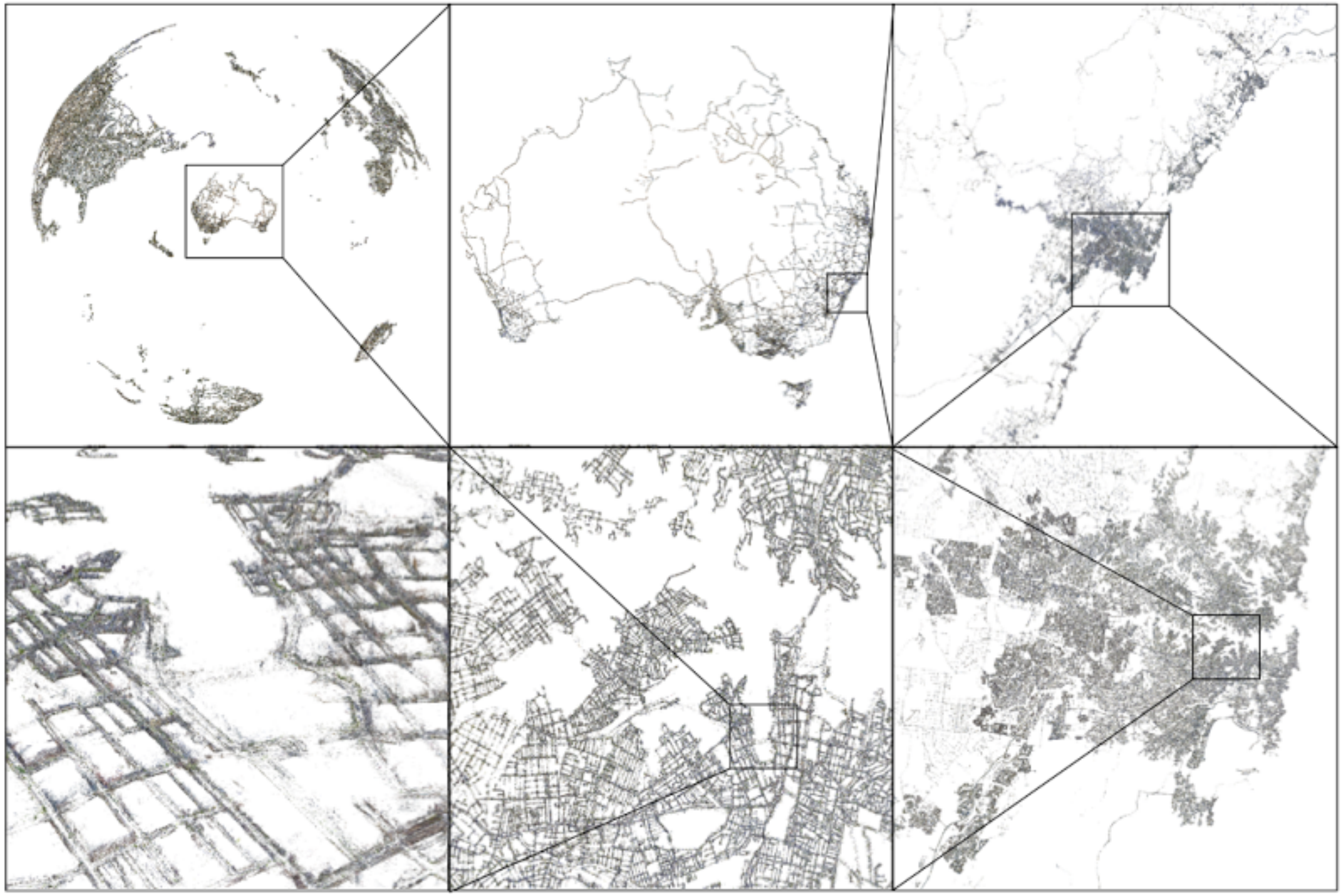
3D Reconstruction with Computer Vision
Meeting 23: Local Features Redux

CS 378 Fall 2014, UT Austin, Bryan Klingner, 13 November

Final Project

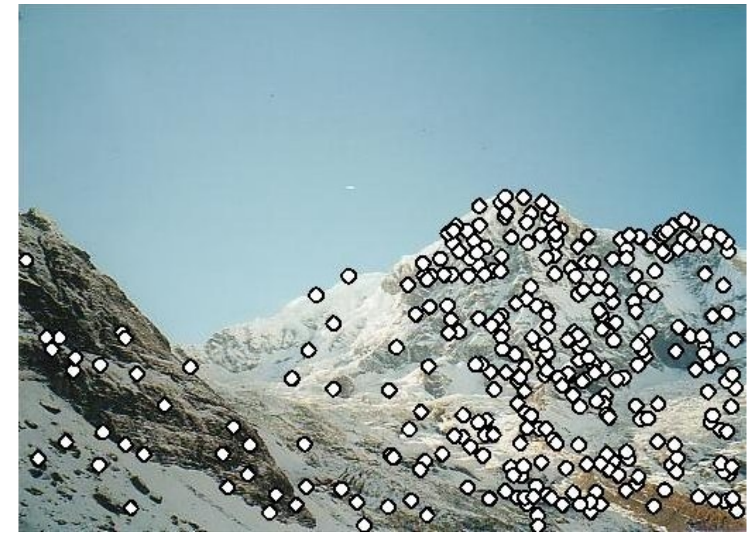


- **Progress report 1 Due Today, 13 November**
- **Progress report 2 Due Thursday, 20 November**

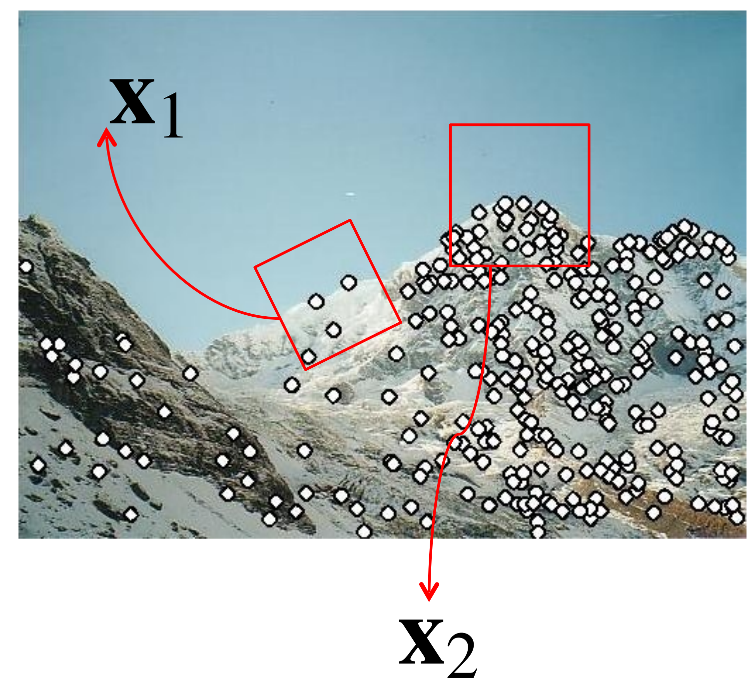
- **First in-class presentations Tuesday, 25 Nov.**
- **Sign up by sending a pull request for project_4/presentations.md**

Local features: main components

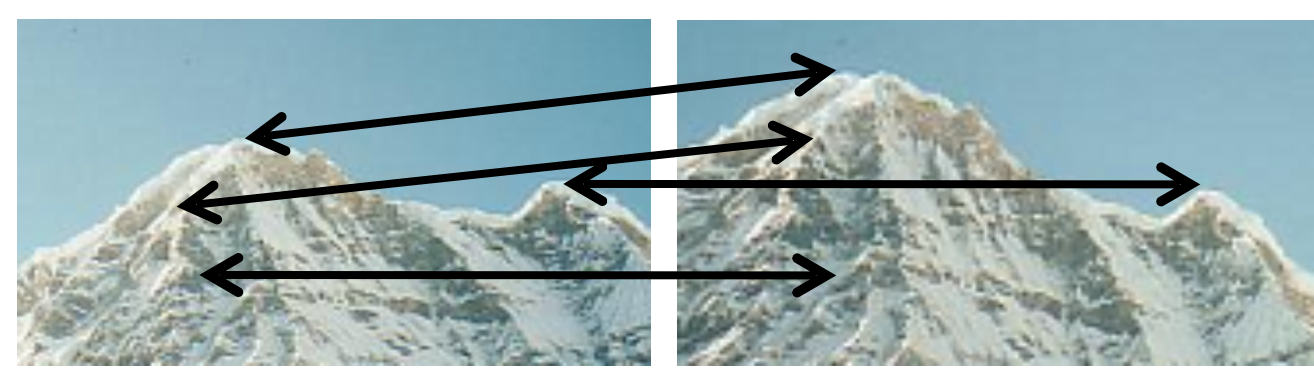
1) Detection: Identify the interest points



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



3) Matching: Determine correspondence between descriptors in two views



Kristen Grauman

Local features: desired properties

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

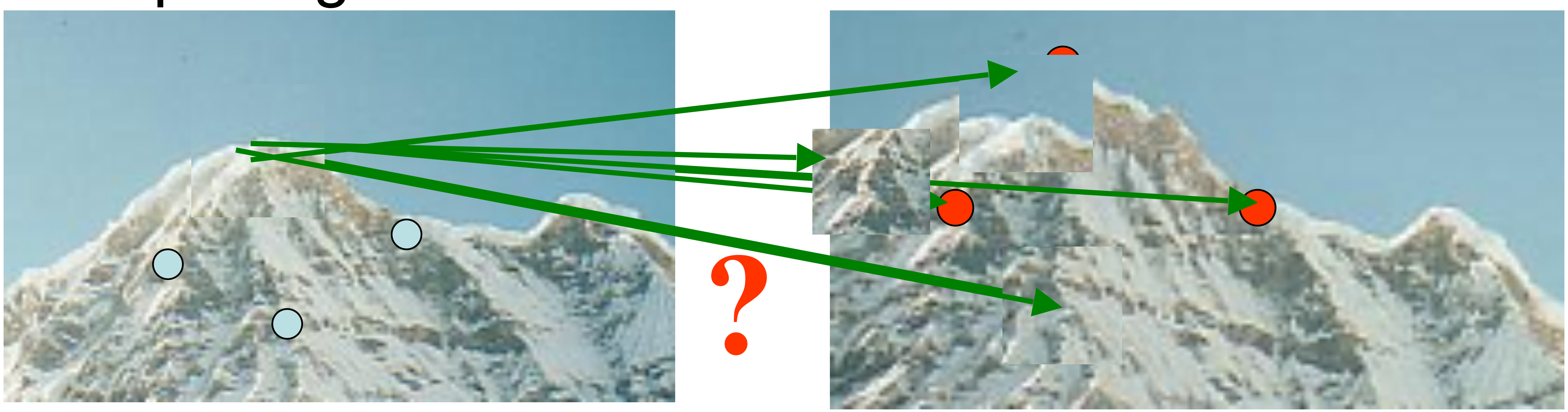
Goal: interest operator repeatability

- We want to detect (at least some of) the same points in both images.
- Yet we have to be able to run the detection procedure *independently* per image.


No chance to find true matches!

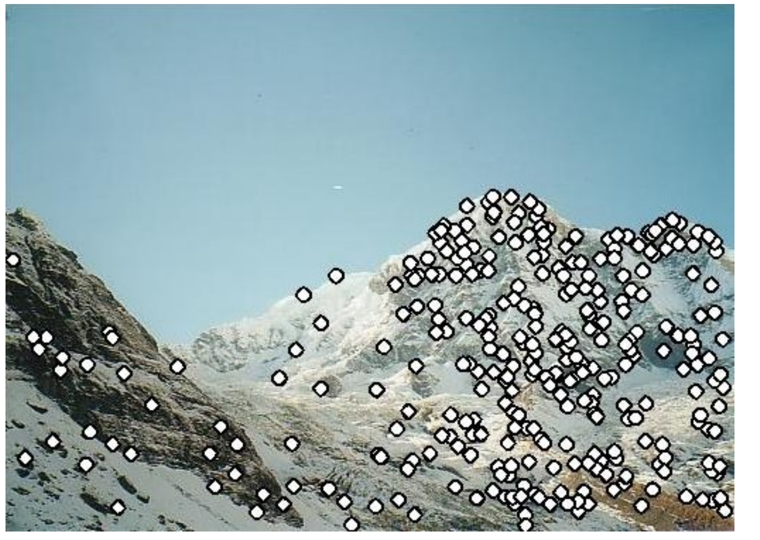
Goal: descriptor distinctiveness

- We want to be able to reliably determine which point goes with which.



- Must provide some invariance to geometric and photometric differences between the two views.

Local features: main components



1) Detection: Identify the interest points

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

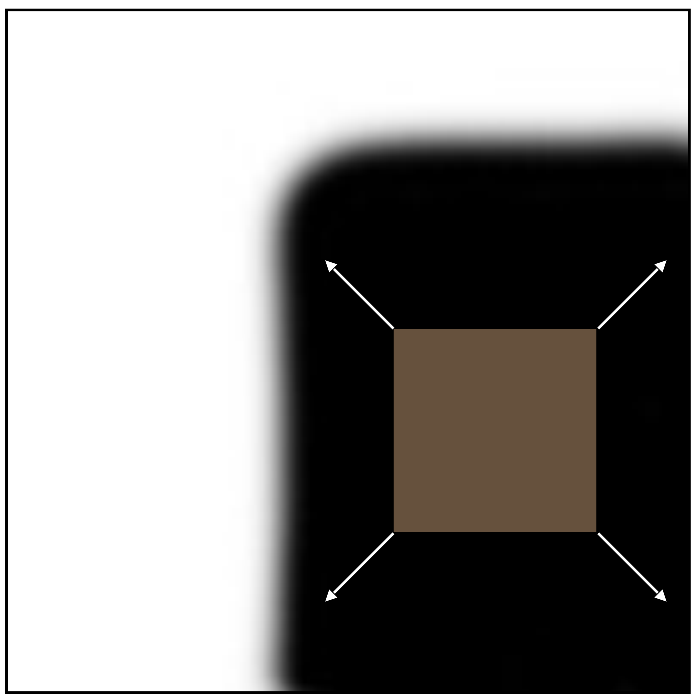
3) Matching: Determine correspondence between descriptors in two views

- What points would you choose?

