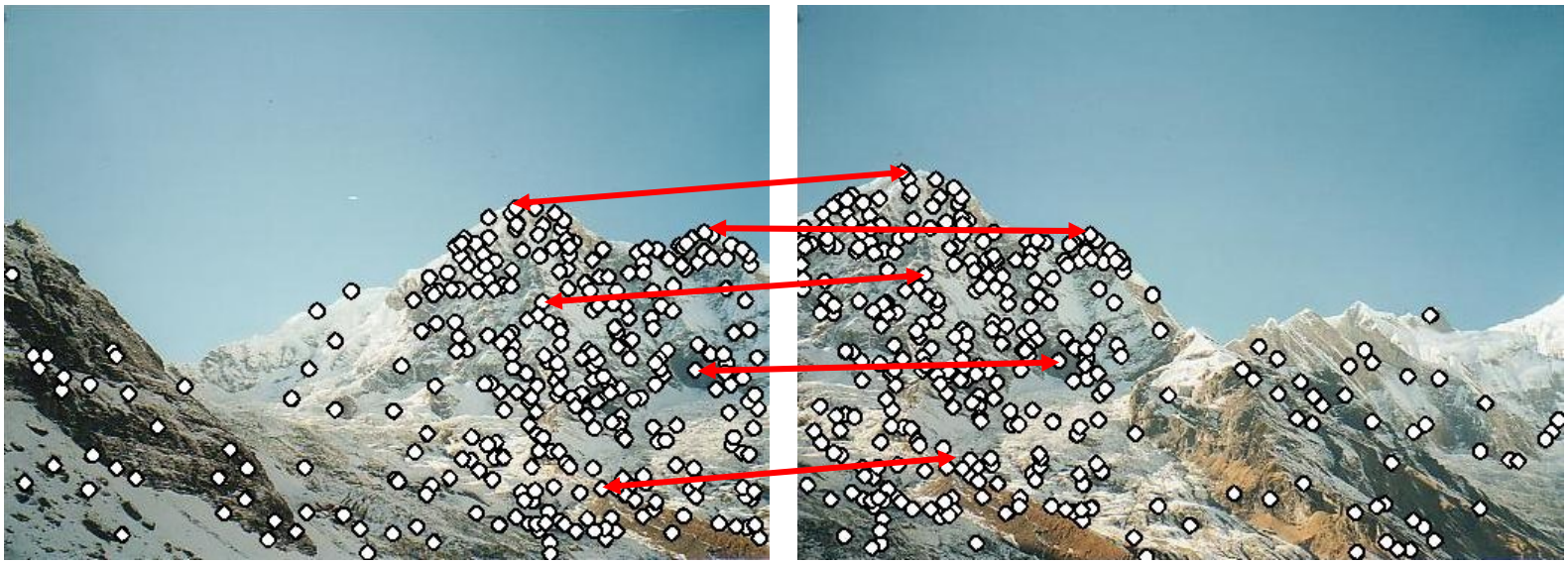
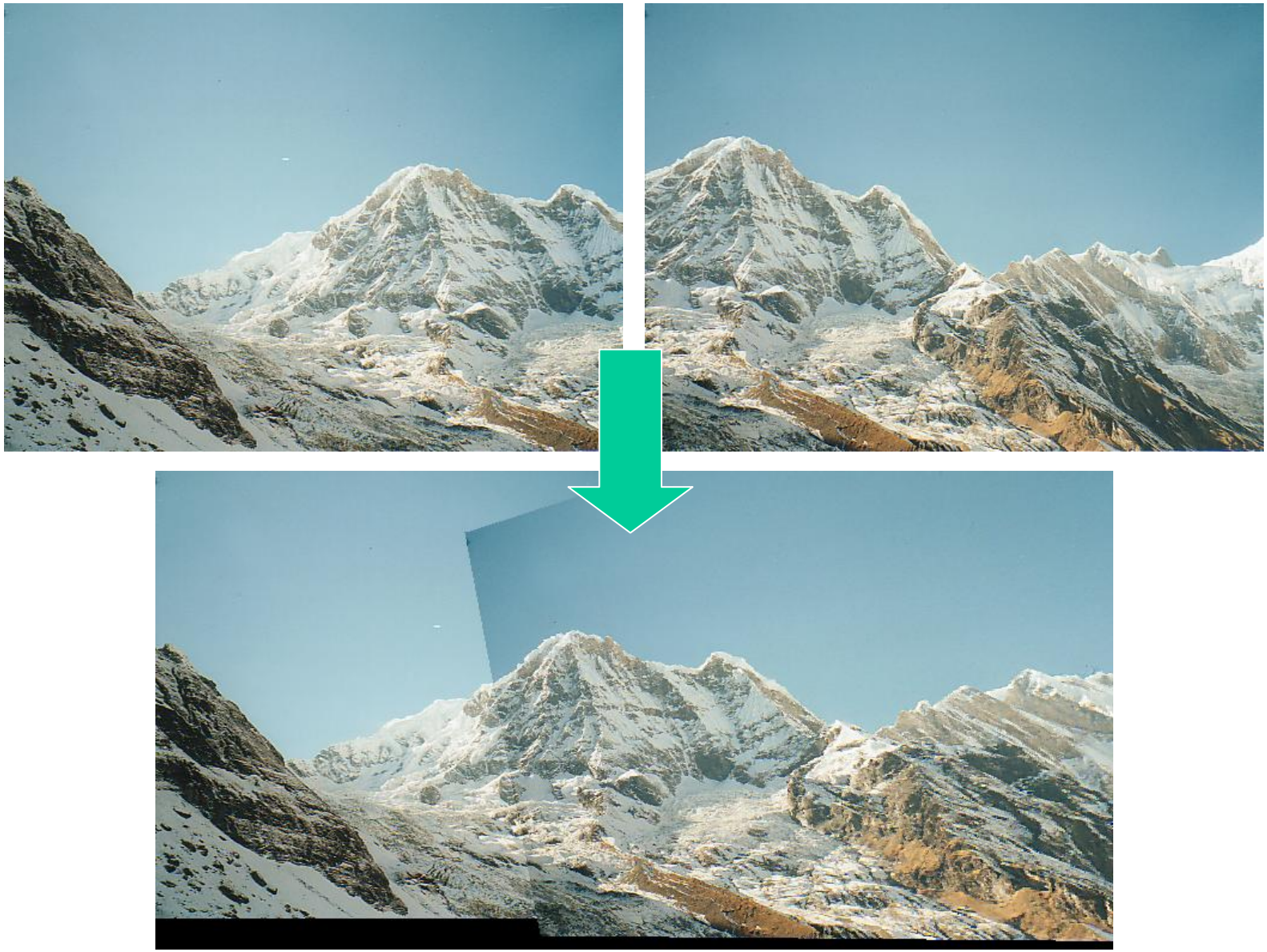
Lecture 6

