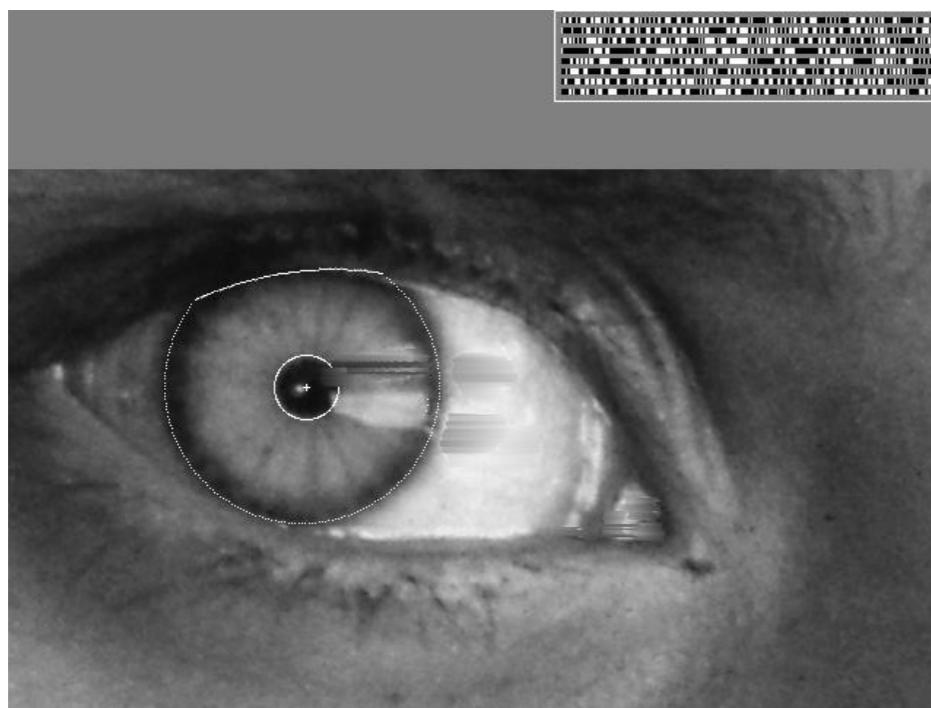
by [scgbt](#)

# Even harder case

---

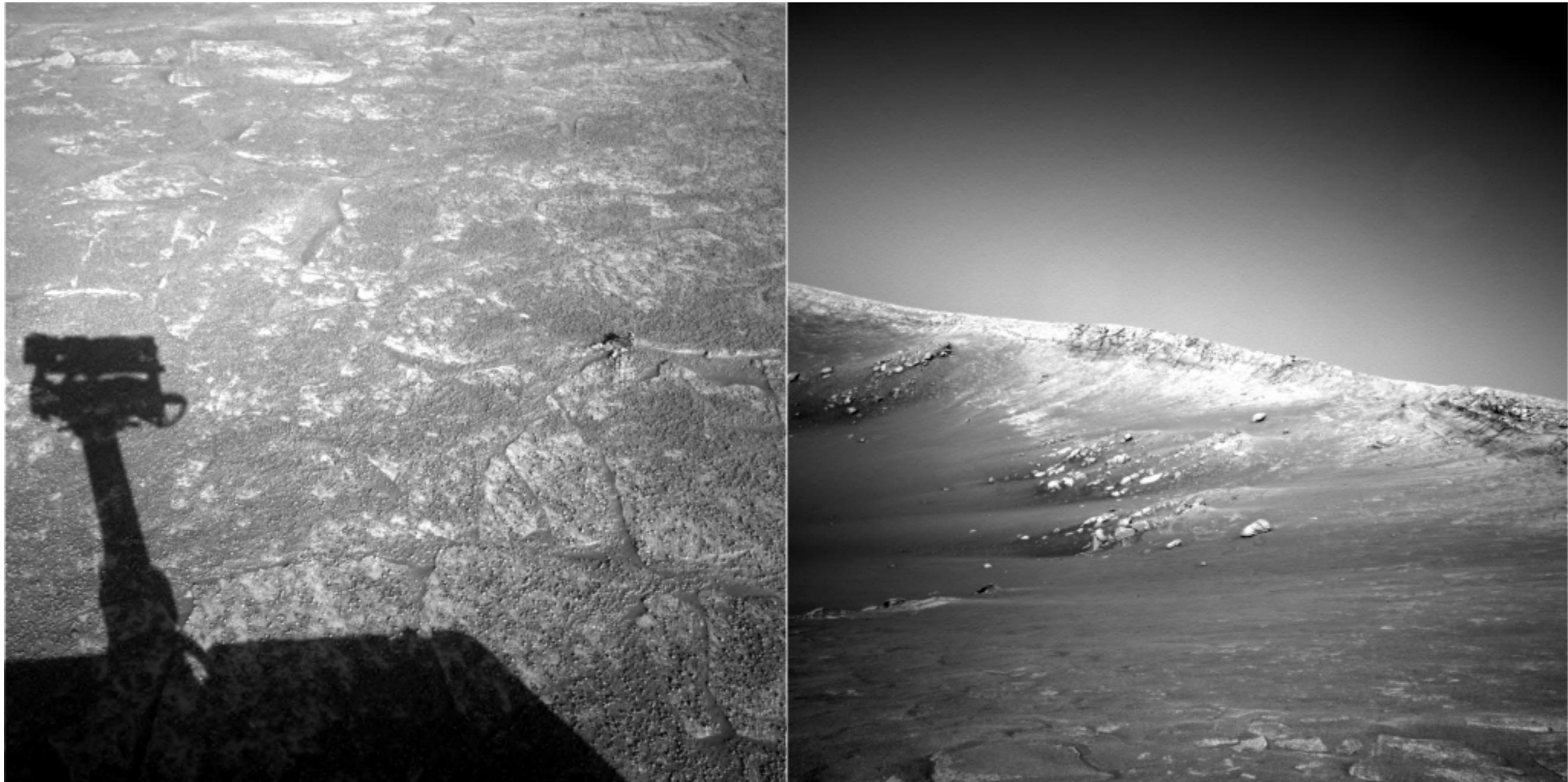


***“How the Afghan Girl was Identified by Her Iris Patterns”*** Read the [story](#)



# Harder still?

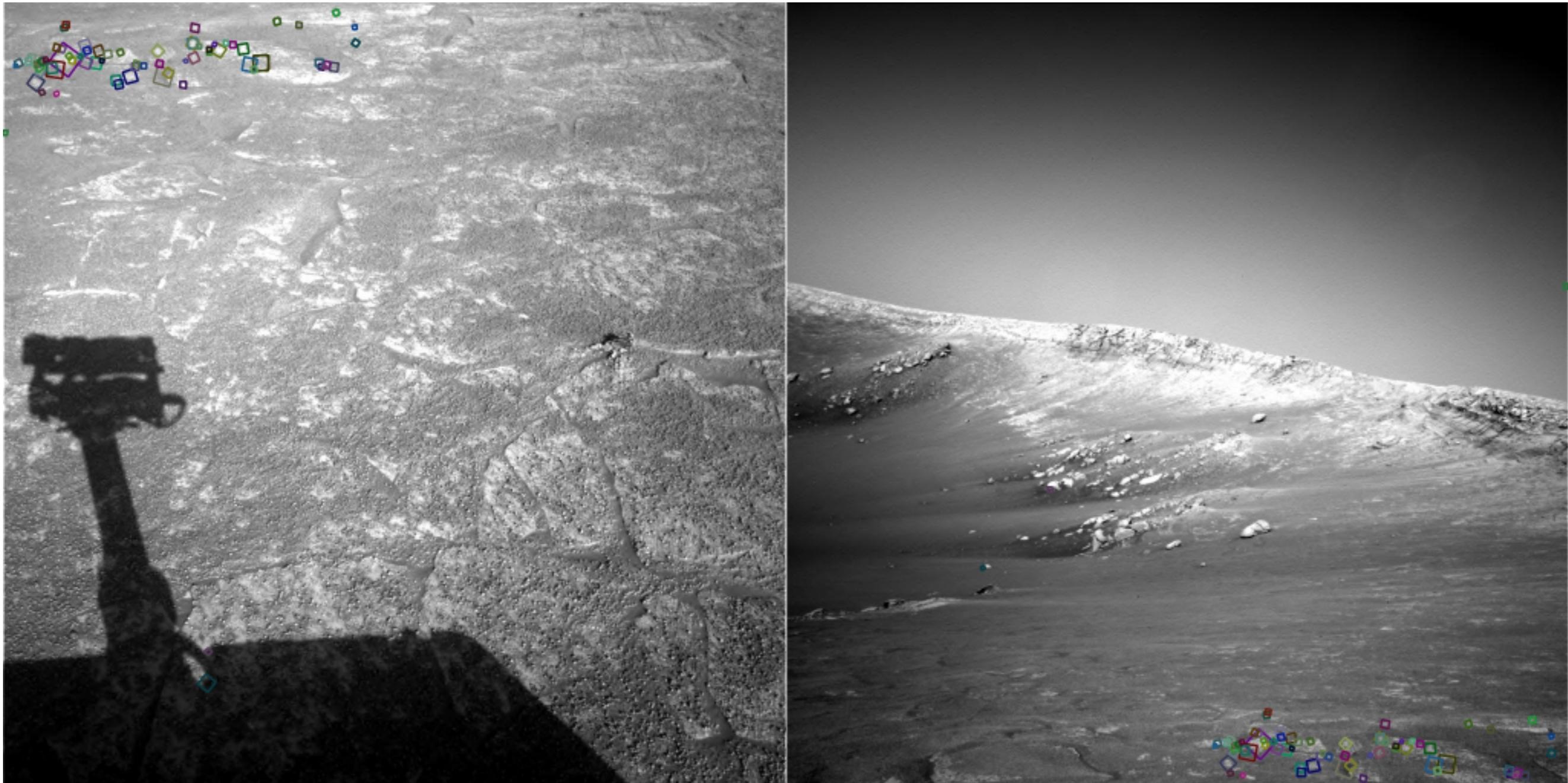
---



NASA Mars Rover images

# Answer below (look for tiny colored squares...)

---



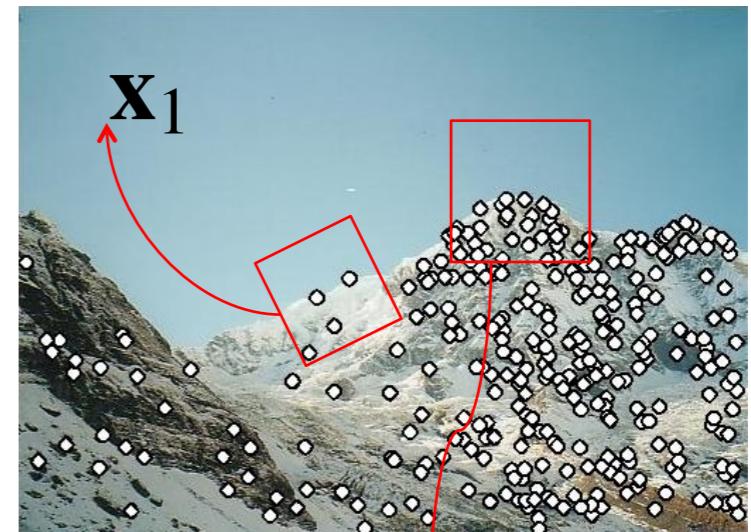
NASA Mars Rover images  
with SIFT feature matches  
Figure by Noah Snavely

# Local features: main components

1) Detection: Identify the interest points

2) Description: Extract vector feature descriptor surrounding each interest point.