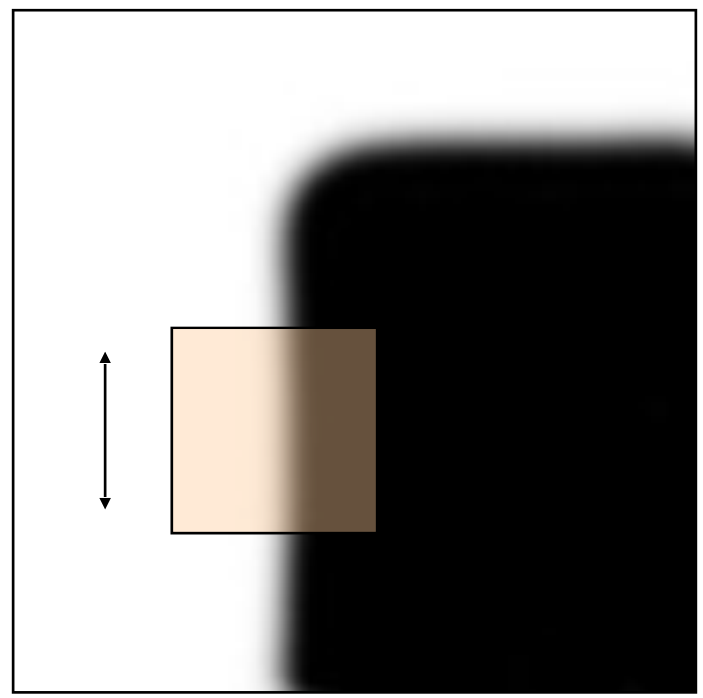
Corners as distinctive interest points

We should easily recognize the point by looking through a small window

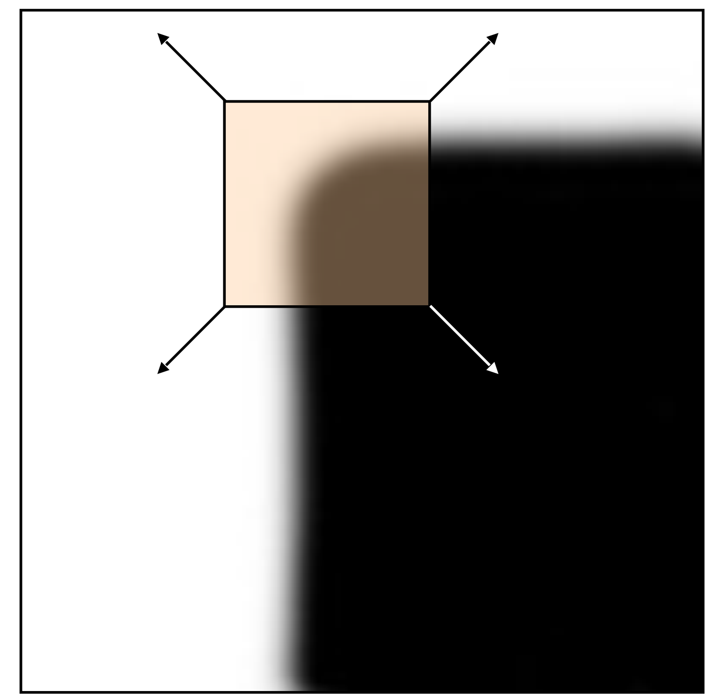
Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction



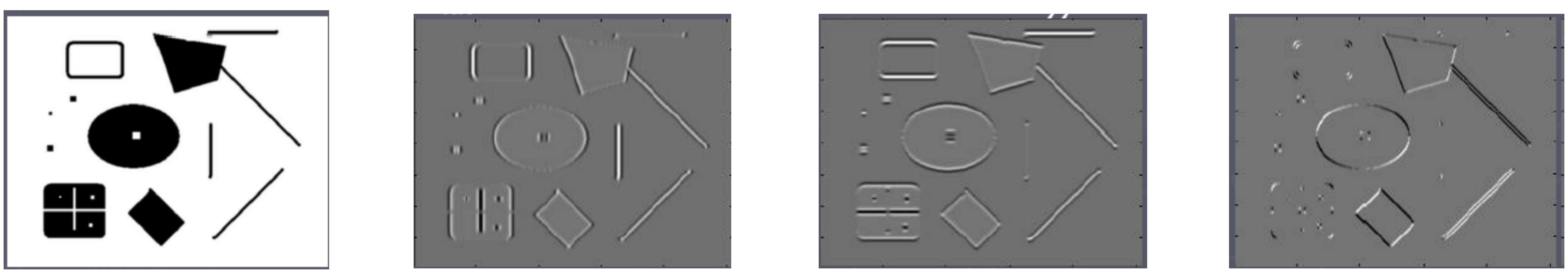
"corner":
significant change in all directions

Slide credit: Alyosha Efros, Darya Frolova, Denis Simakov

Corners as distinctive interest points

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

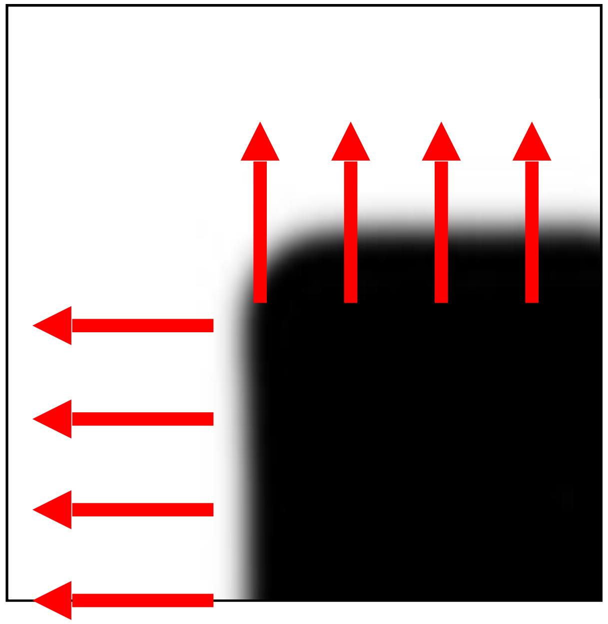
2 x 2 matrix of image derivatives (averaged in neighborhood of a point).



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

What does this matrix reveal?

First, consider an axis-aligned corner:



What does this matrix reveal?

First, consider an axis-aligned corner:

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis

Look for locations where **both** λ's are large.

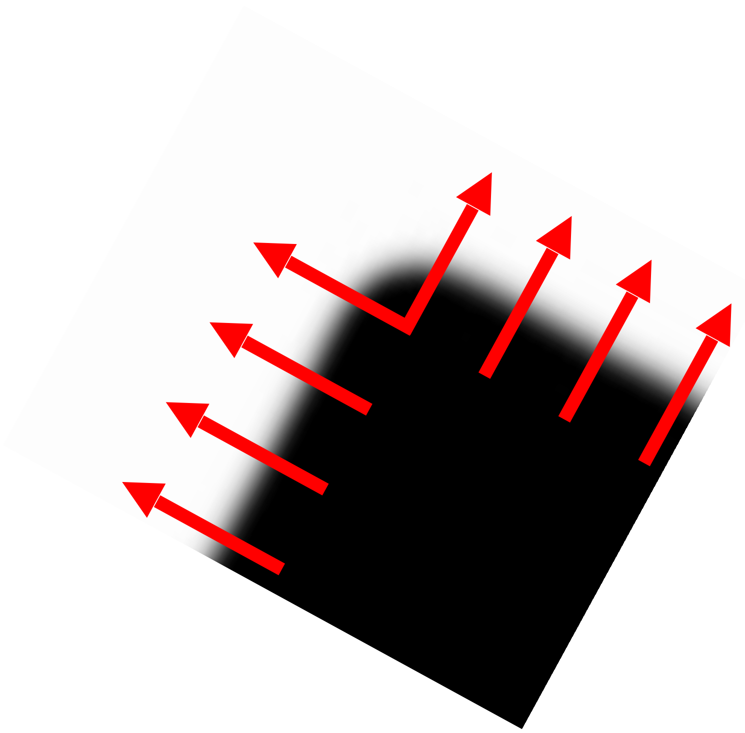
If either λ is close to 0, then this is **not** corner-like.

What if we have a corner that is not aligned with the image axes?

What does this matrix reveal?

Since *M* is symmetric, we have

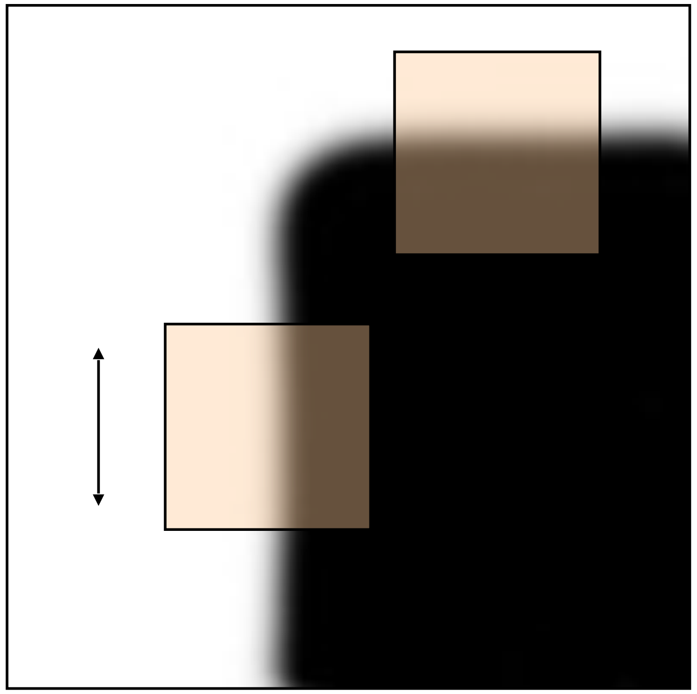
$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$



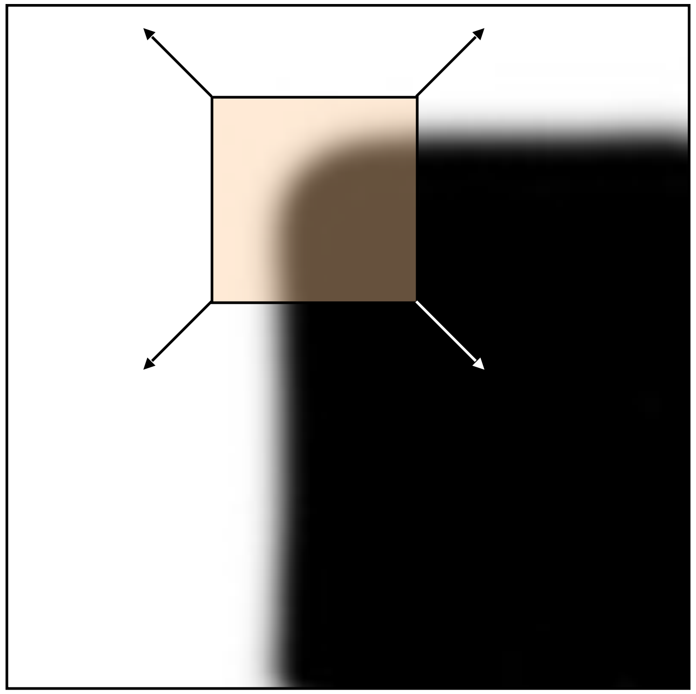
$$Mx_i = \lambda_i x_i$$

The *eigenvalues* of *M* reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

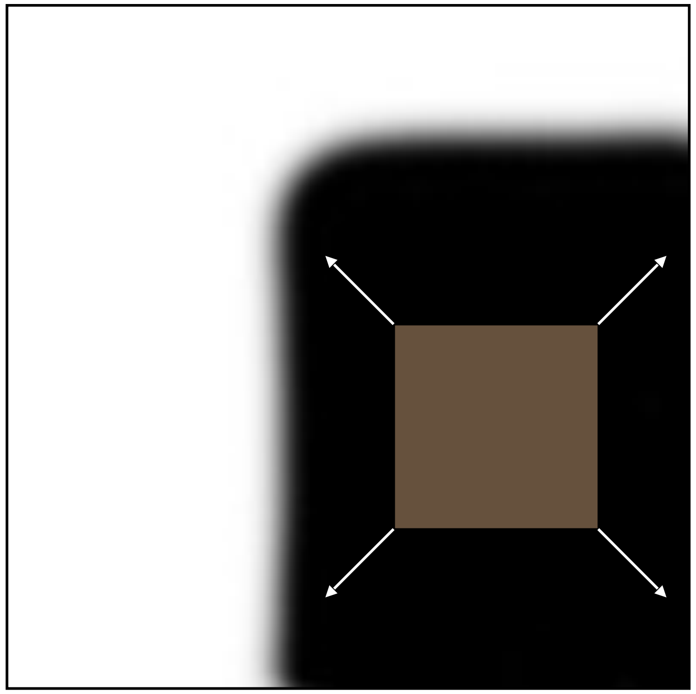
Corner response function



"edge":
$\lambda_1 >> \lambda_2$
$\lambda_2 >> \lambda_1$



"corner":
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;



"flat" region
λ_1 and λ_2 are sma

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

Harris corner detector

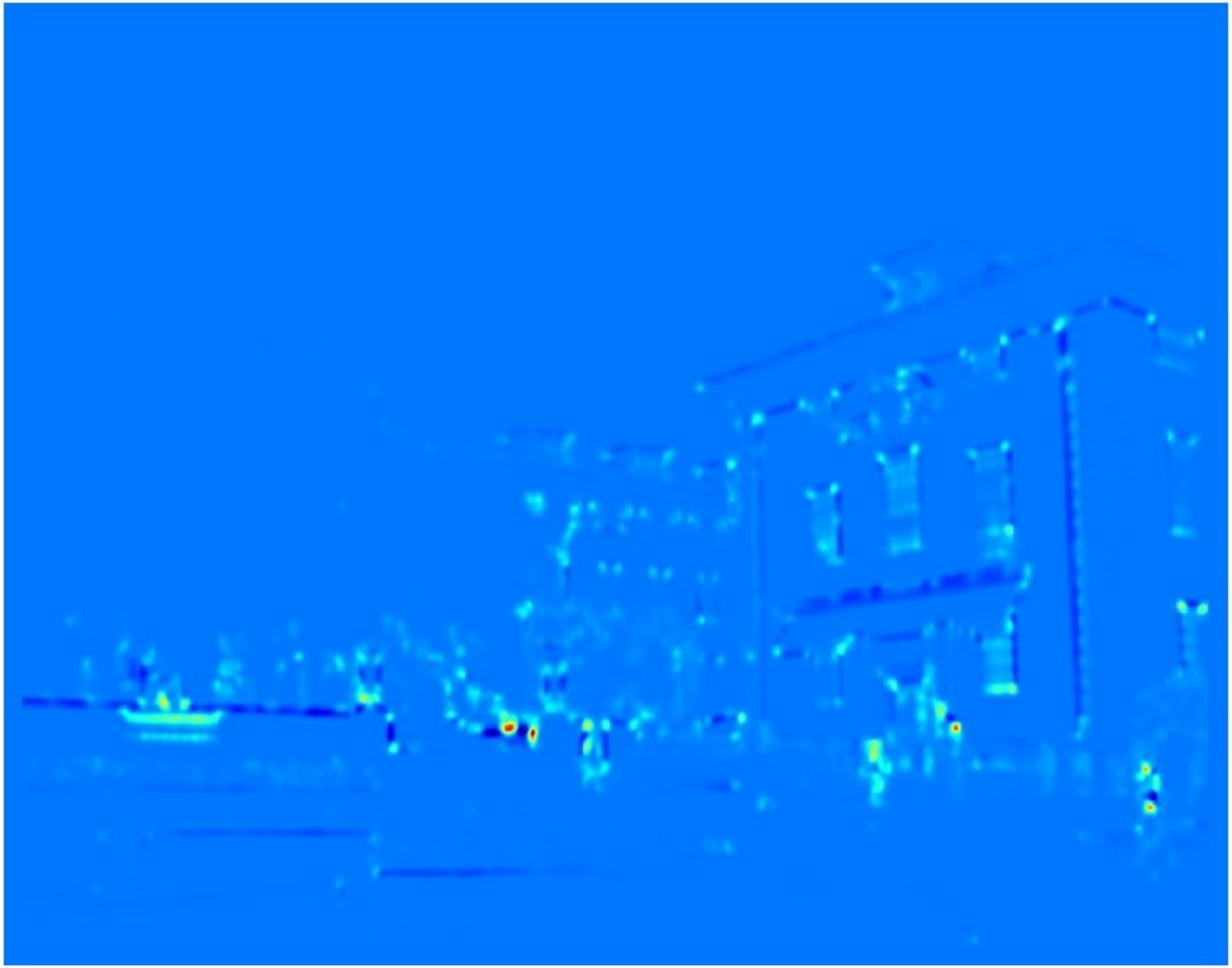
1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (f> threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

Example of Harris application

Kristen Grauman

Example of Harris application

Compute corner response at every pixel.