Features and Image Matching



Suppose you want to create a panorama

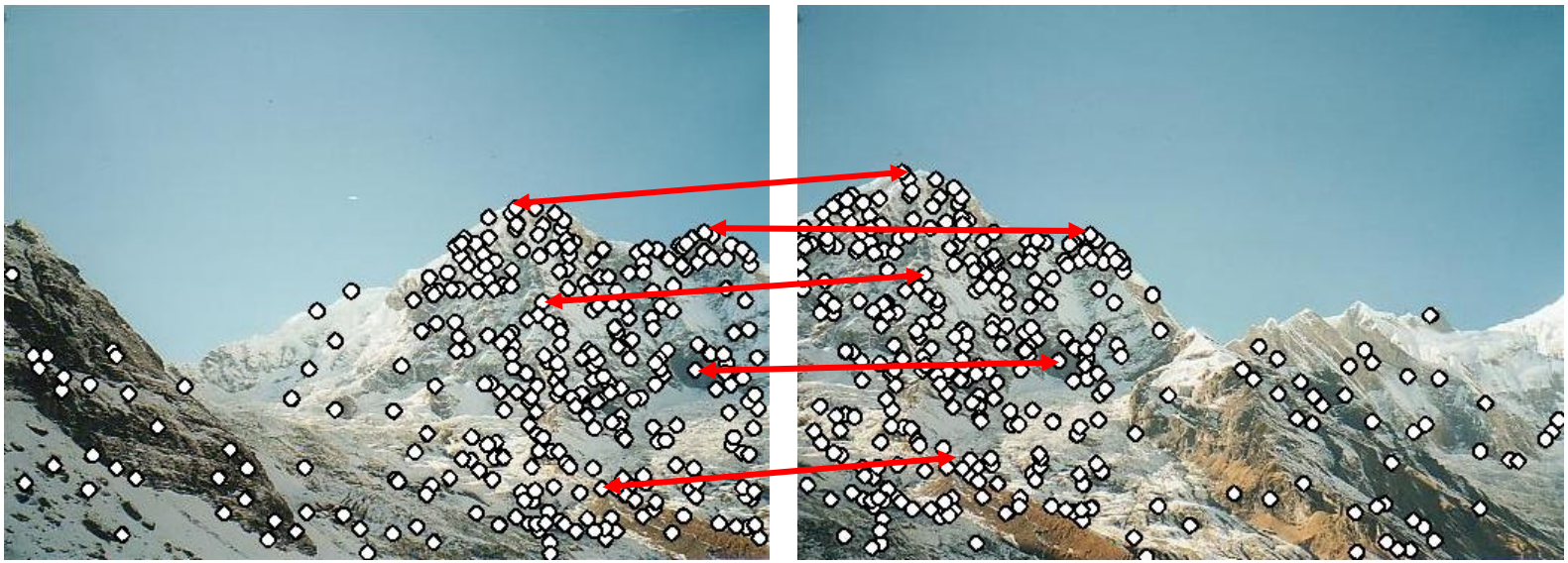


From Matthew Brown

What is the first step?