3) Matching: Determine correspondence between descriptors in two views

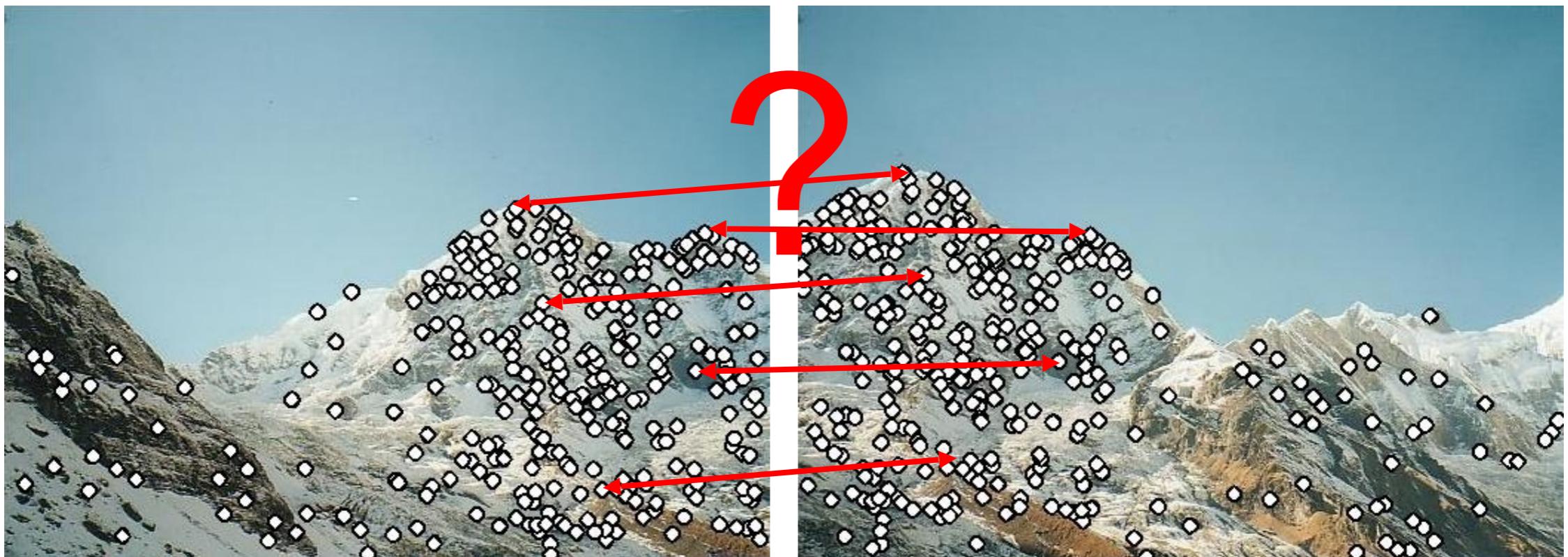


# Feature descriptors

---

We know how to detect good points

Next question: **How to match them?**

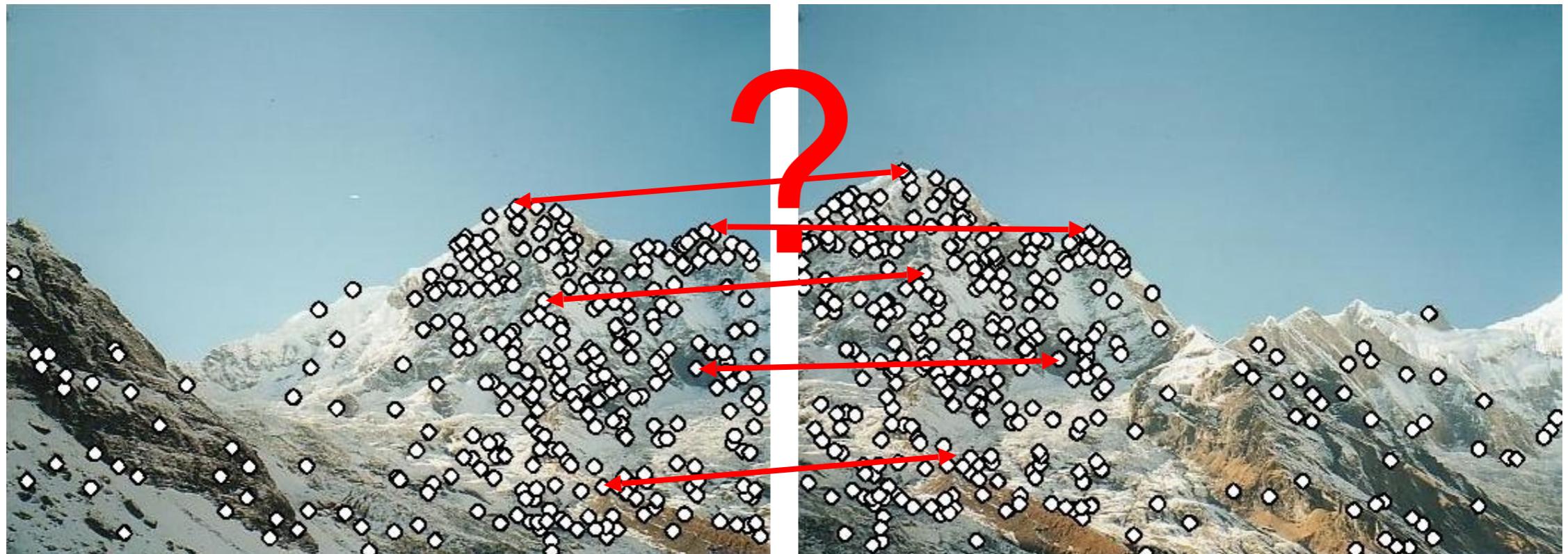


# Feature descriptors

---

We know how to detect good points

Next question: **How to match them?**



Lots of possibilities (this is a popular research area)

- Simple option: match square windows around the point
- State of the art approach: SIFT
  - David Lowe, UBC <http://www.cs.ubc.ca/~lowe/keypoints/>

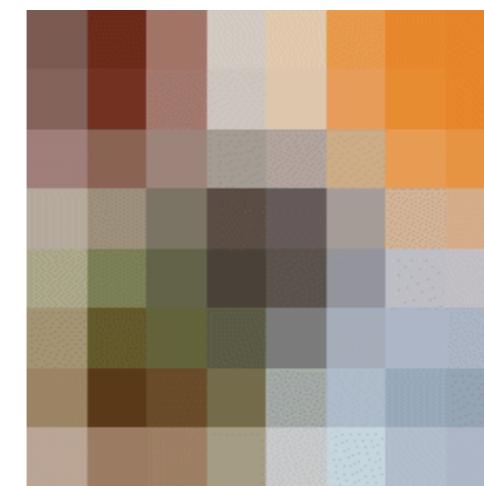
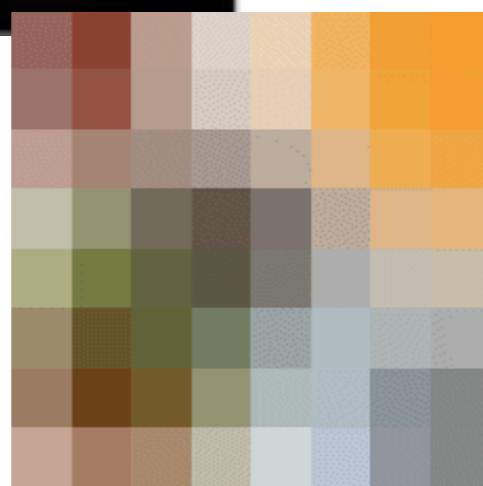
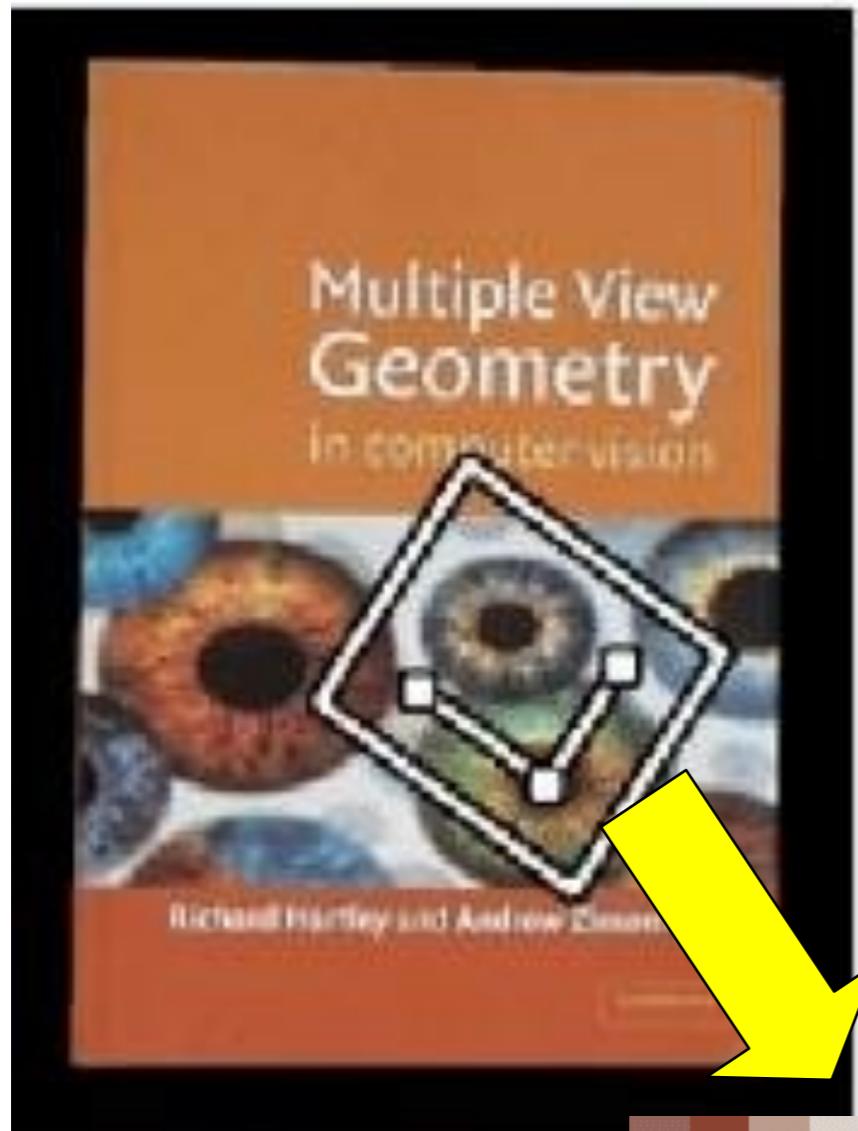
# Invariance

---

Suppose we are comparing two images  $I_1$  and  $I_2$

- $I_2$  may be a transformed version of  $I_1$
- What kinds of transformations are we likely to encounter in practice?