Kristen Grauman

Example of Harris application

Kristen Grauman

Properties of the Harris corner detector

Rotation invariant? Yes

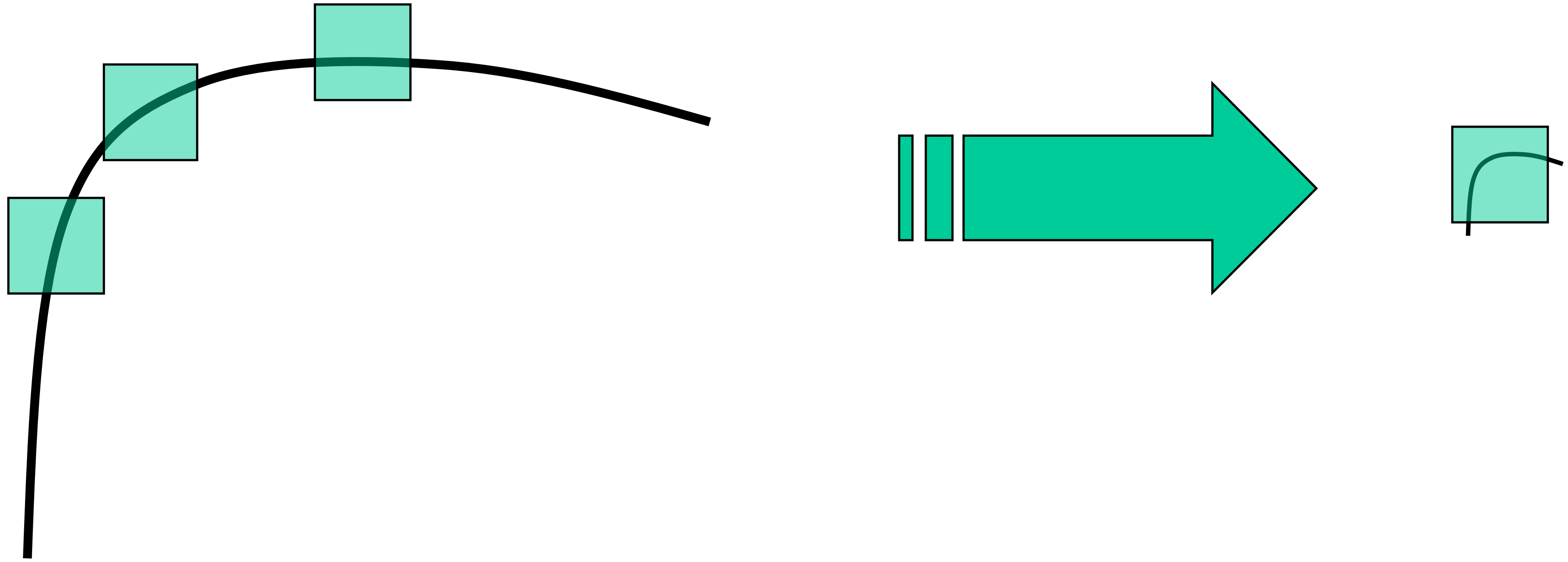
$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$

Scale invariant?

Properties of the Harris corner detector

Rotation invariant? Yes

Scale invariant? No



All points will be classified as edges

Corner !

Scale invariant interest points

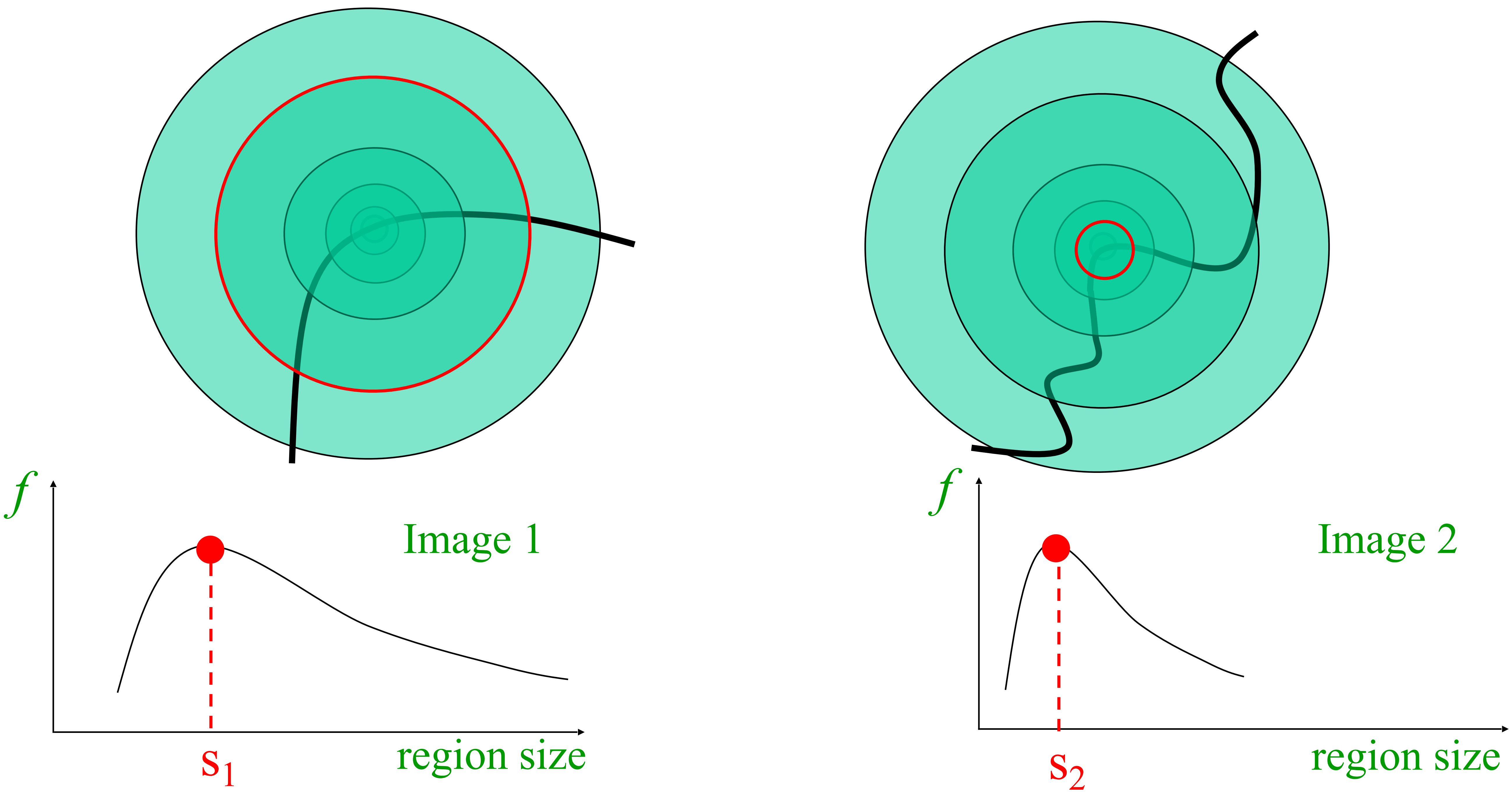
How can we independently select interest points in each image, such that the detections are repeatable across different scales?

Kristen Grauman

Automatic scale selection

Intuition:

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



Steve Seitz

Image matching

by [Diva Sian]

by [swashford]