Need to match portions of images

Solution: Match image regions using local features



Another example

by Diva Sian

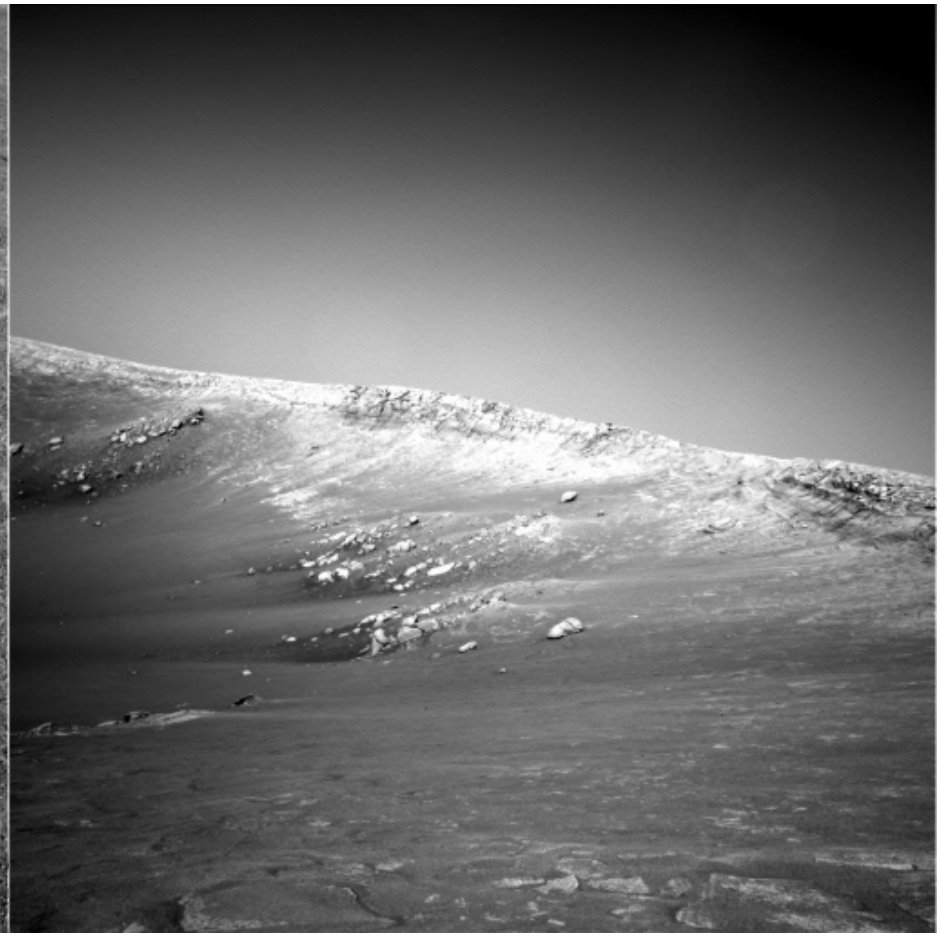
by swashford

Harder case

by Diva Sian

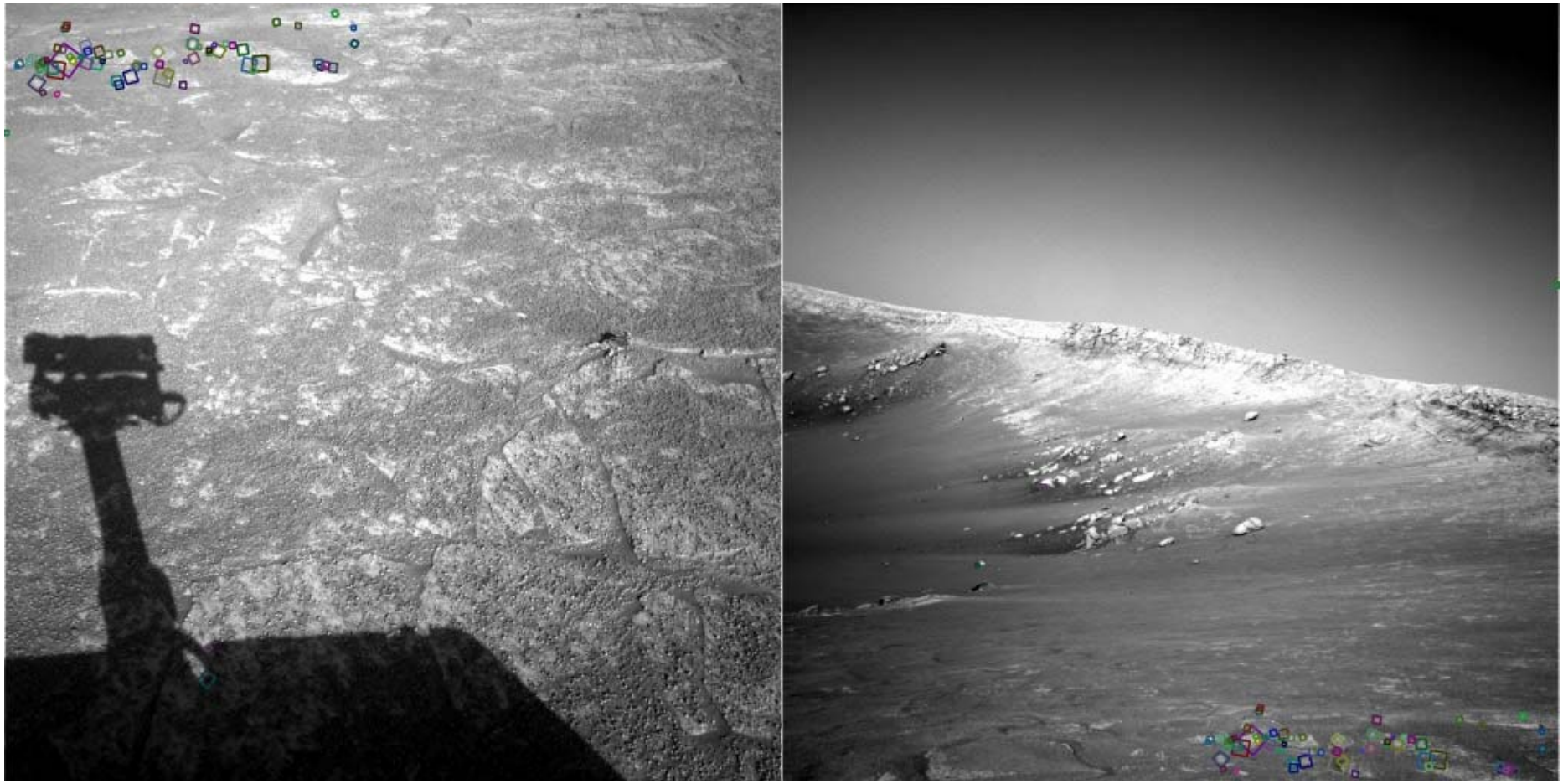
by scgbt

Harder still?