# Invariant to: geometric transformations



e.g. scale,  
translation,  
rotation

# Invariant to: photometric transformations

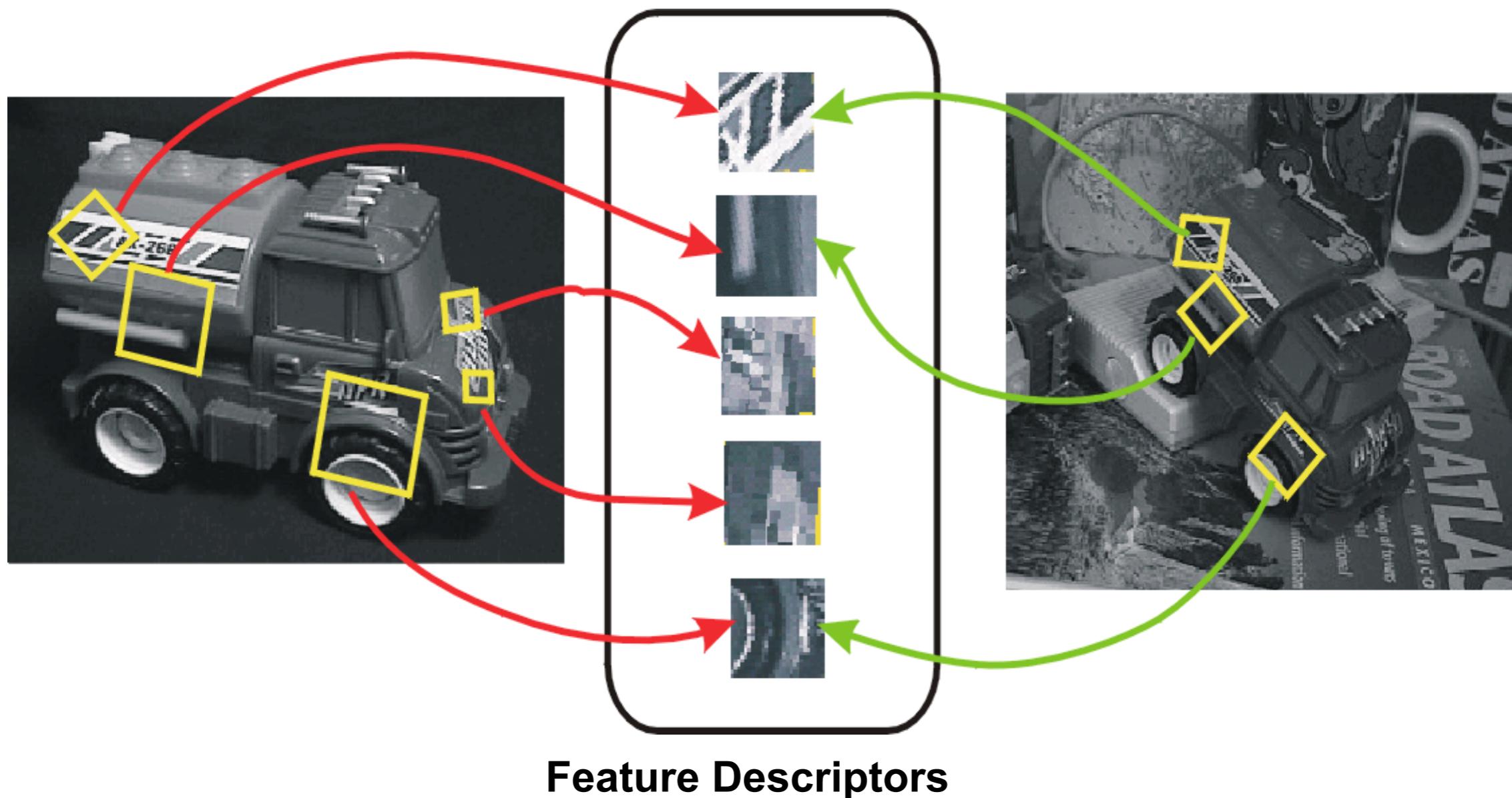


Figure from T. Tuytelaars ECCV 2006 tutorial

# Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, ...



# Invariance

---

Suppose we are comparing two images  $I_1$  and  $I_2$

- $I_2$  may be a transformed version of  $I_1$
- What kinds of transformations are we likely to encounter in practice?

We'd like to find the same features regardless of the transformation

- This is called transformational ***invariance***
- Most feature methods are designed to be invariant to
  - Translation, 2D rotation, scale
- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 60 degrees)
  - Limited affine transformations (some are fully affine invariant)
  - Limited illumination/contrast changes

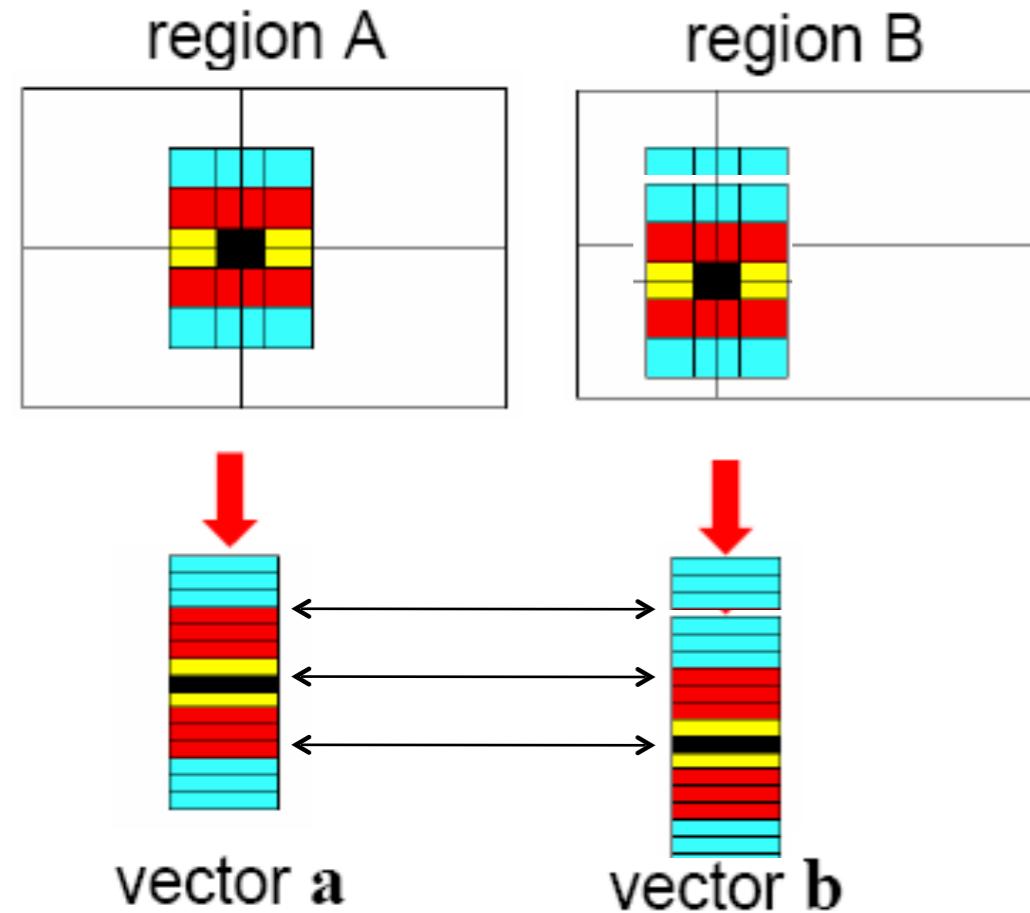
# How to achieve invariance

---

Need both of the following:

1. Make sure your detector is invariant
  - Harris is invariant to translation and rotation
  - Scale is trickier
    - SIFT uses automatic scale selection (previous slides)
    - simpler approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS) and add them all to database
2. Design an invariant feature *descriptor*
  - A descriptor captures the information in a region around the detected feature point
  - The simplest descriptor: a square window of pixels
    - What's this invariant to?
  - Let's look at some better approaches...

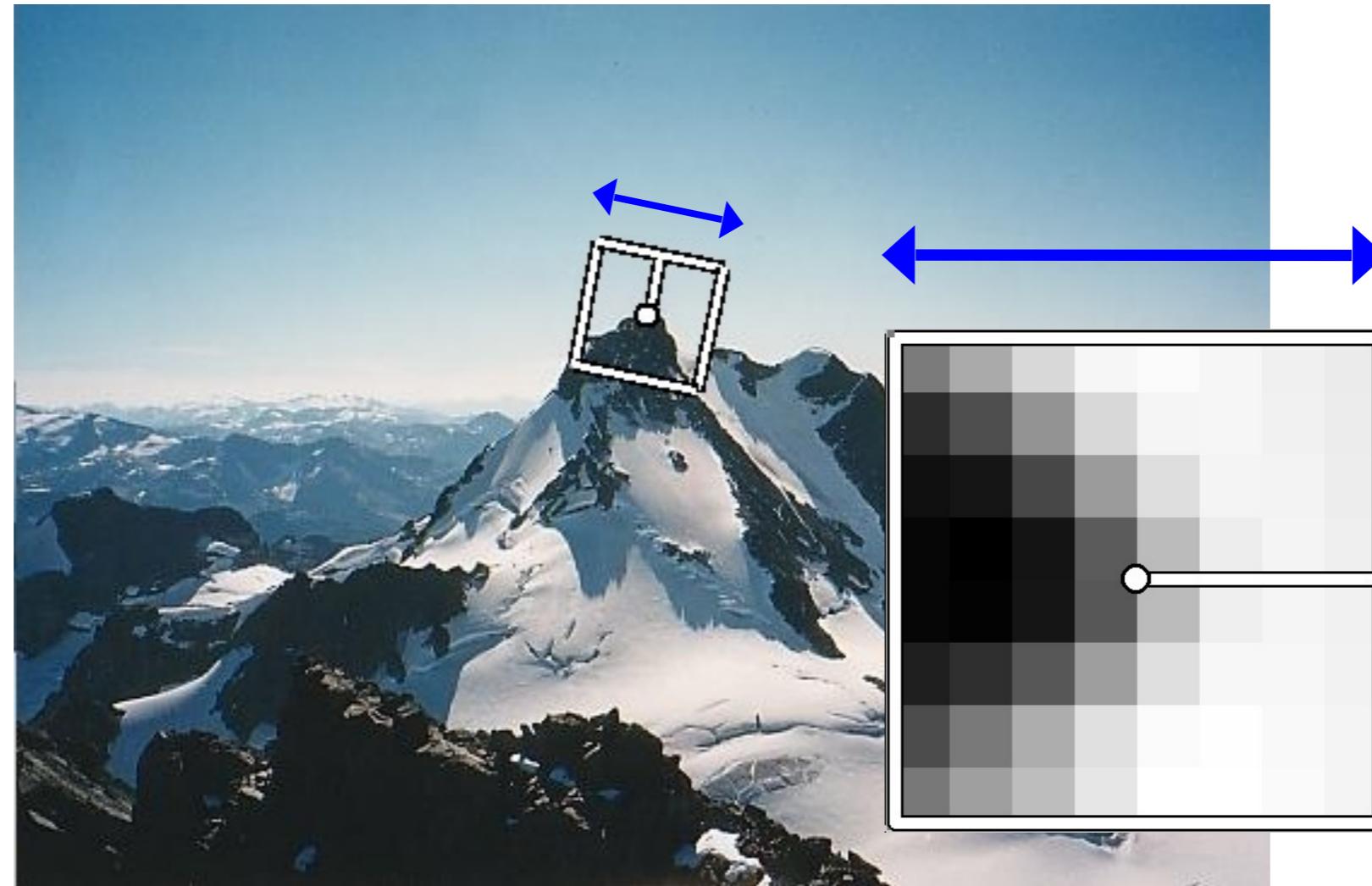
# Are raw patches good descriptors?