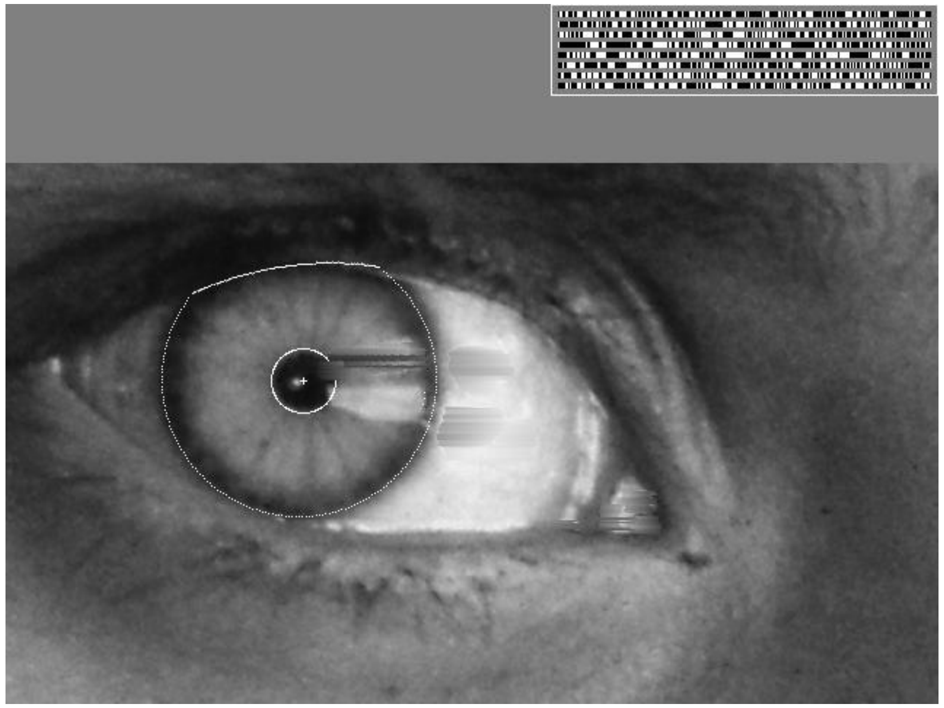
Harder case

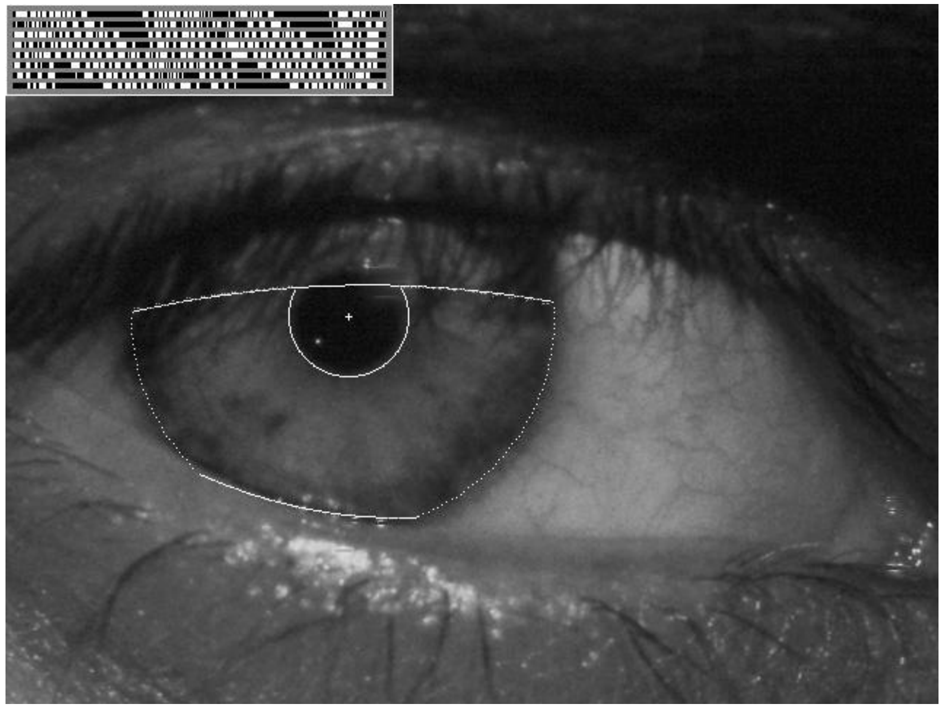
by Diva Sian

by scgbt

Even harder case

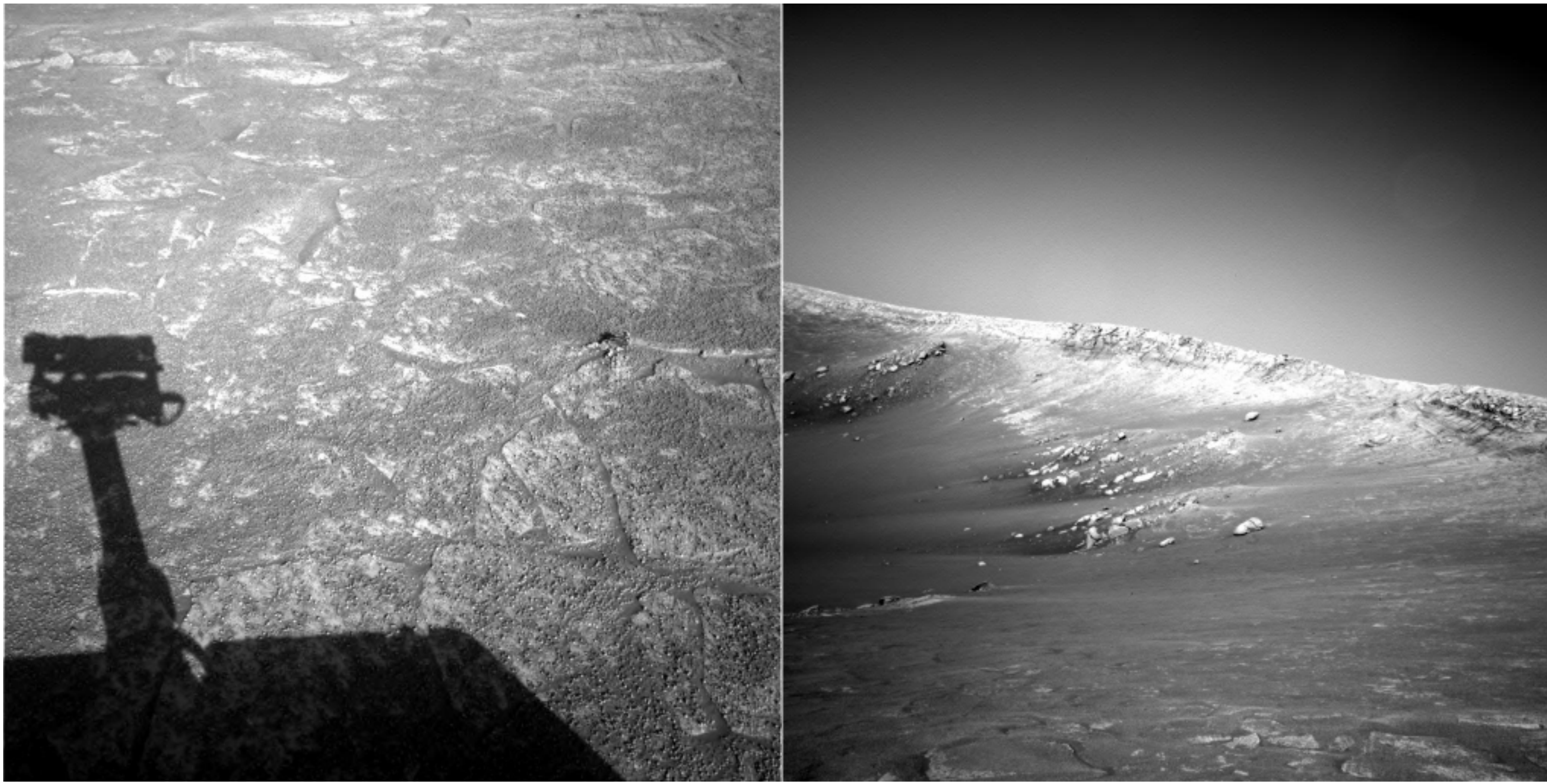
"*How the Afghan Girl was Identified by Her Iris Patterns*" **Read the story**





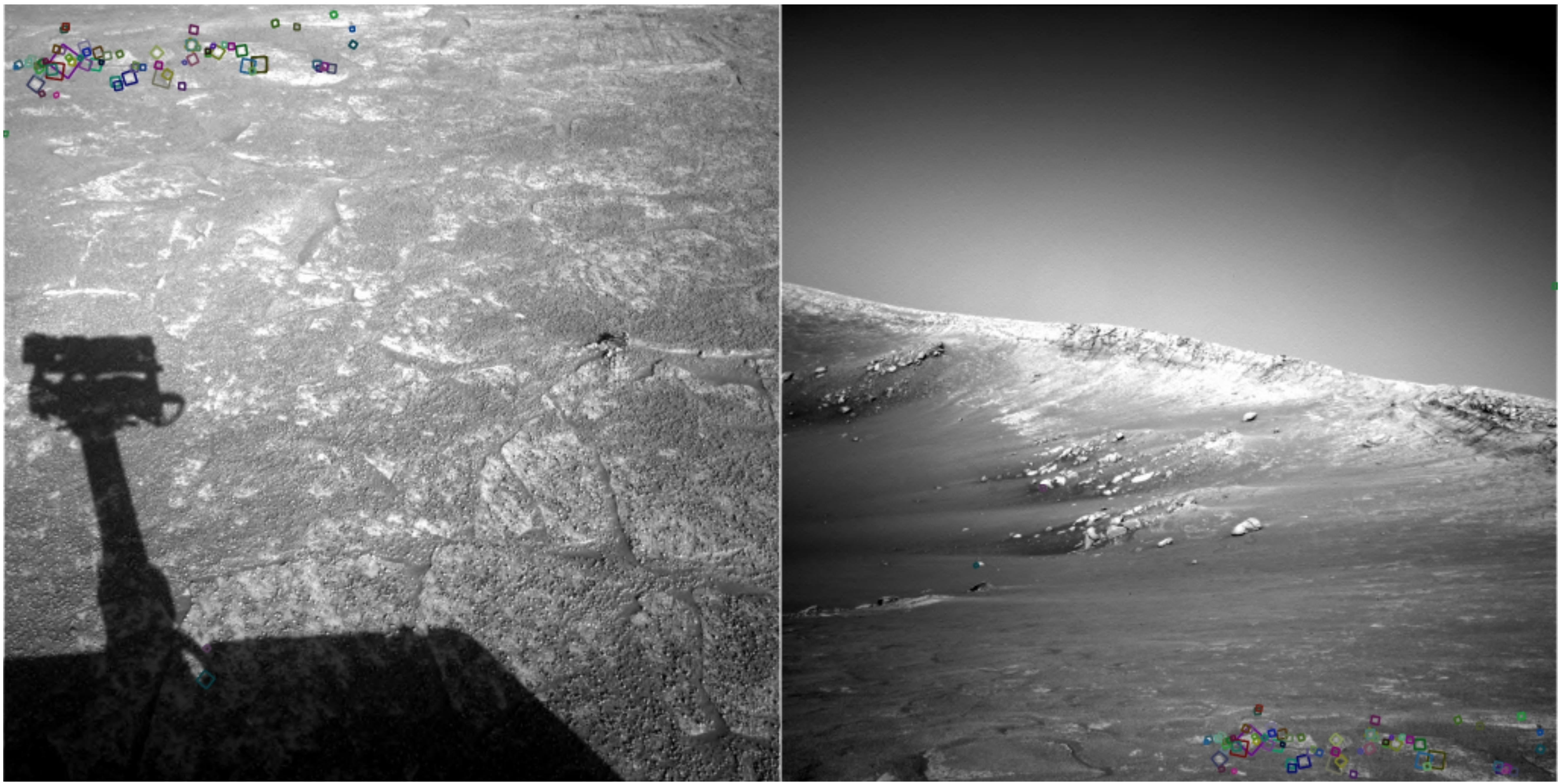
Steve Seitz

Harder still?



NASA Mars Rover images

Answer below (look for tiny colored squares…)

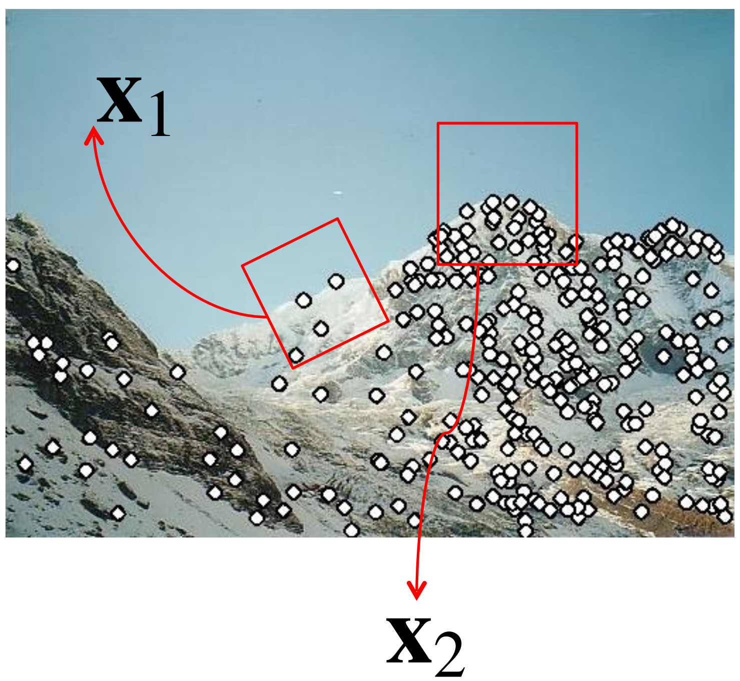


NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Local features: main components

1) Detection: Identify the interest points

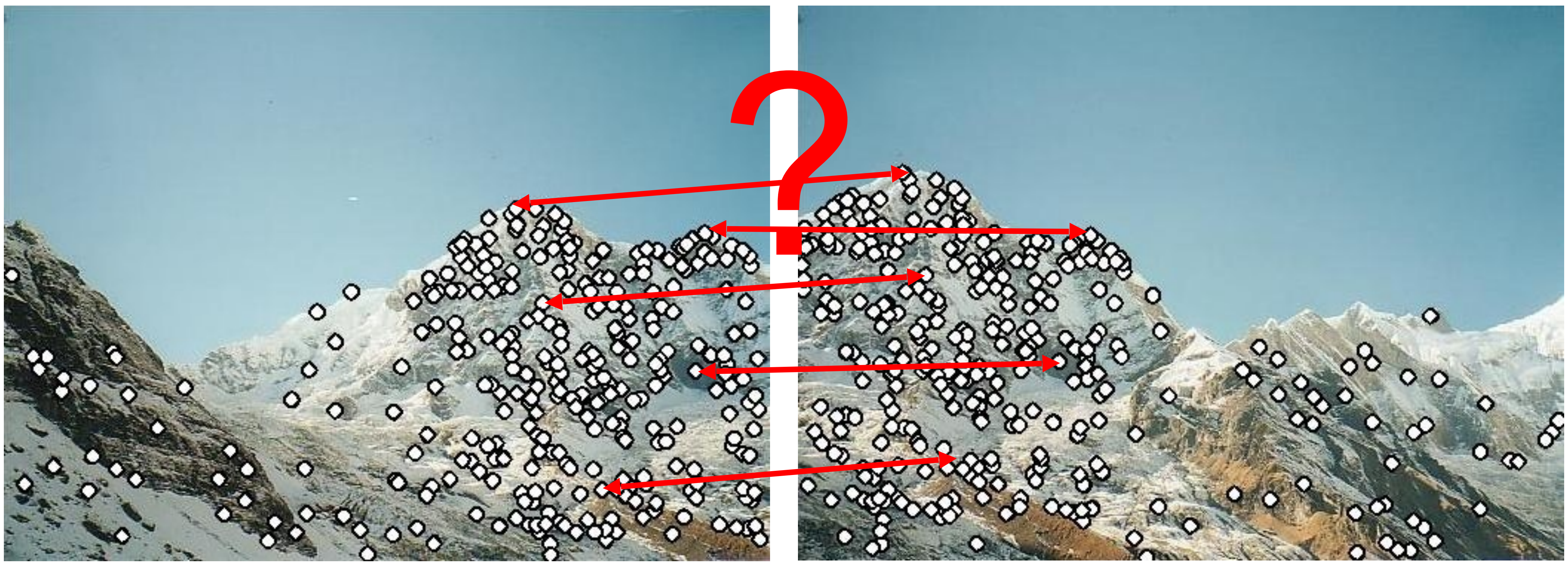
2) Description: Extract vector feature descriptor surrounding each interest point.



3) Matching: Determine correspondence between descriptors in two views

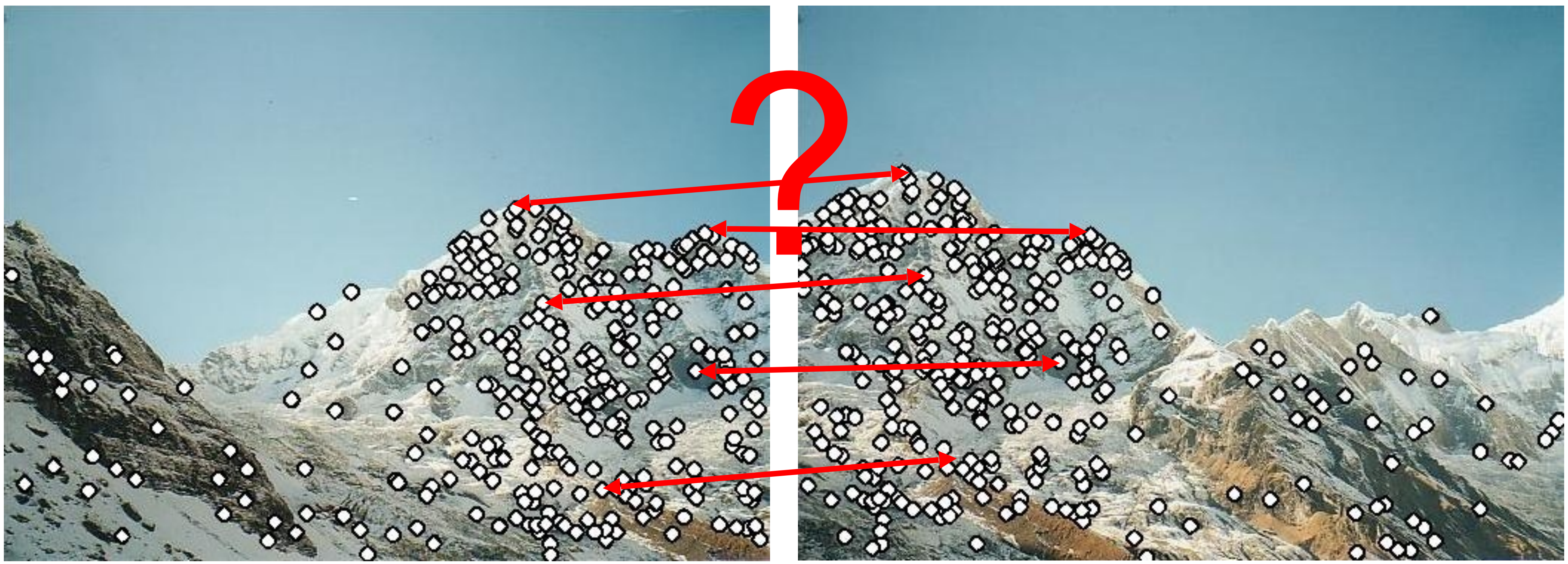
Feature descriptors

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

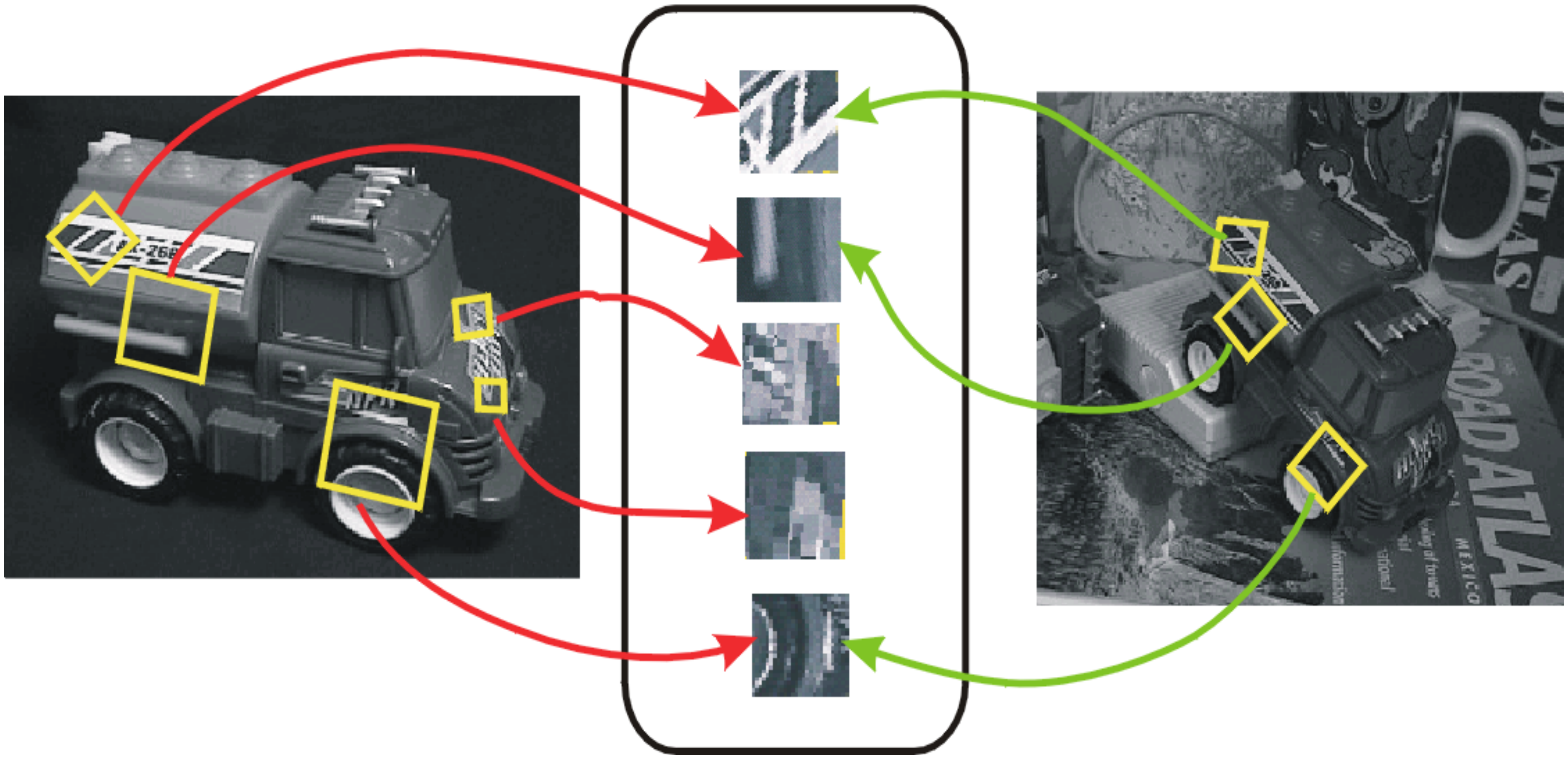
Figure from T. Tuytelaars ECCV 2006 tutorial

Steve Seitz

Invariant local features

Find features that are invariant to transformations

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



Feature Descriptors

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

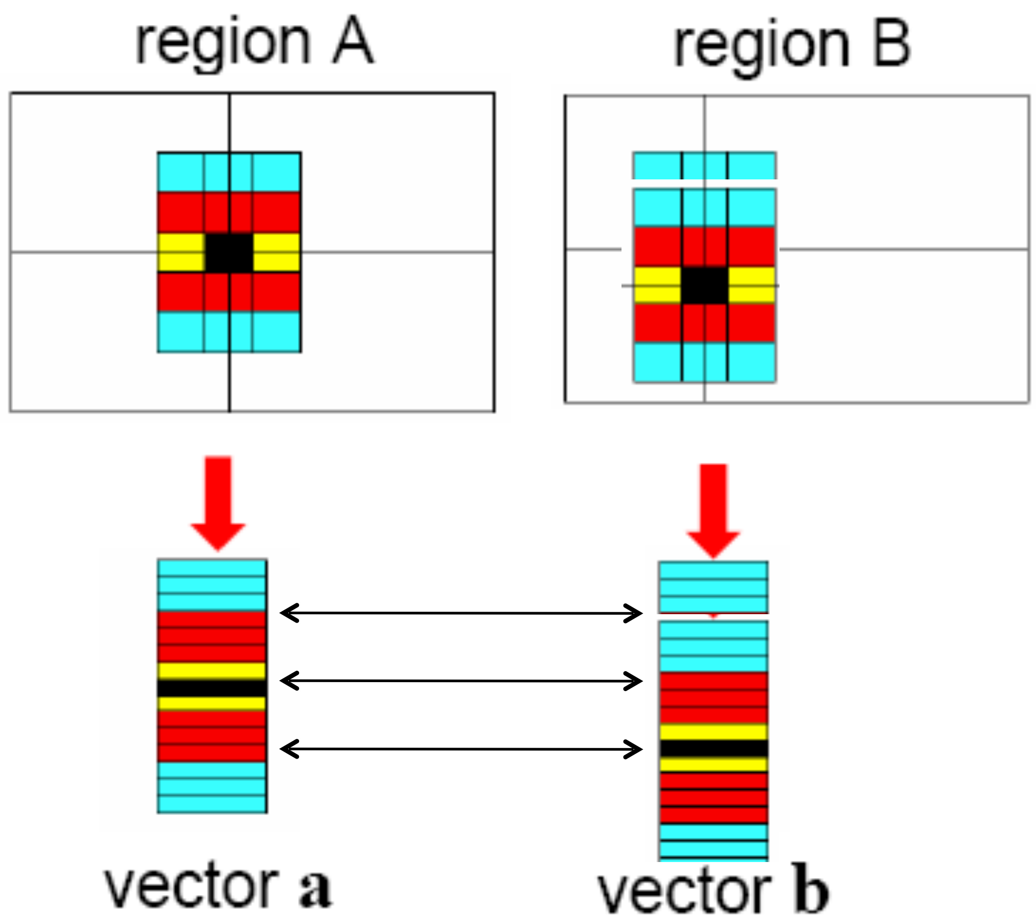
- This is called transformational ***invariance***
- Most feature methods are designed to be invariant to
  - Translation, 2D rotation, scale
- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 60 degrees)
  - Limited affine transformations (some are fully affine invariant)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

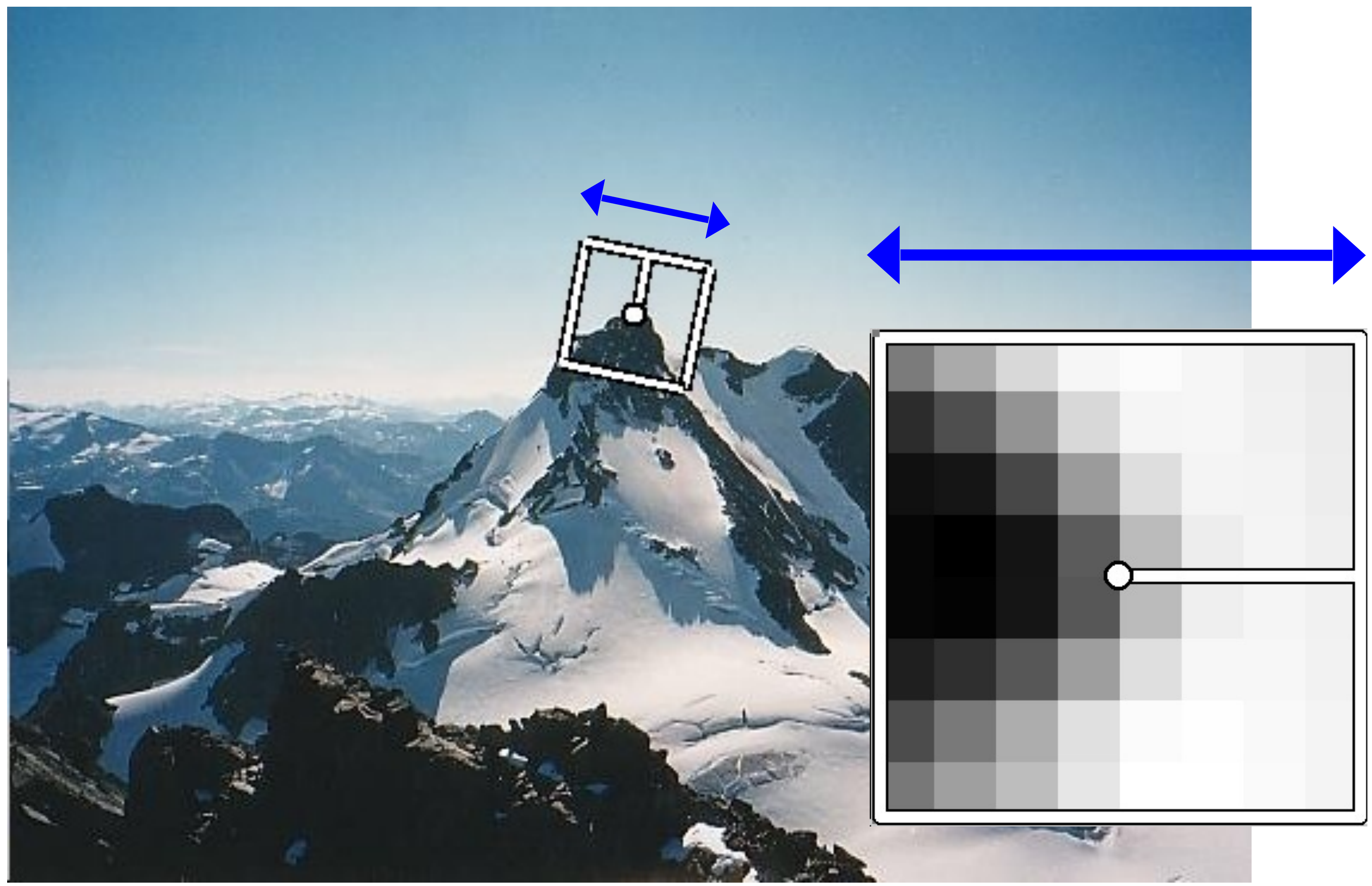
Are raw patches good descriptors?



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.

Image from Matthew Brown

Steve Seitz

Rotation invariance for feature descriptors

Find dominant orientation of the image window

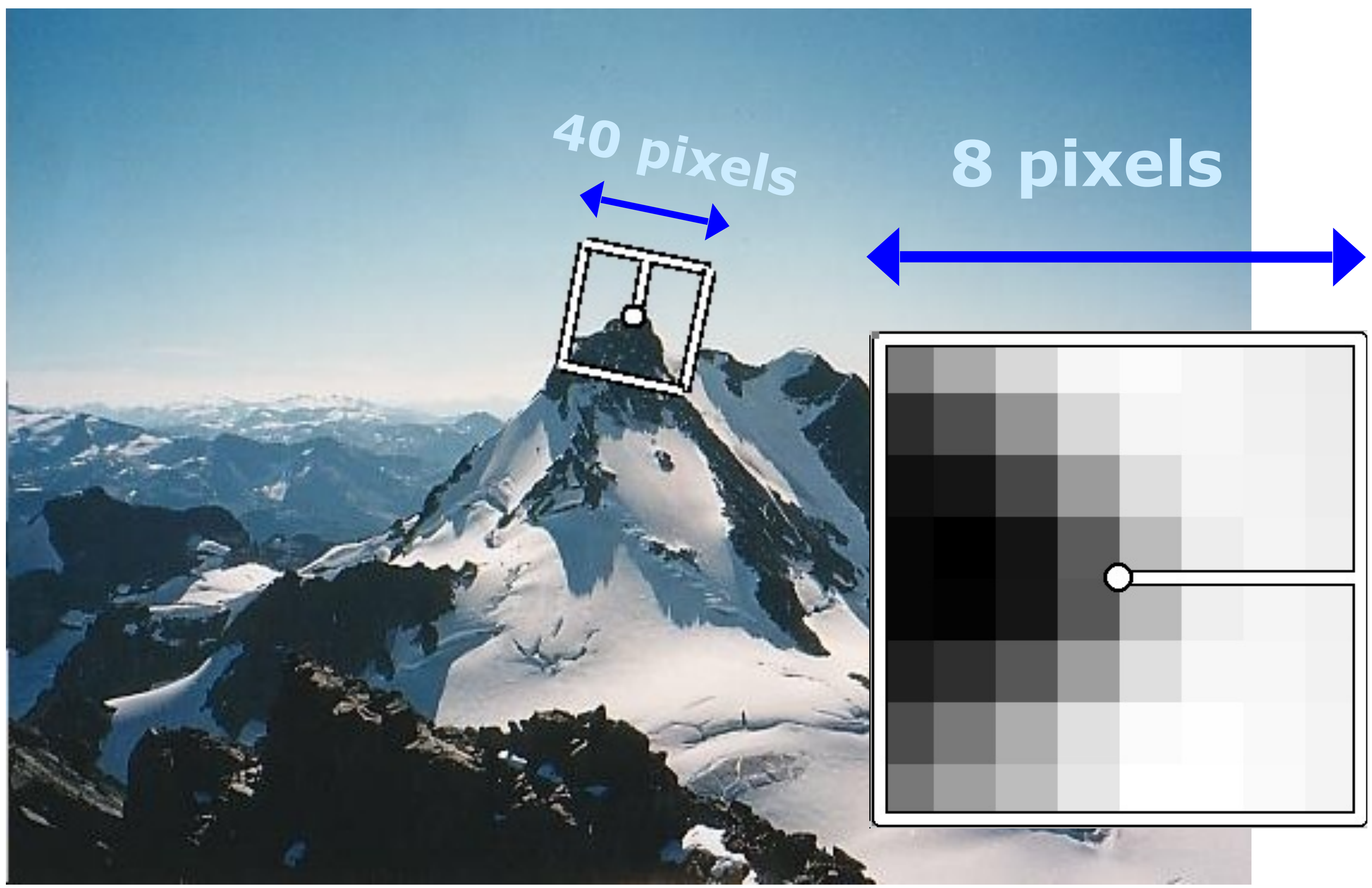
- This is given by $\mathbf{x}_+$, the eigenvector of $\mathbf{H}$ corresponding to λ_+
  - λ_+ is the *larger* eigenvalue
- Rotate the window according to this angle