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.

# Making descriptor rotation invariant



- Rotate patch according to its dominant gradient orientation
- This puts the patches into a canonical orientation.

# Rotation invariance for feature descriptors

---

Find dominant orientation of the image window

- This is given by  $\mathbf{x}_+$ , the eigenvector of  $\mathbf{H}$  corresponding to  $\lambda_+$ 
  - $\lambda_+$  is the *larger* eigenvalue
- Rotate the window according to this angle

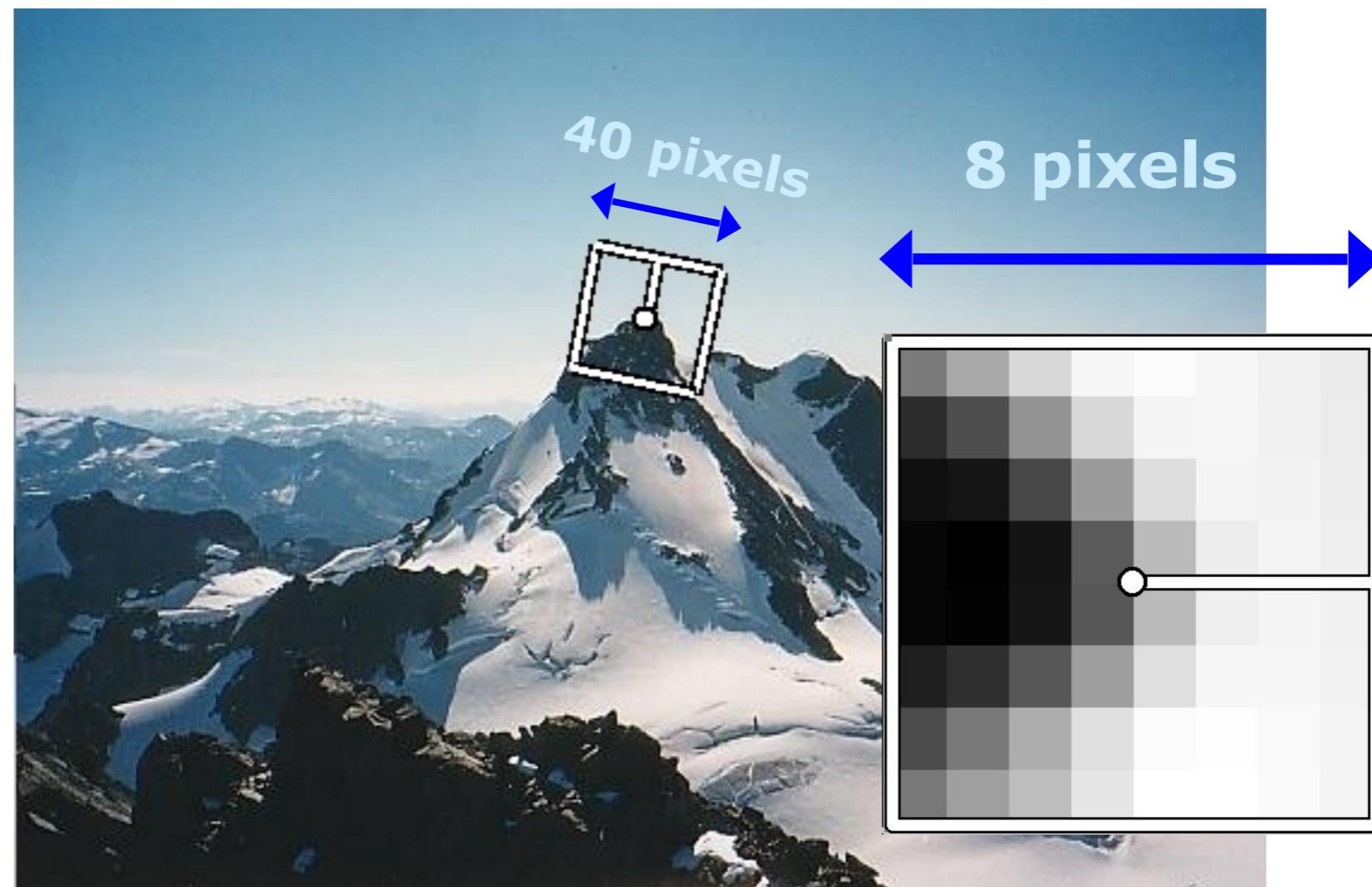


Figure by Matthew Brown

# Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

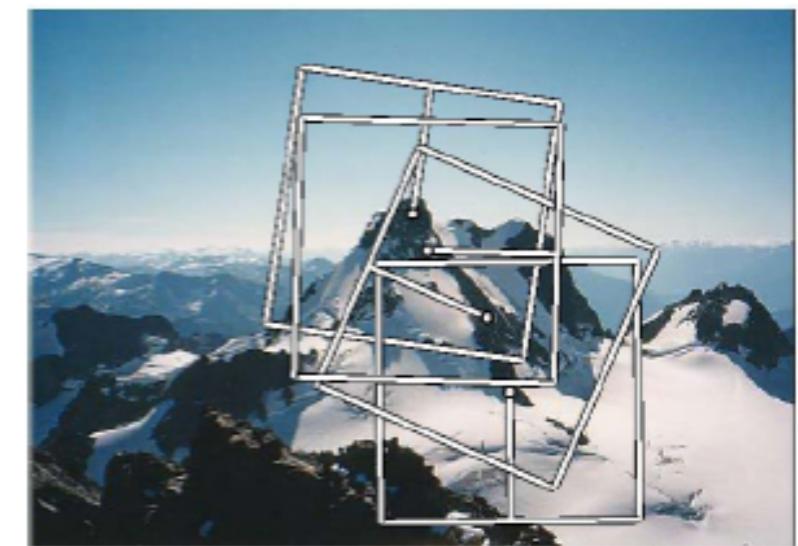
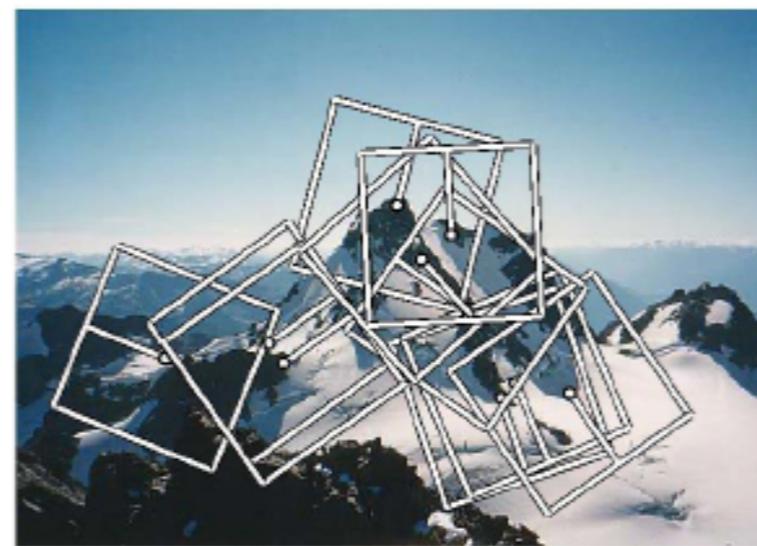
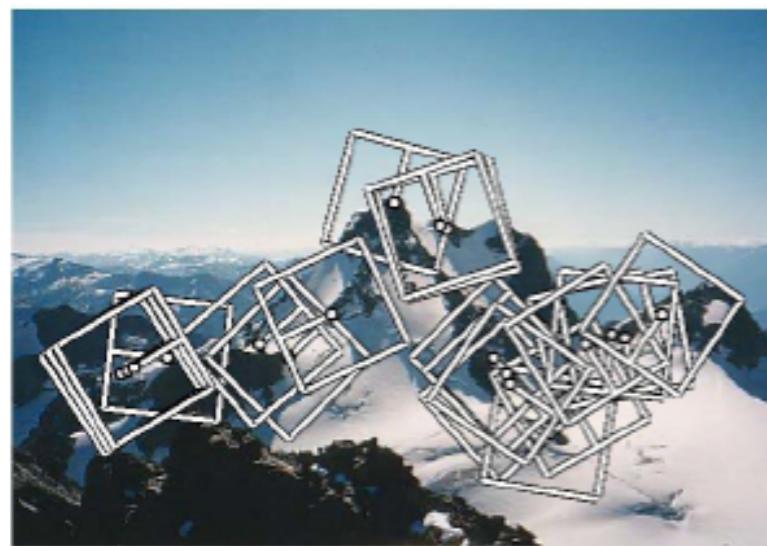
- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



Adapted from slide by Matthew Brown

# Detections at multiple scales

---

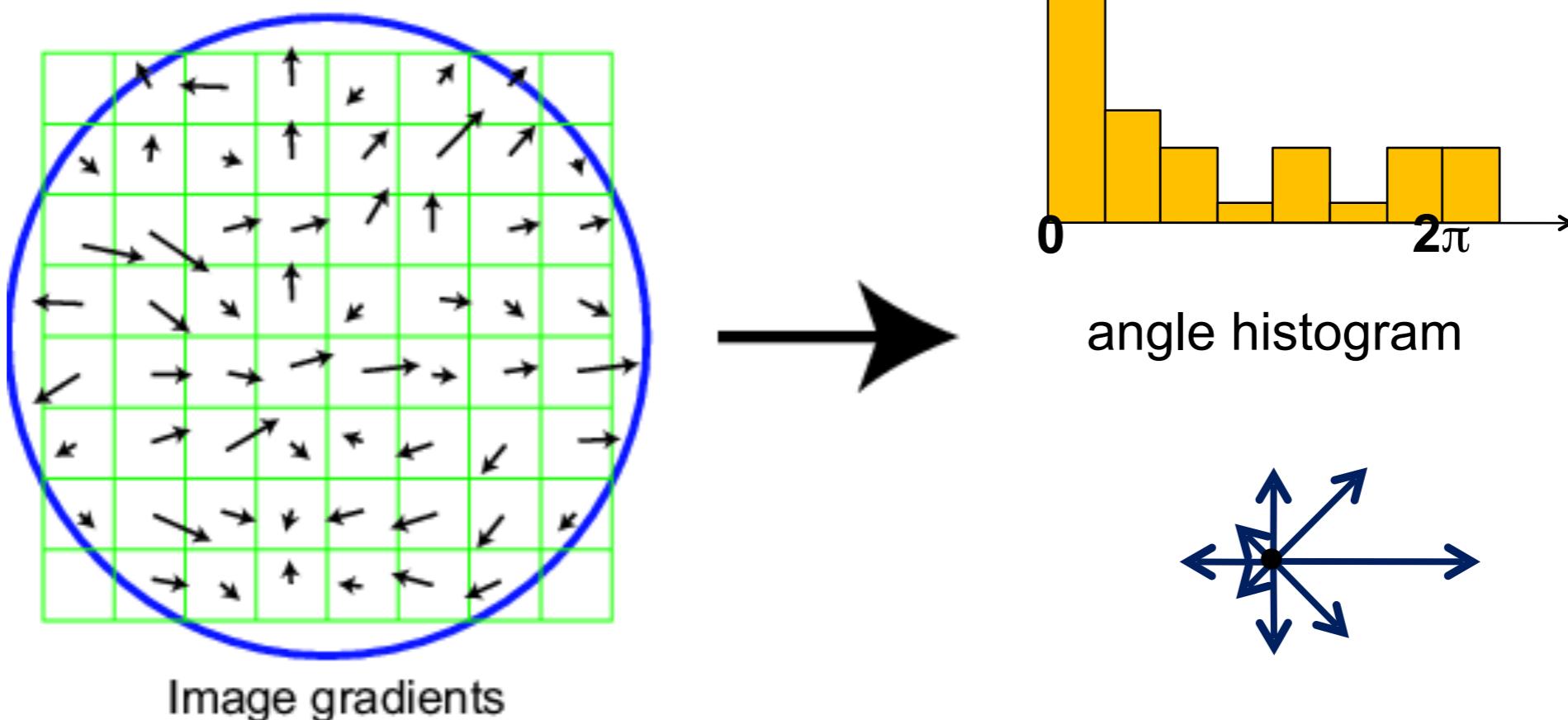


*Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.*

# Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

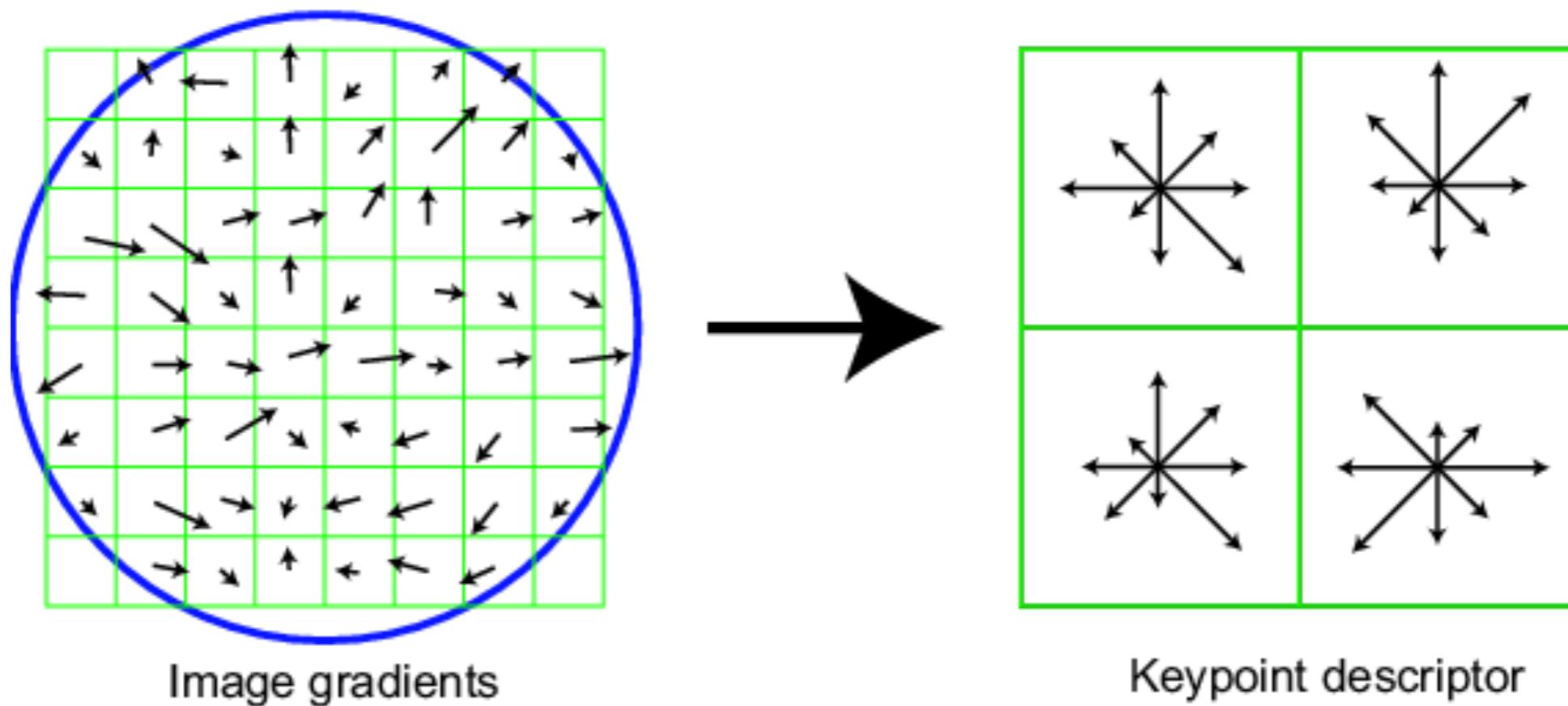


Adapted from slide by David Lowe

# SIFT descriptor

## Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells \* 8 orientations = 128 dimensional descriptor



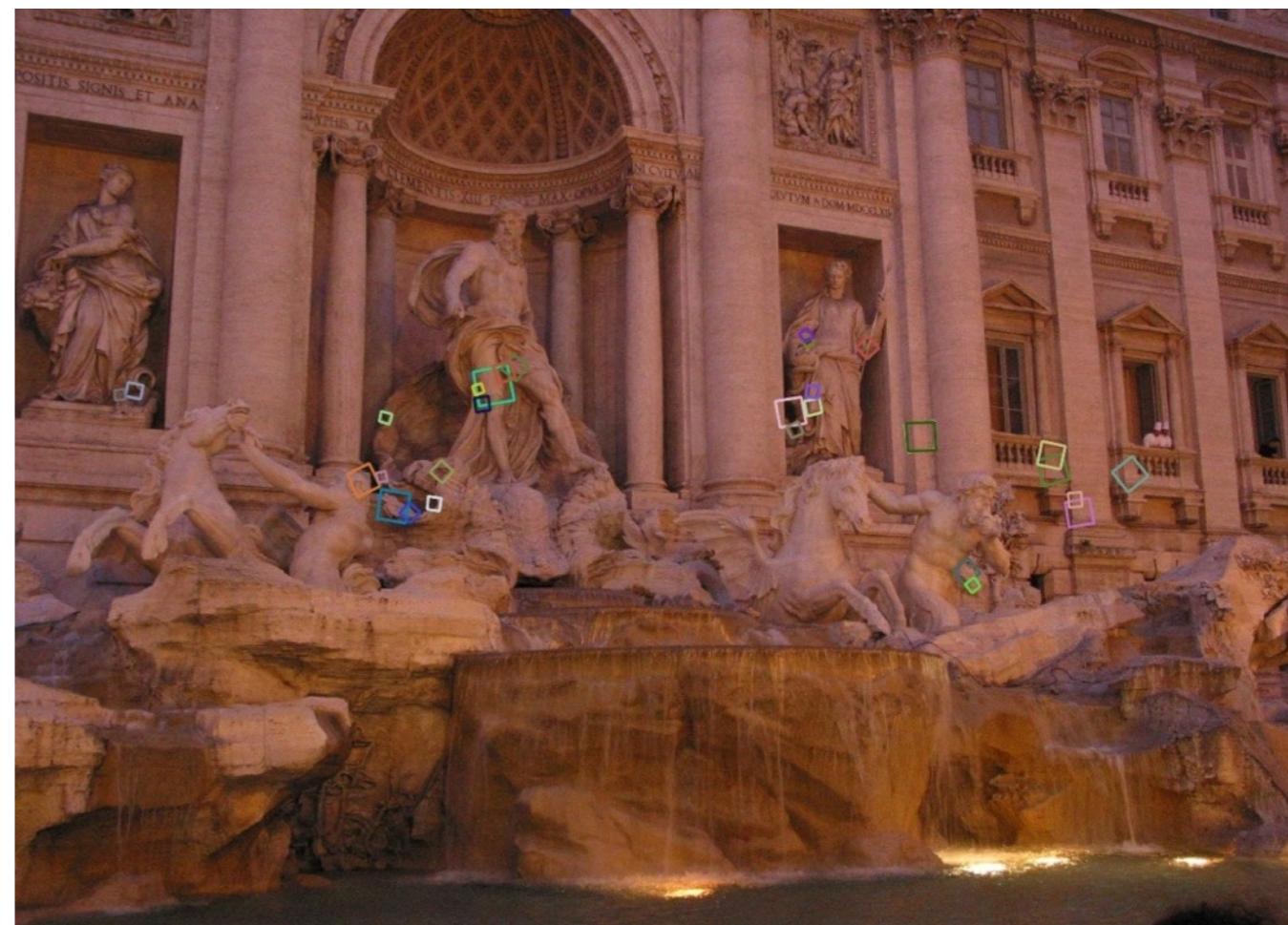
Adapted from slide by David Lowe

# Properties of SIFT

---

Extraordinarily robust matching technique

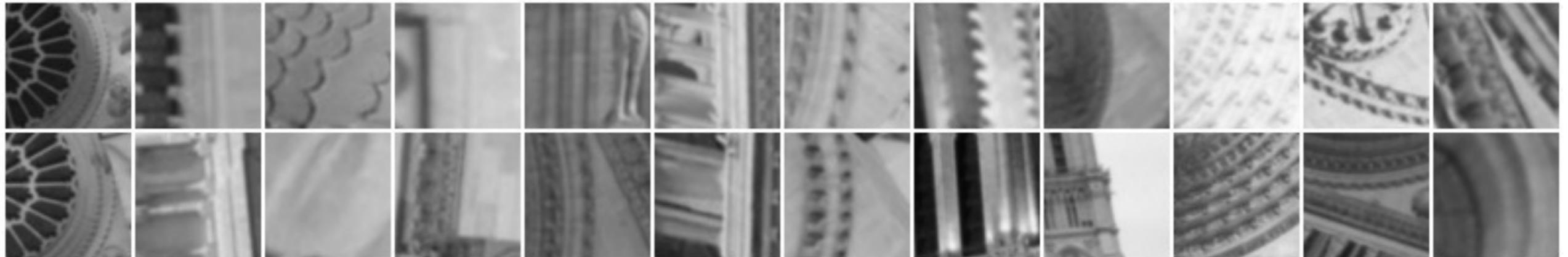
- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
  - [http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known\\_implementations\\_of\\_SIFT](http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT)