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

Steve Seitz

Detections at multiple scales

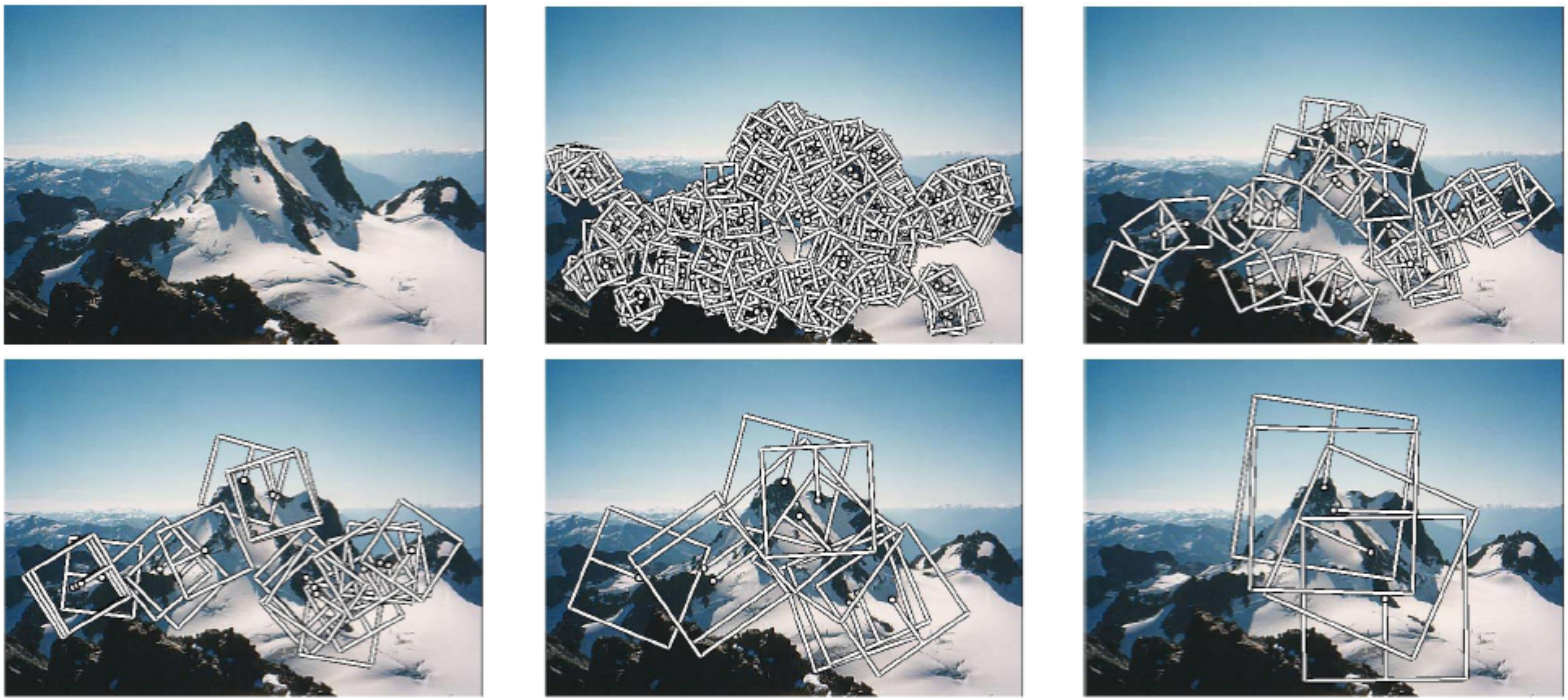


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Steve Seitz

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

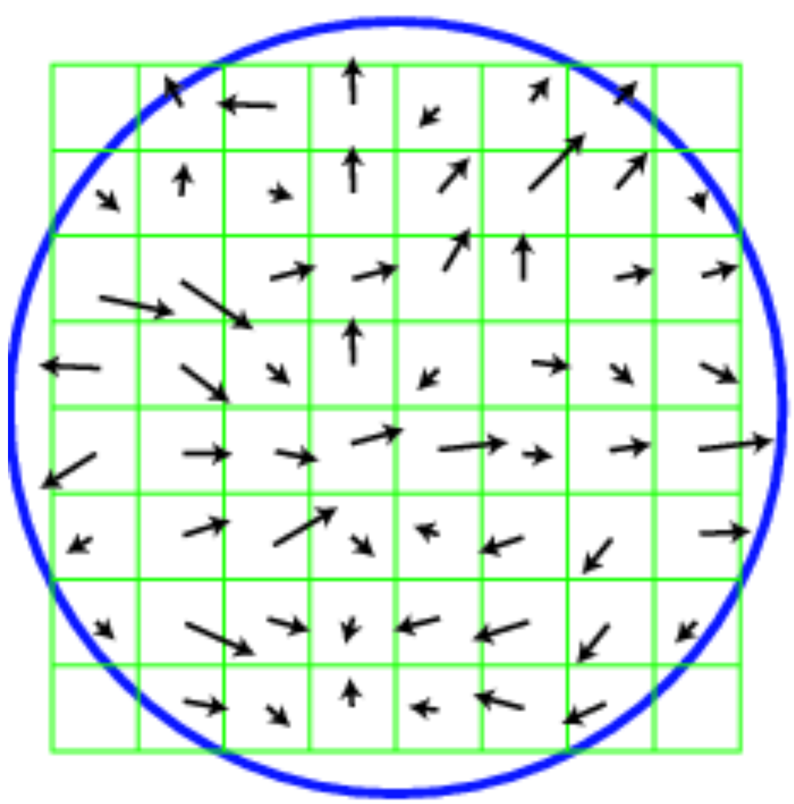
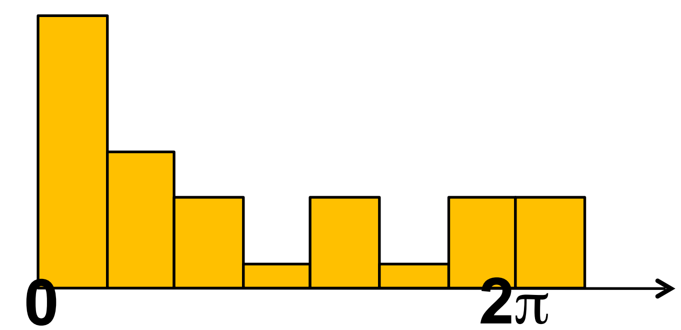
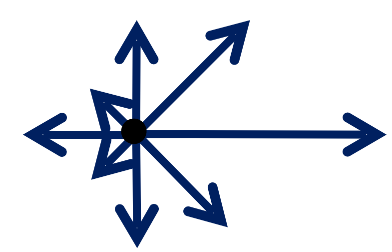


Image gradients



angle histogram



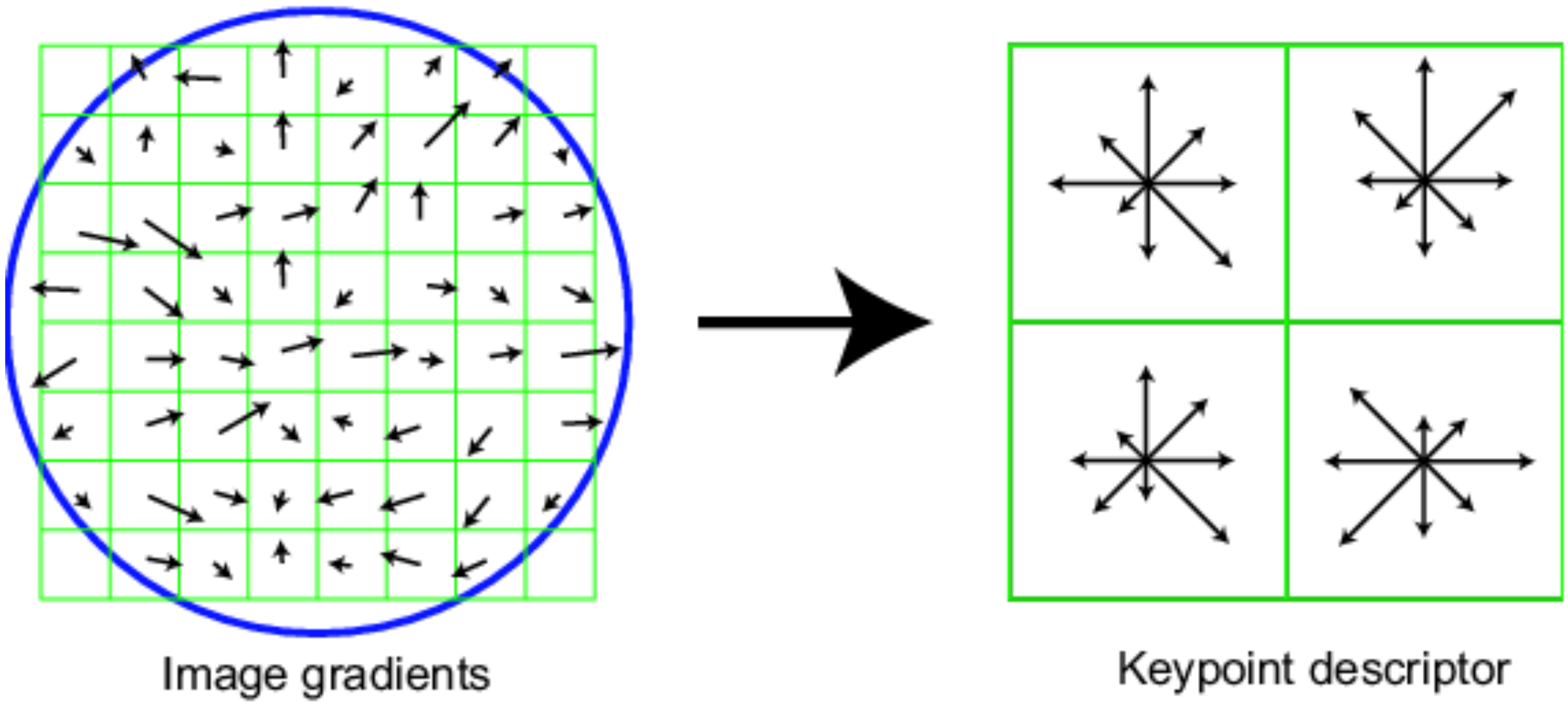
Adapted from slide by David Lowe

Steve Seitz

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



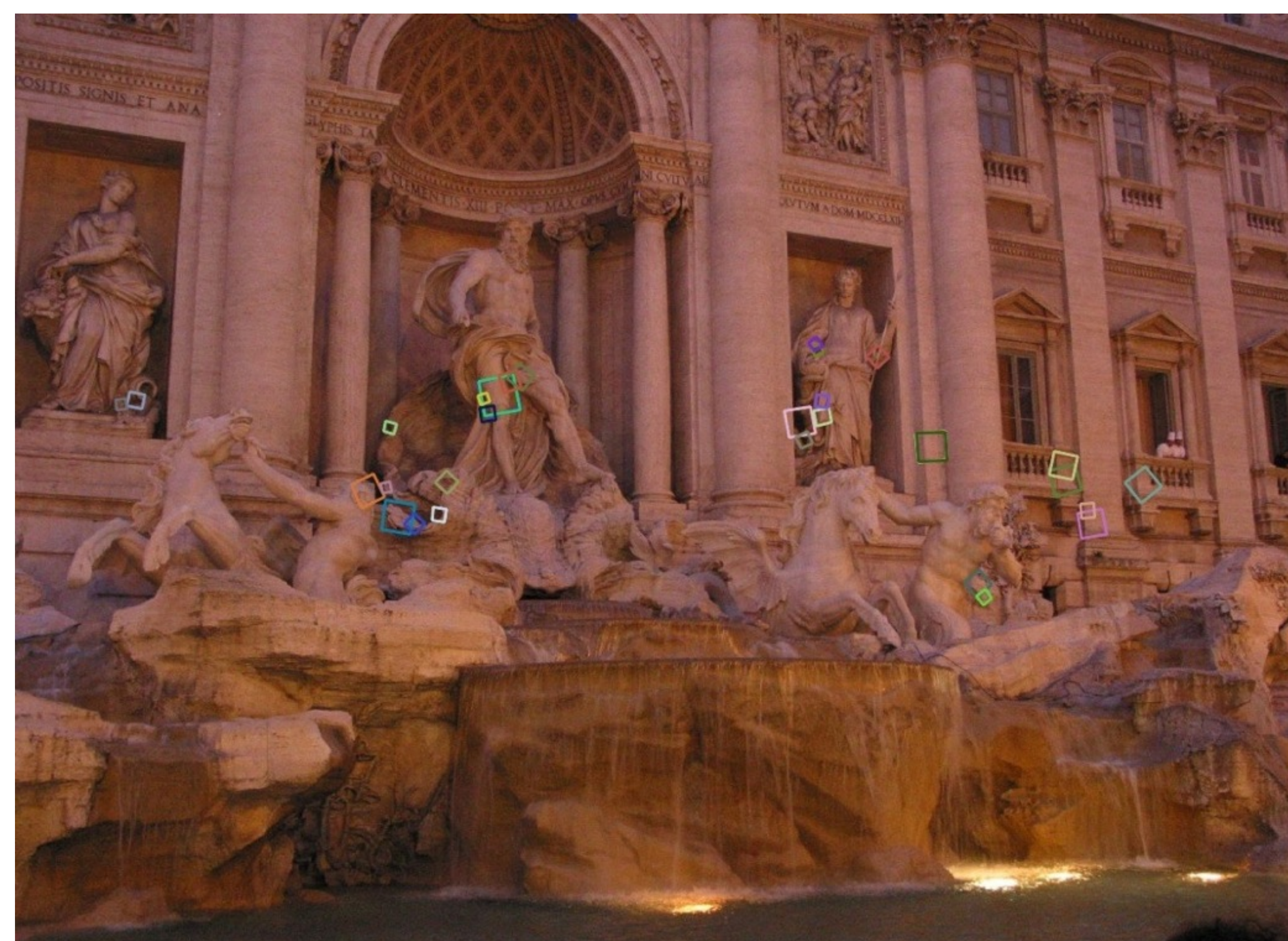
Adapted from slide by David Lowe

Steve Seitz

Properties of SIFT

Extraordinarily robust matching technique

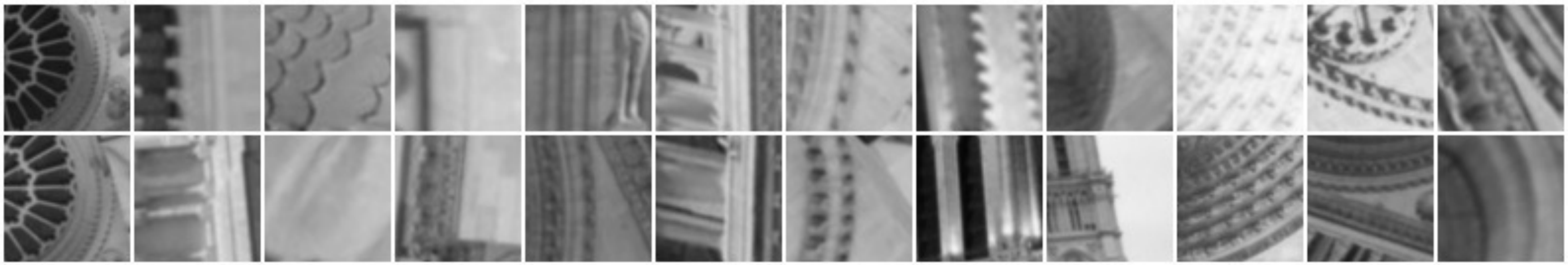
- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
  - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



Steve Seitz

When does SIFT fail?