# When does SIFT fail?

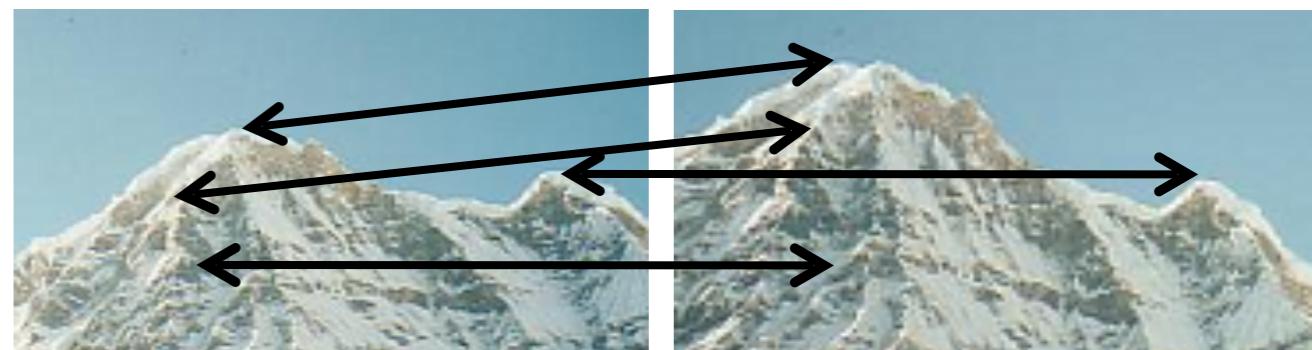
---

Patches SIFT thought were the same but aren't:



# Local features: main components

- 1) Detection: Identify the interest points
- 2) Description: Extract vector feature descriptor surrounding each interest point.
- 3) Matching: Determine correspondence between descriptors in two views



# Feature matching

---

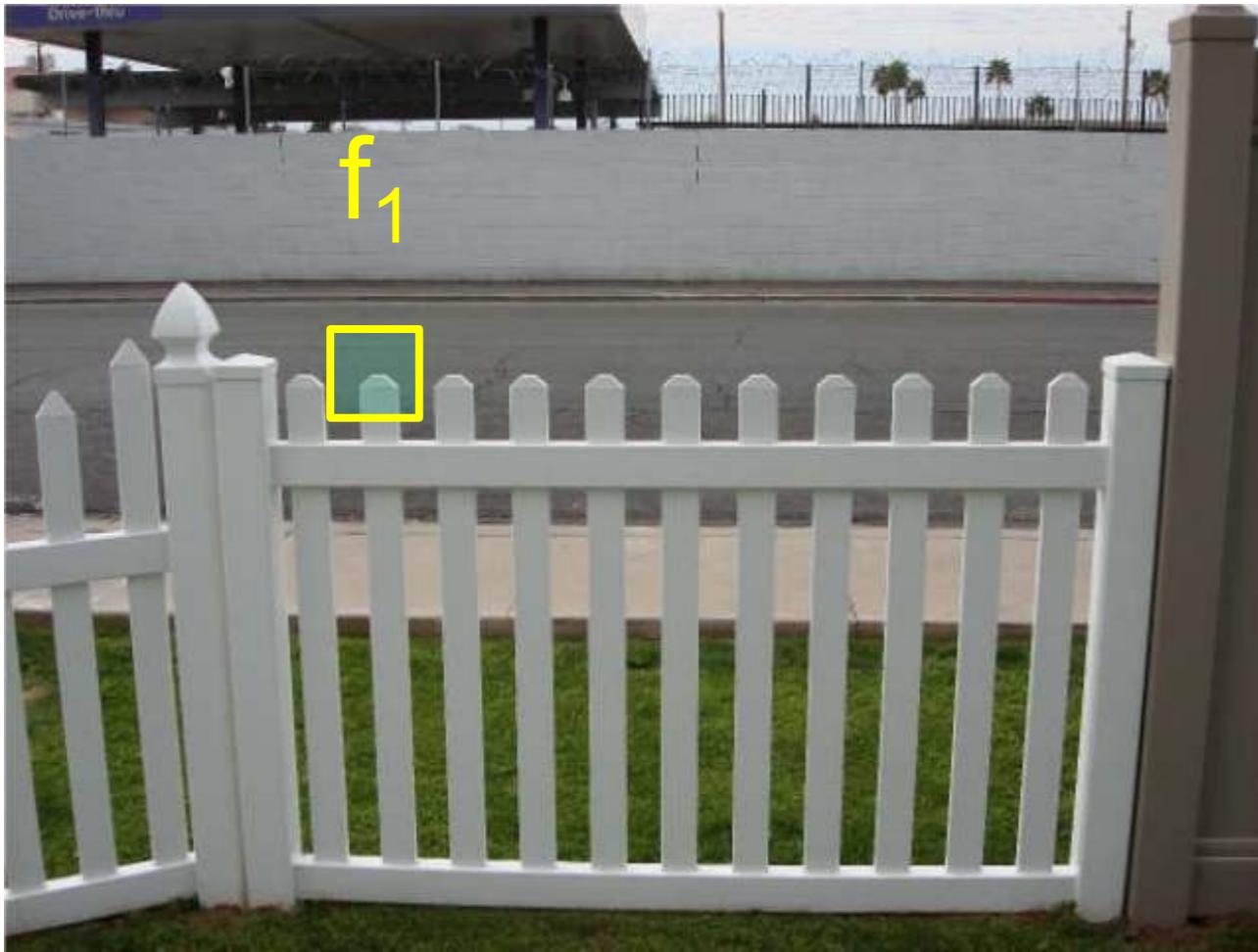
Given a feature in  $I_1$ , how to find the best match in  $I_2$ ?

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

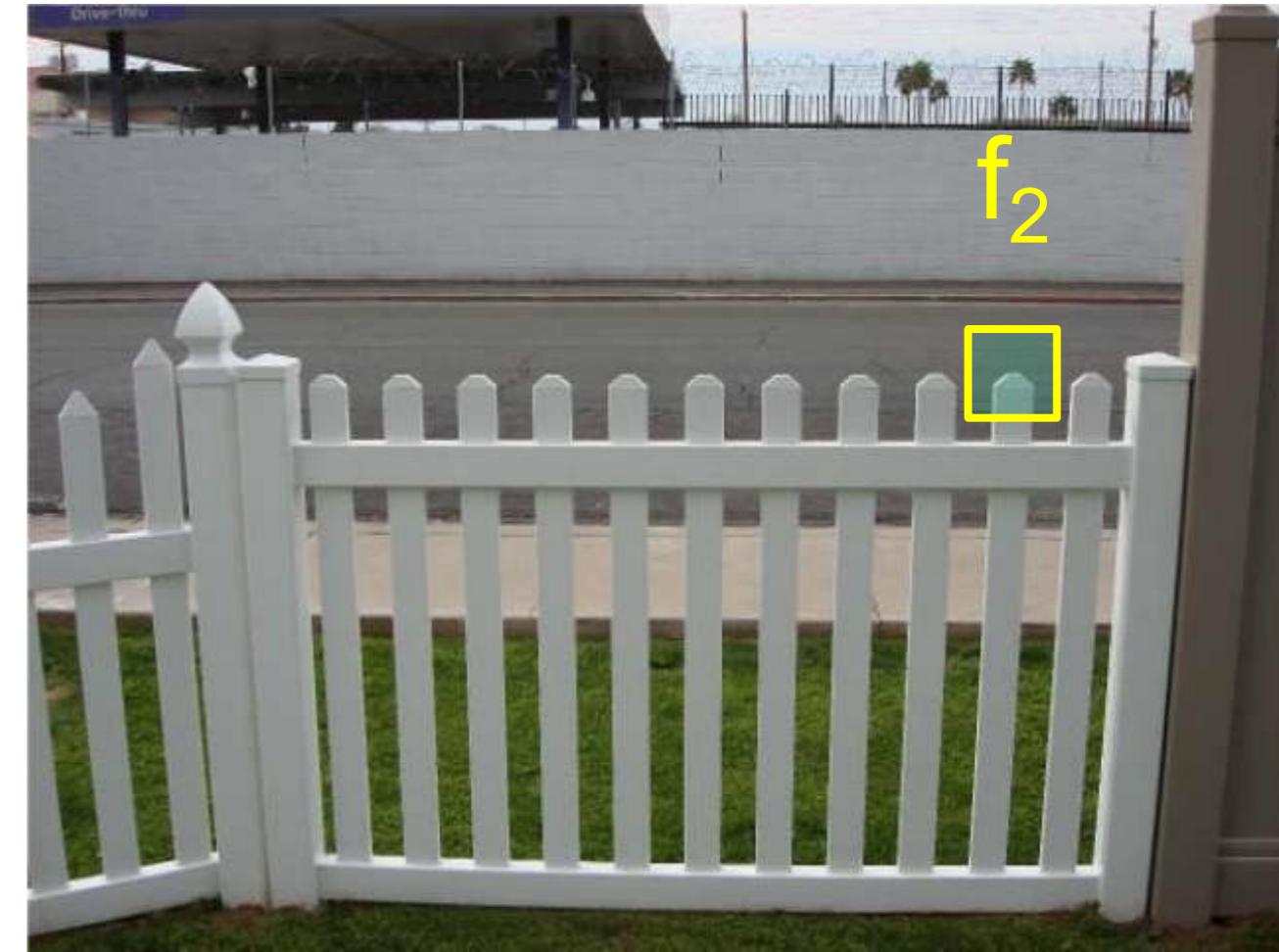
# Feature distance

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

- Simple approach is  $\text{SSD}(f_1, f_2)$ 
  - sum of square differences between entries of the two descriptors
  - can give good scores to very ambiguous (bad) matches



|  
1

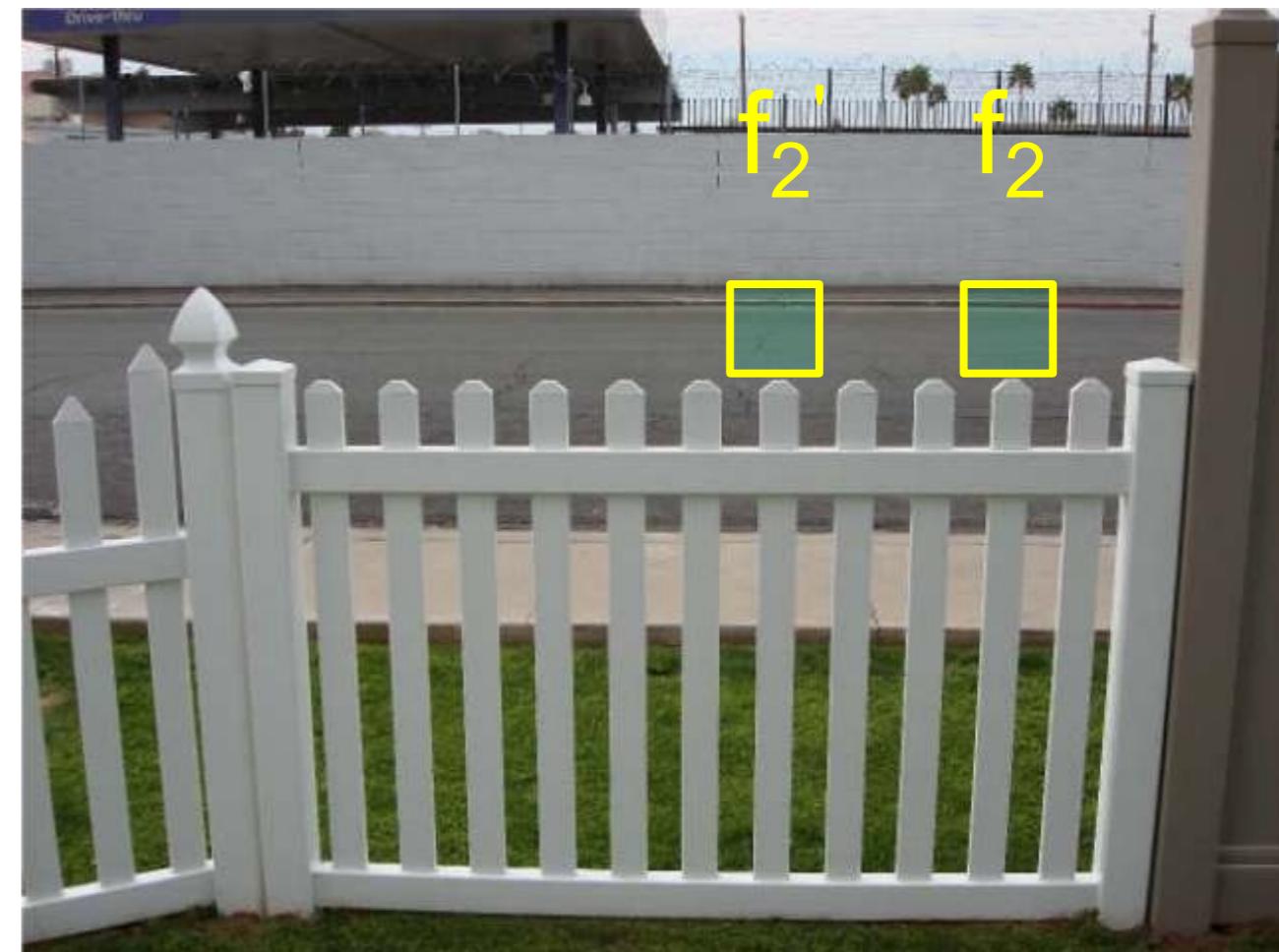
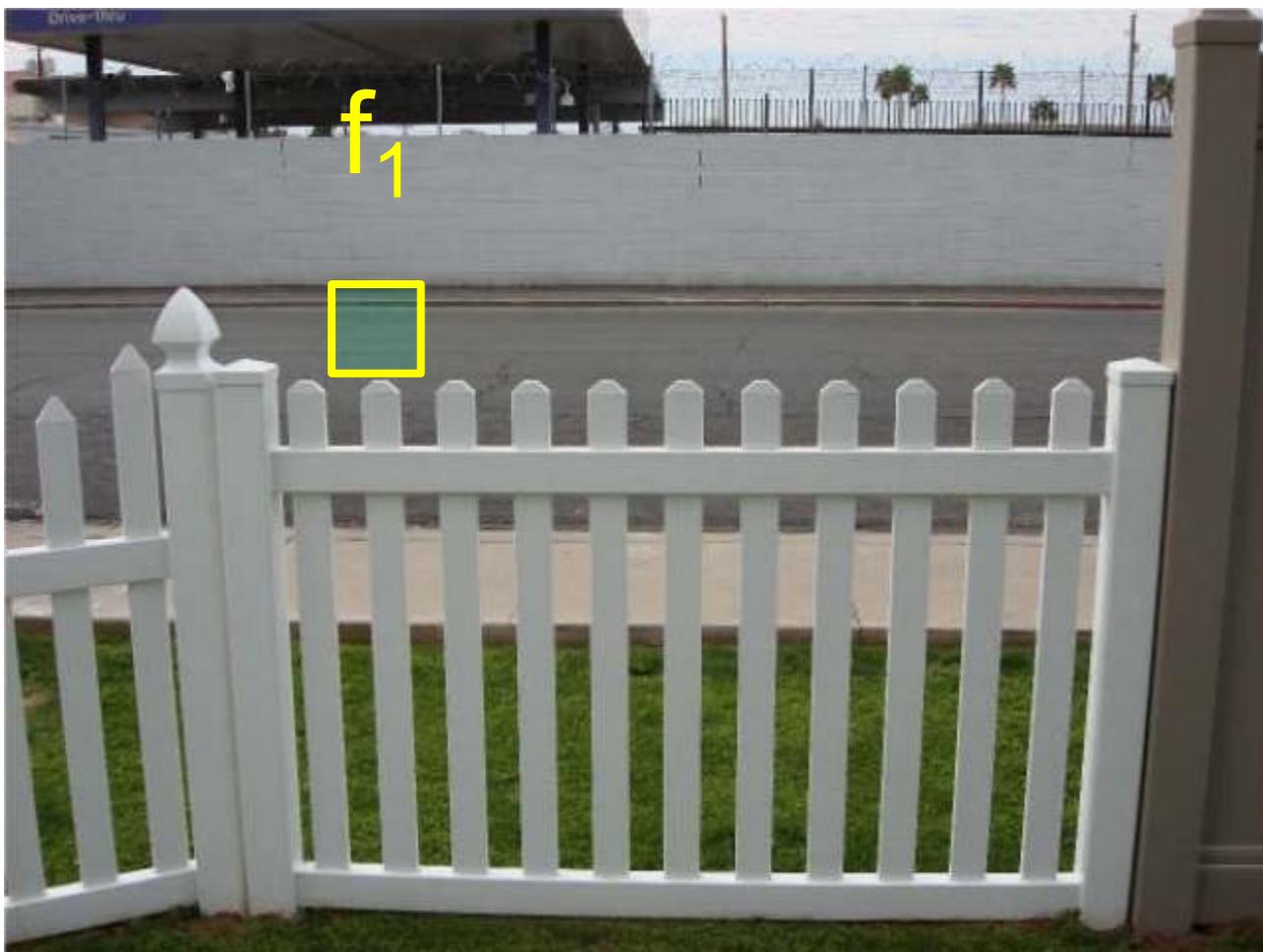


|  
2

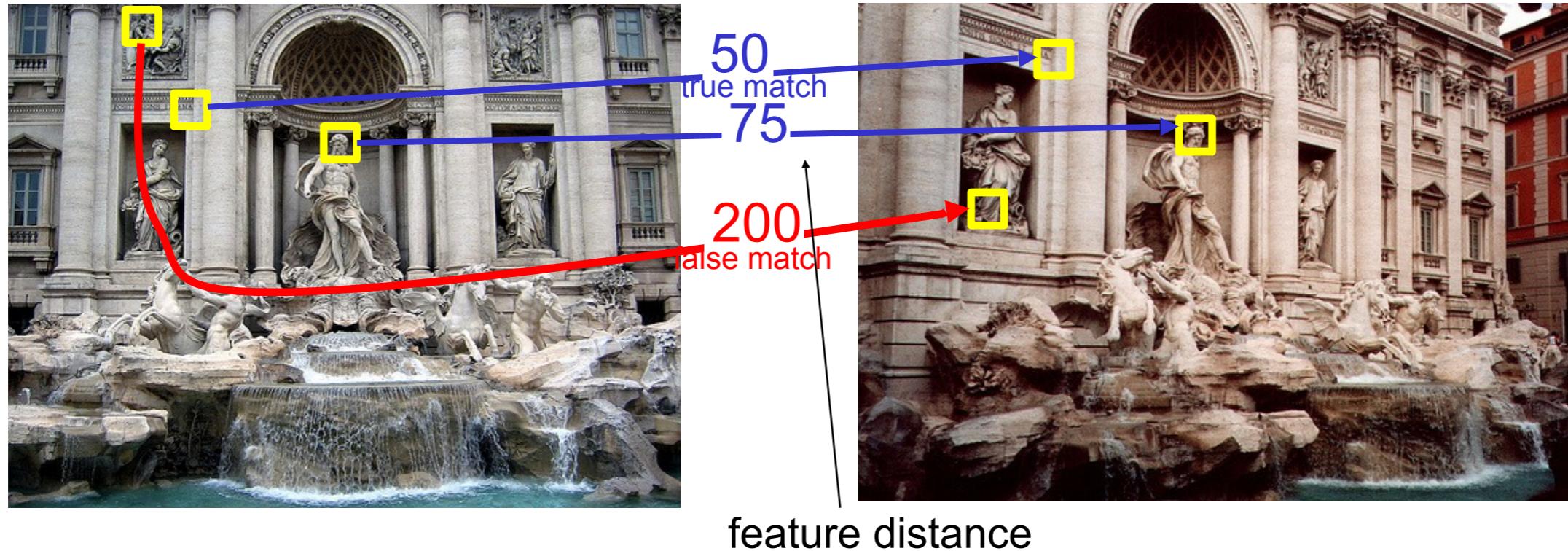
# Feature distance

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

- Better approach: ratio distance =  $\text{SSD}(f_1, f_2) / \text{SSD}(f_1, f_2')$ 
  - $f_2$  is best SSD match to  $f_1$  in  $I_2$
  - $f_2'$  is 2<sup>nd</sup> best SSD match to  $f_1$  in  $I_2$
  - gives large values for ambiguous matches
  - “Lowe ratio”