Patches SIFT thought were the same but aren't:

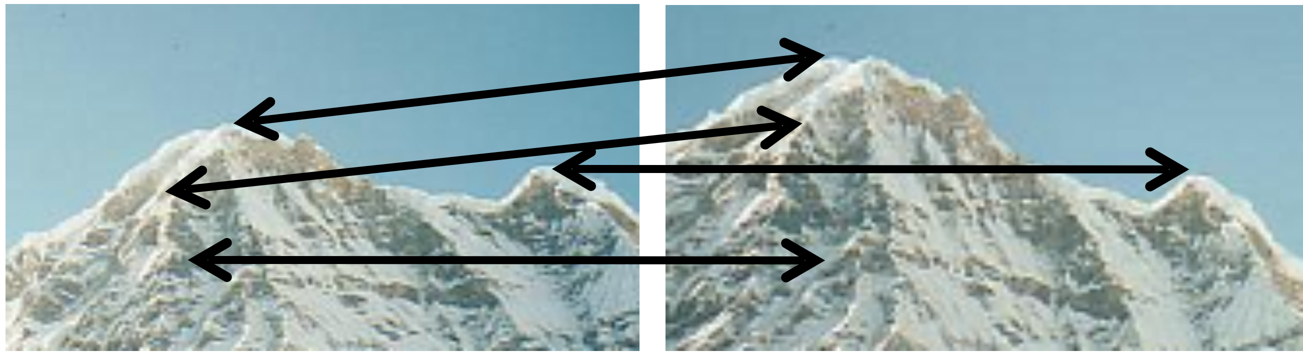


Local features: main components

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?

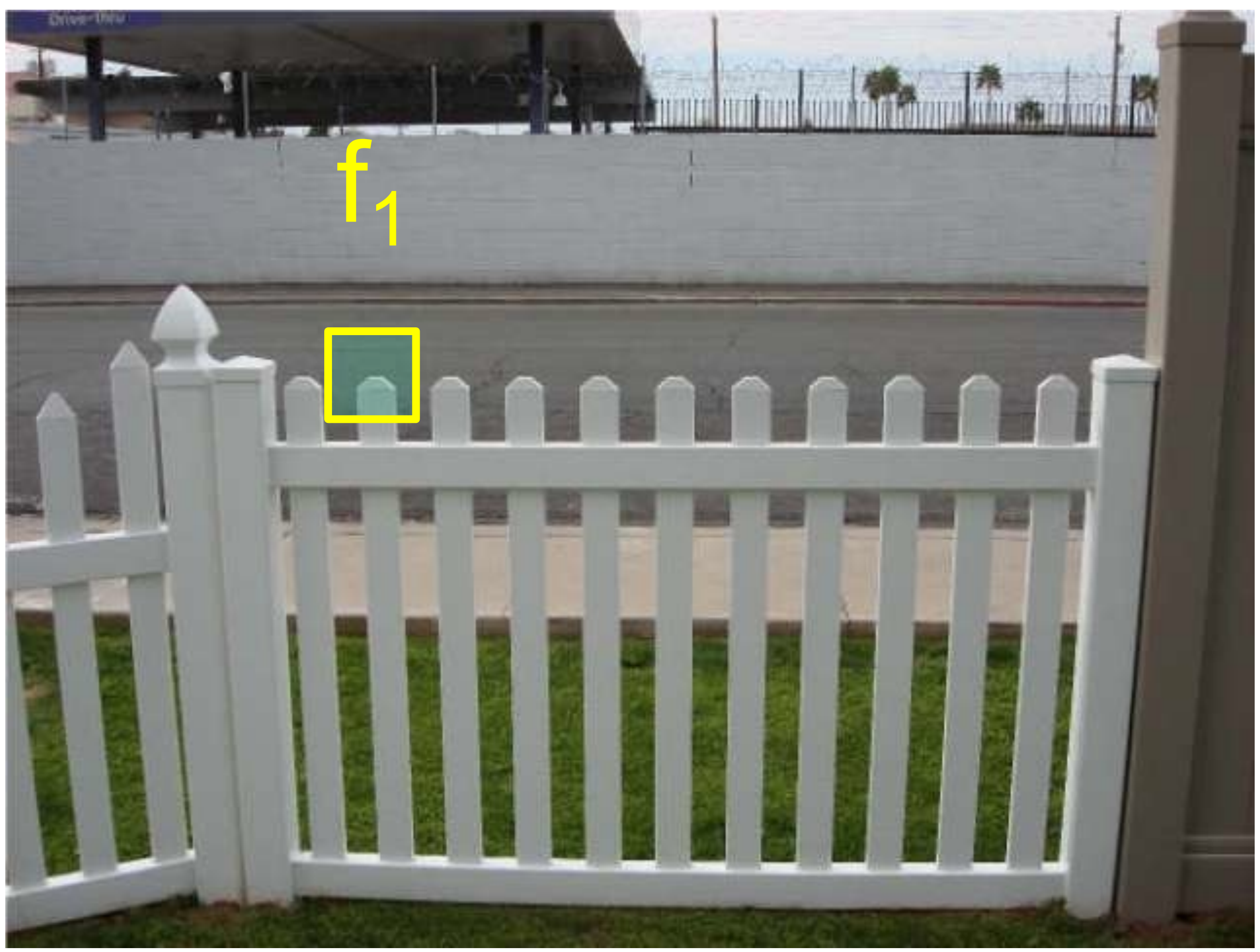
1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