# Eliminating bad matches

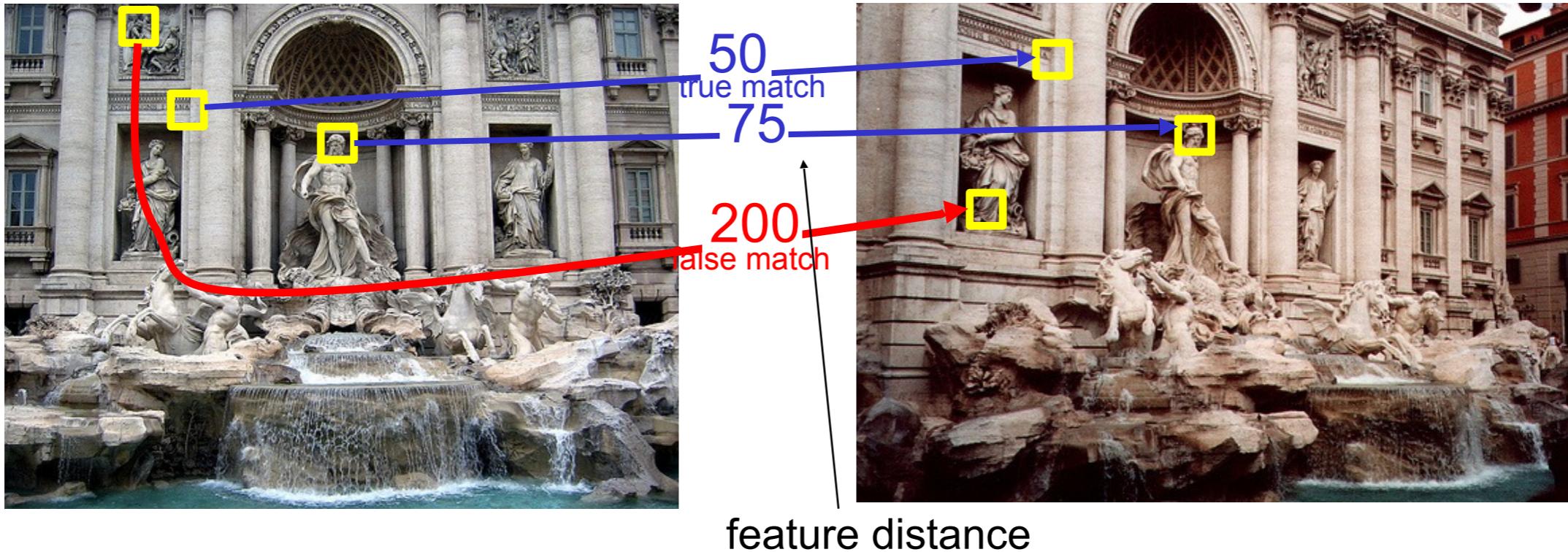


Throw out features with  $\text{distance} > \text{threshold}$

- How to choose the threshold?

# True/false positives

---



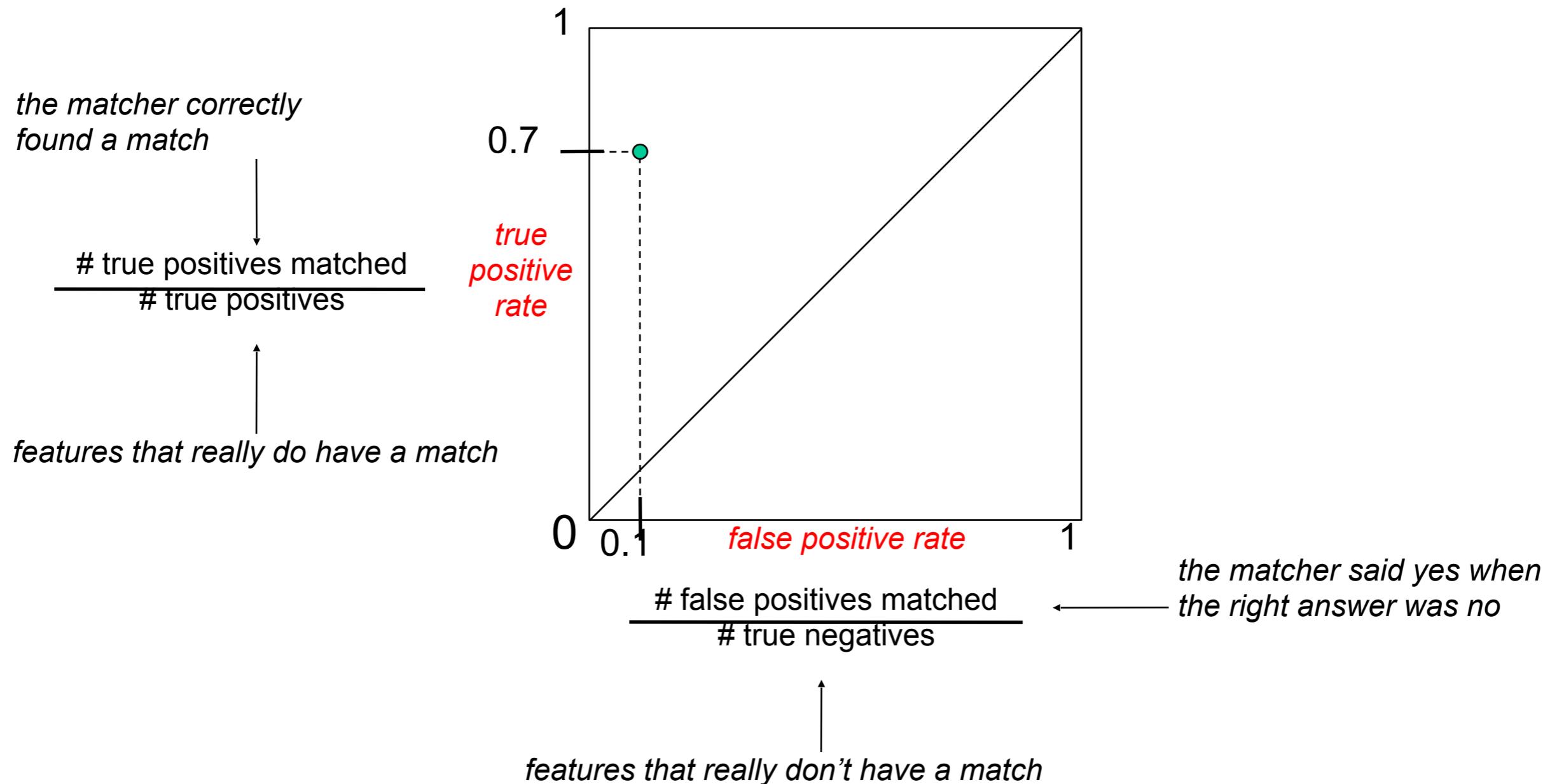
Throw out features with  $\text{distance} > \text{threshold}$

The threshold affects performance

- **True positives** = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- **False positives** = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

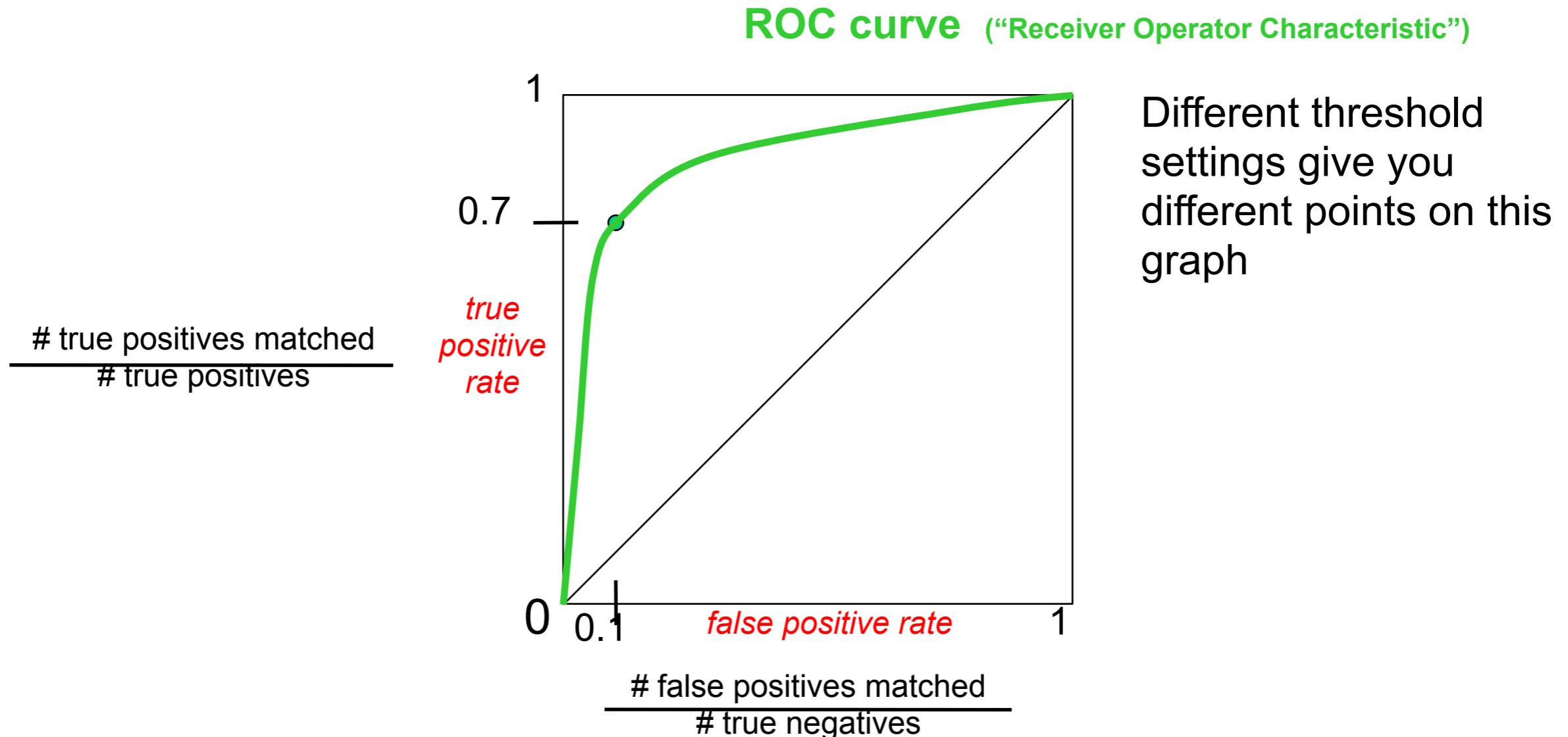
# Evaluating the results

---



# Evaluating the results

How can we measure the performance of a feature matcher?



## ROC Curves

- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods
- For more info: [http://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic](http://en.wikipedia.org/wiki/Receiver_operating_characteristic)

# Worktime: Project 1



# 11 September 2014 Consultations

# Bryan

# Group #

# Pengxiang

# Group #