Steve Seitz

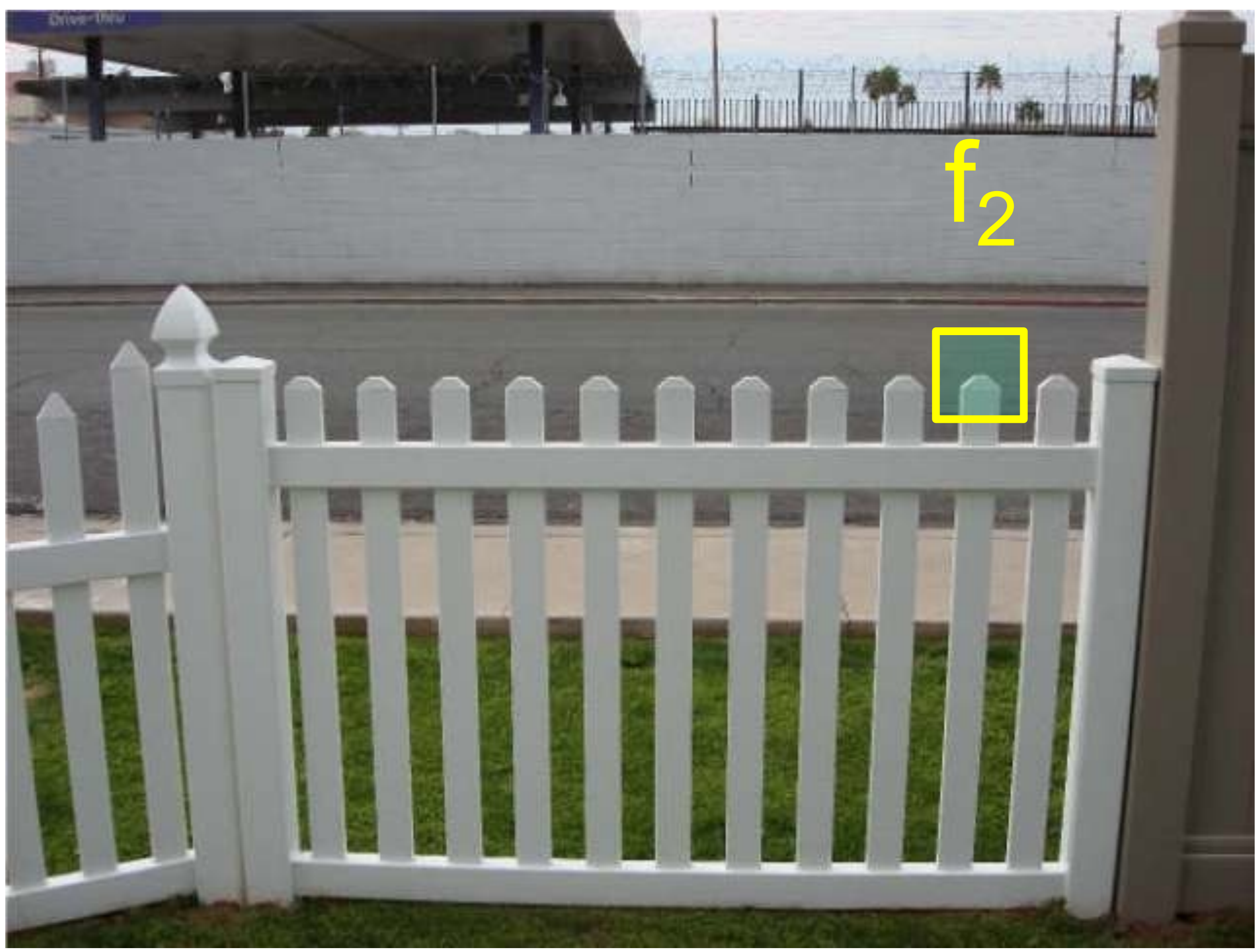
Feature distance

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

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



I_1



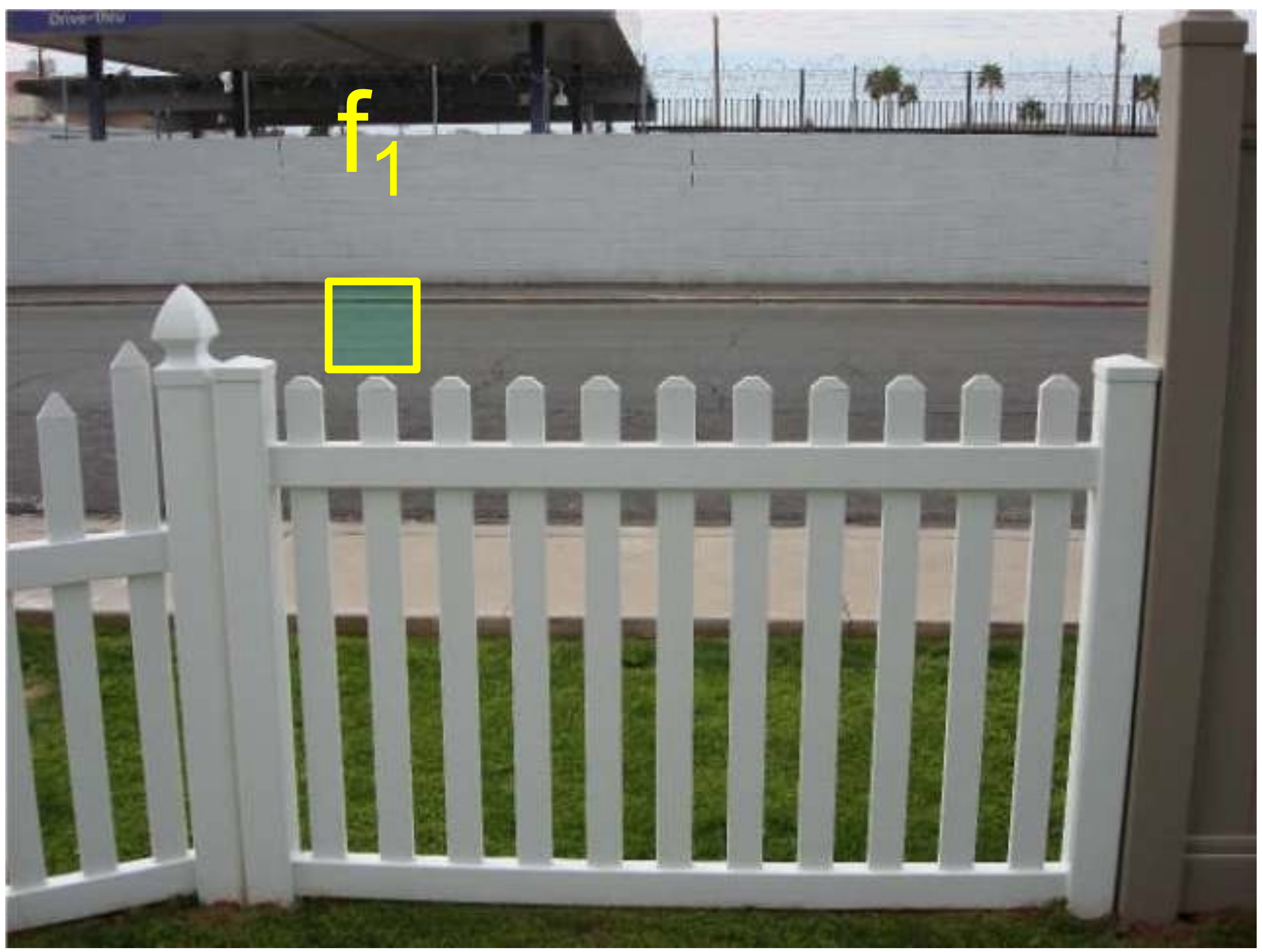
I_2

Steve Seitz

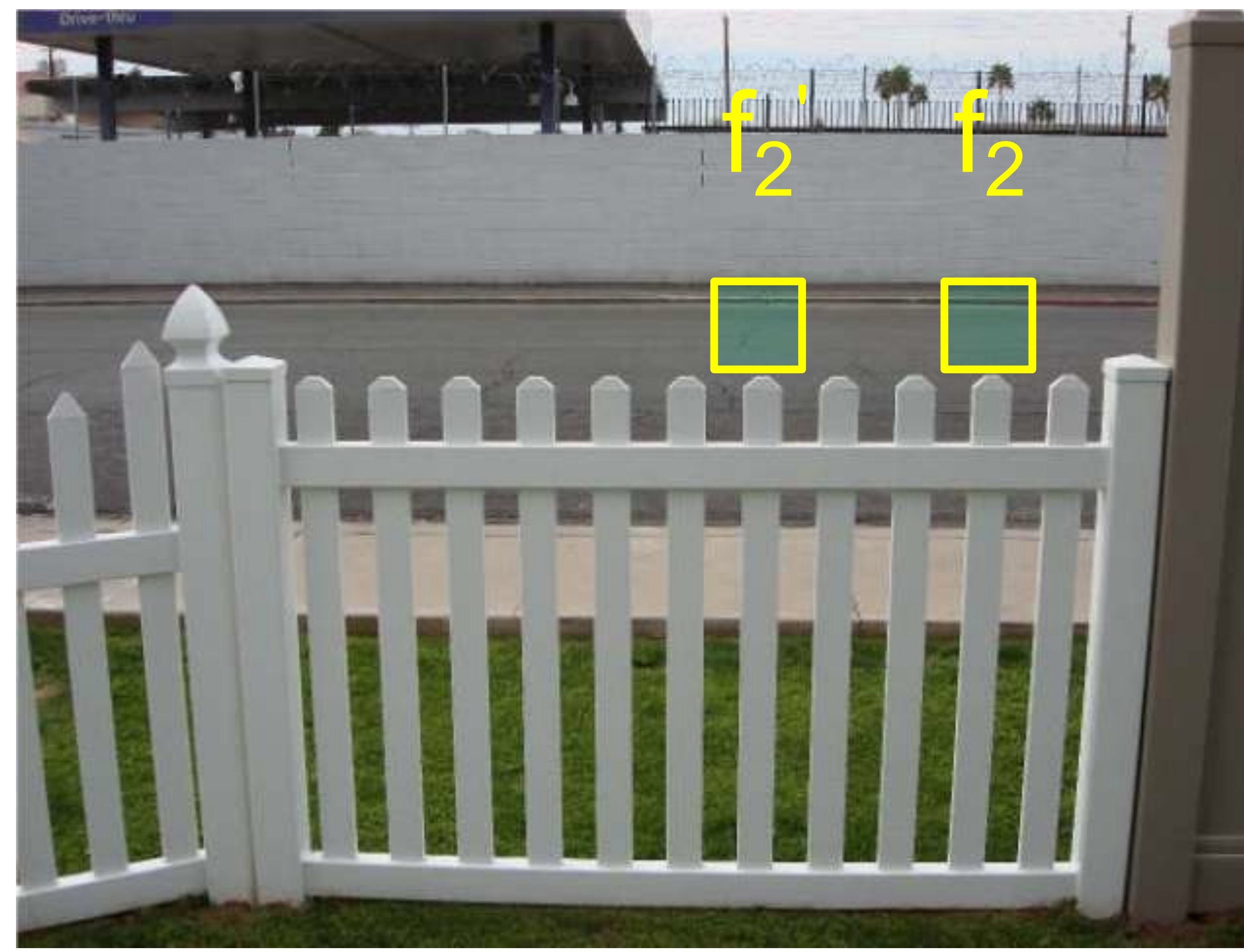
Feature distance

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

- Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2nd best SSD match to f_1 in I_2
  - gives large values for ambiguous matches
  - "Lowe ratio"

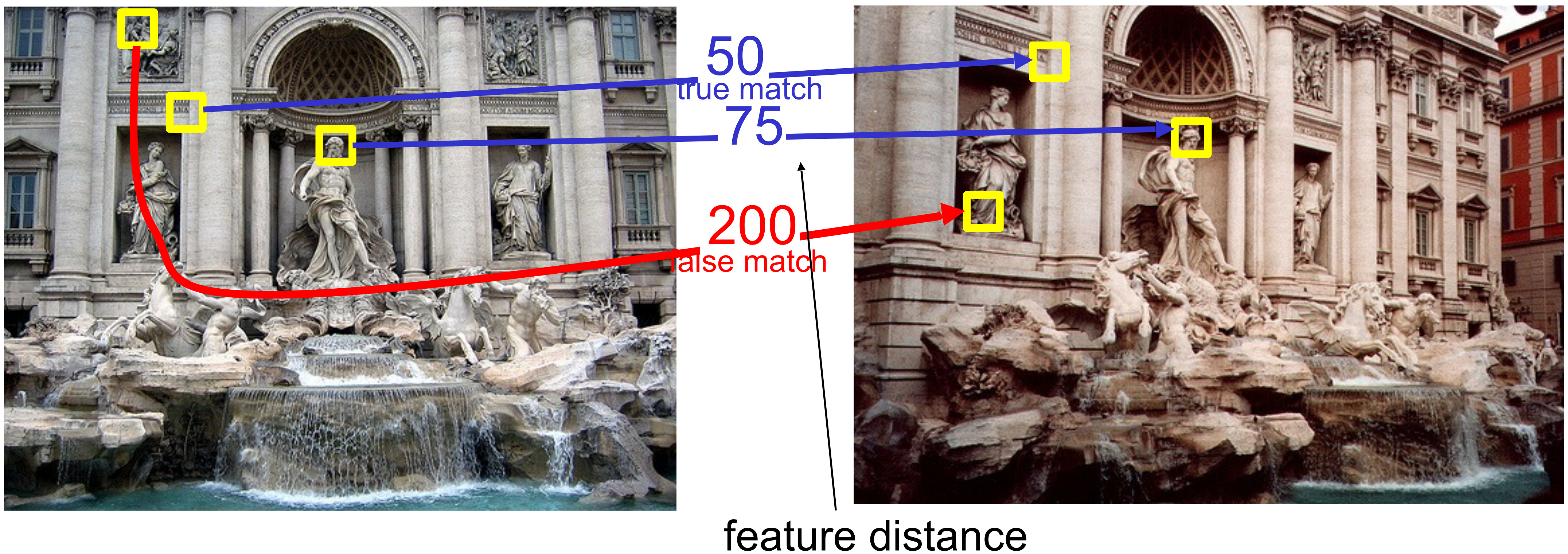


I_1



I_2

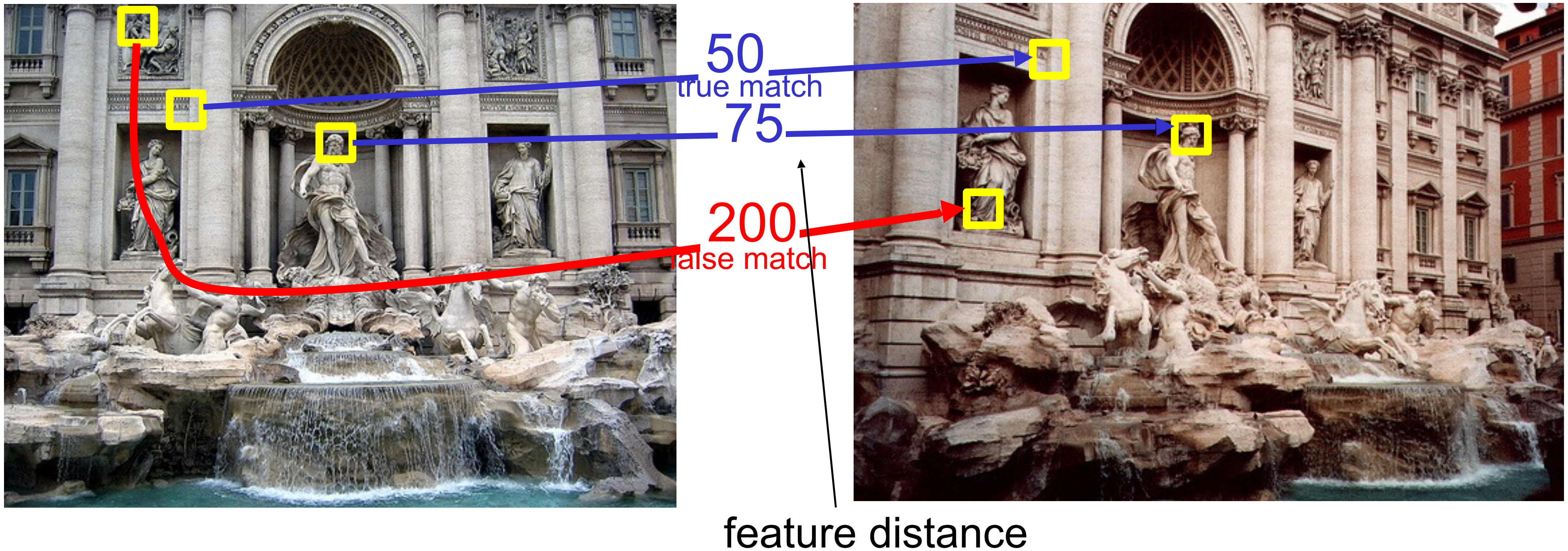
Eliminating bad matches



Throw out features with distance > threshold

- How to choose the threshold?

True/false positives



Throw out features with distance > threshold

The threshold affects performance

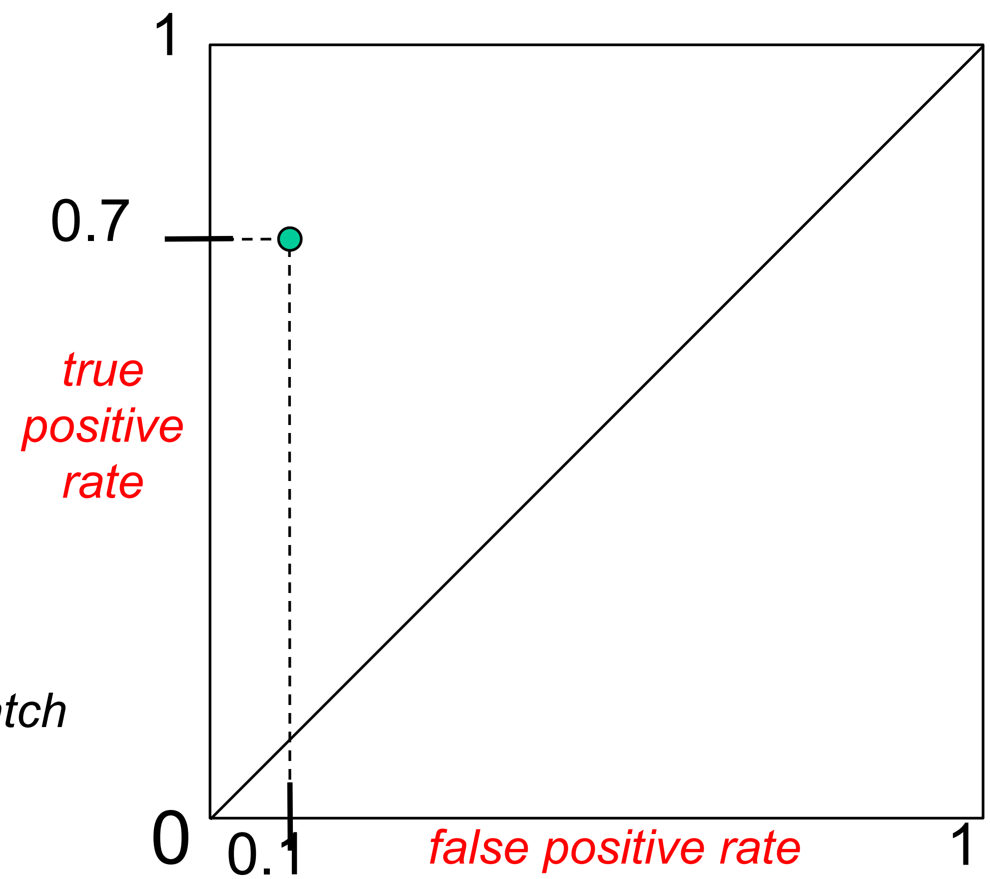
- **True positives** = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- **False positives** = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

Evaluating the results

the matcher correctly found a match

$$\frac{\text{\# true positives matched}}{\text{\# true positives}}$$

features that really do have a match



$$\frac{\text{\# false positives matched}}{\text{\# true negatives}}$$

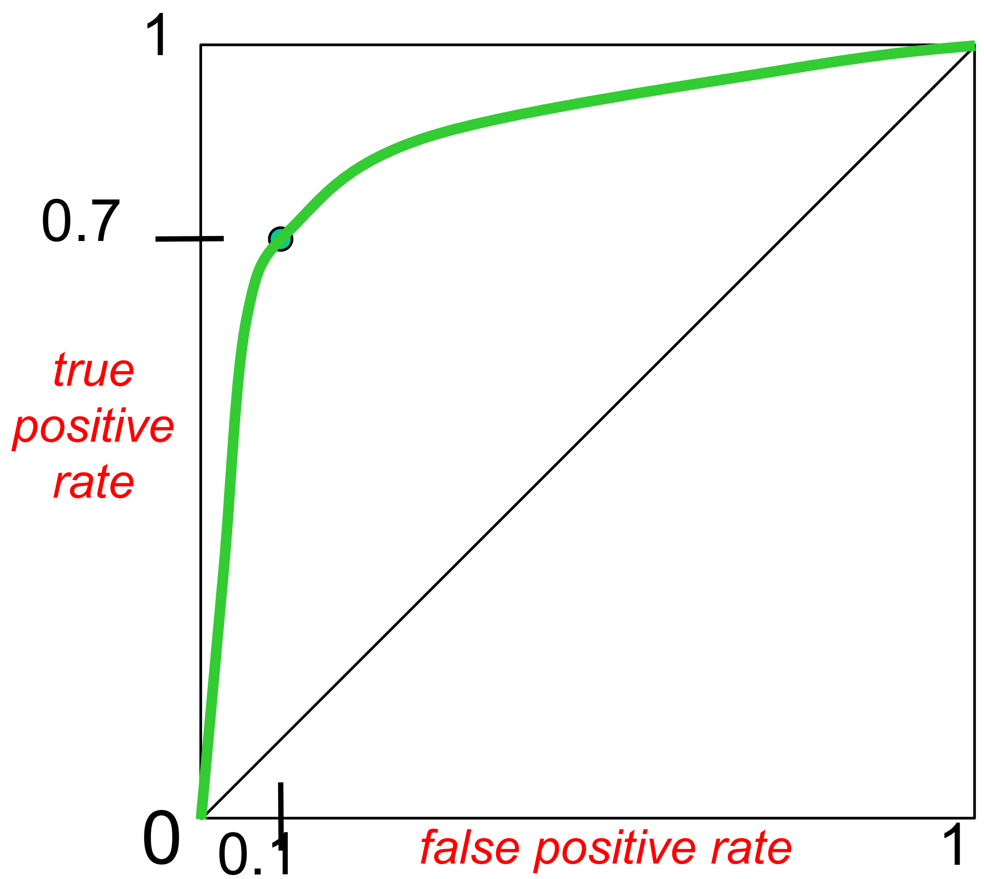
the matcher said yes when the right answer was no

features that really don't have a match

Evaluating the results

How can we measure the performance of a feature matcher?

ROC curve **("Receiver Operator Characteristic")**



$$\frac{\text{\# true positives matched}}{\text{\# true positives}}$$

$$\frac{\text{\# false positives matched}}{\text{\# true negatives}}$$

Different threshold settings give you different points on this graph

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