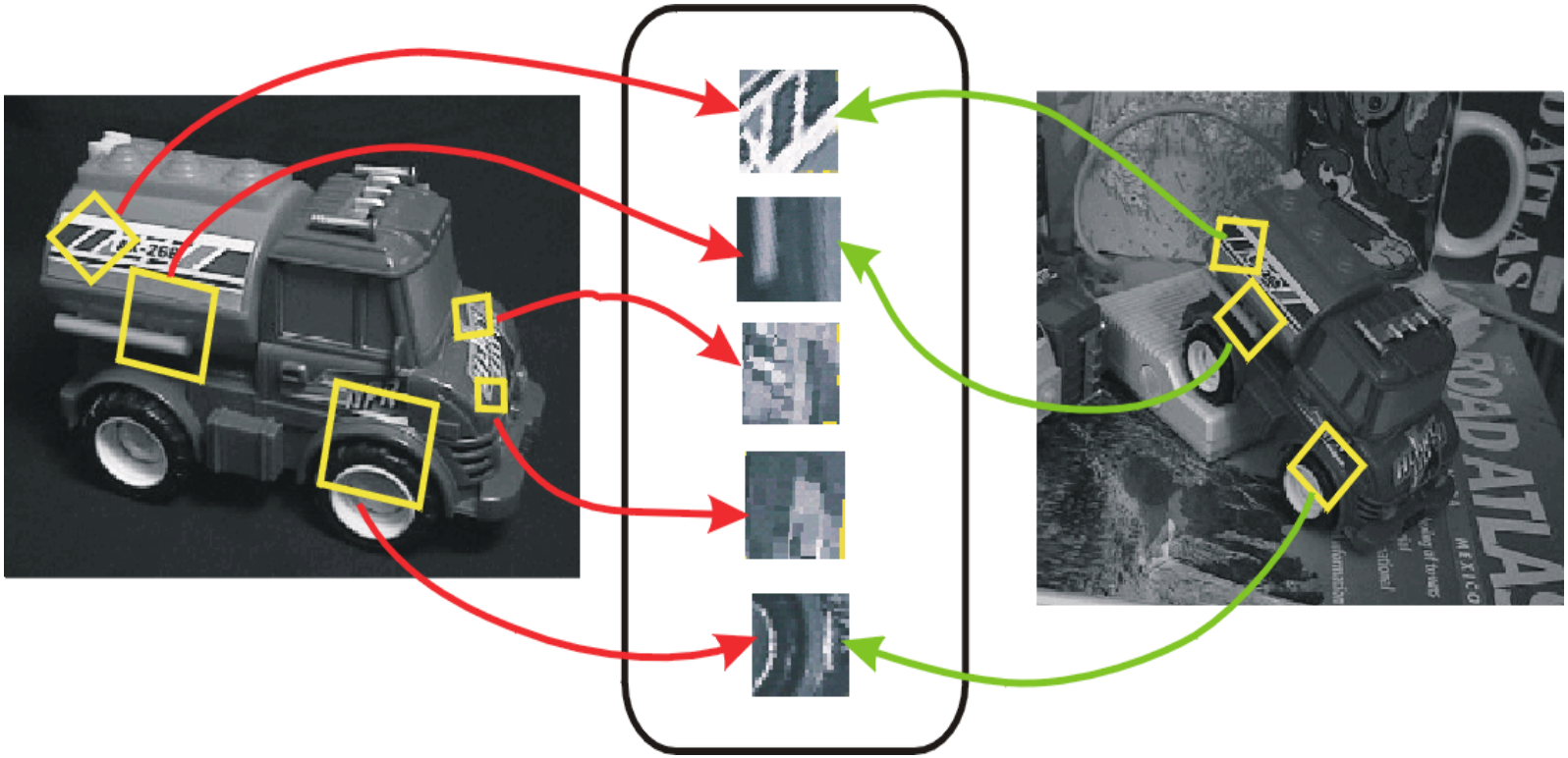
NASA Mars Rover images

Answer below (look for tiny colored squares…)



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

Features can also be used for object recognition



Feature Descriptors

Why local features

Locality

- features are local, so robust to occlusion and clutter

Distinctiveness:

- can differentiate a large database of objects

Quantity

- hundreds or thousands in a single image

Efficiency

- real-time performance achievable

Generality

- exploit different types of features in different situations

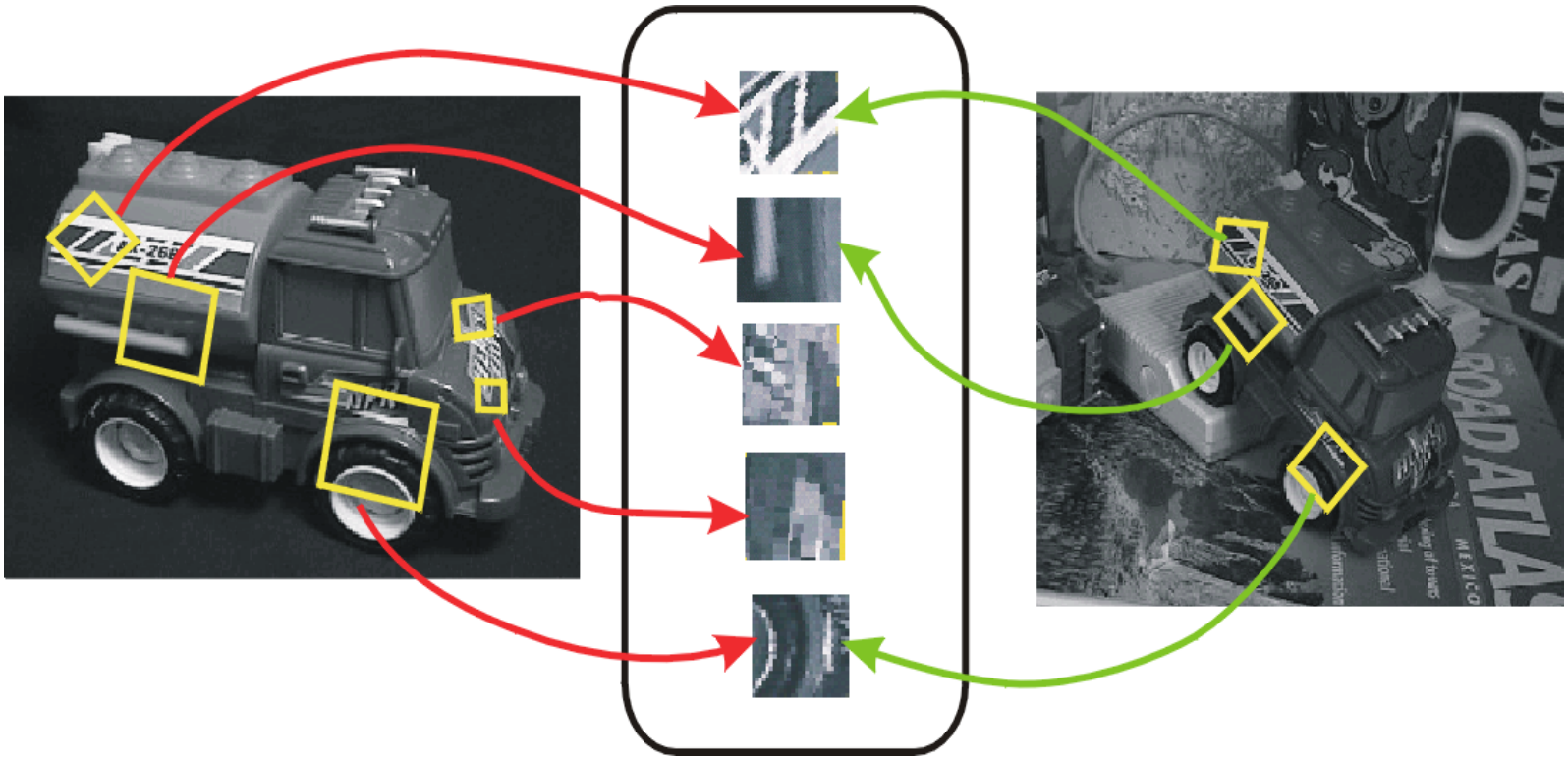
Applications

Features are used for:

- Image alignment (e.g., panoramic mosaics)
- Object recognition
- 3D reconstruction (e.g., stereo)
- Motion tracking
- Indexing and content-based retrieval
- Robot navigation
- …

Want features that are invariant to transformations

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



Feature Descriptors

What about edges?

- Edges can be invariant to brightness changes but typically not invariant to other transformations

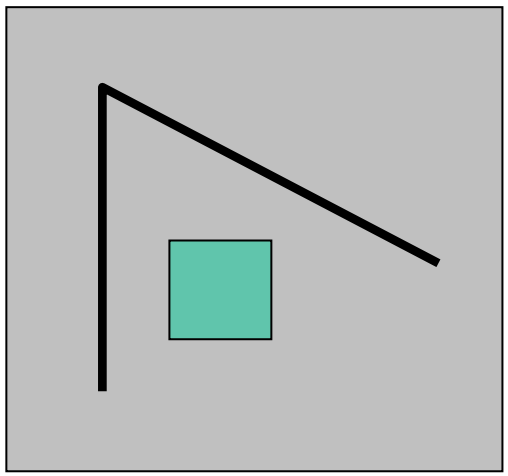
What makes a good feature?

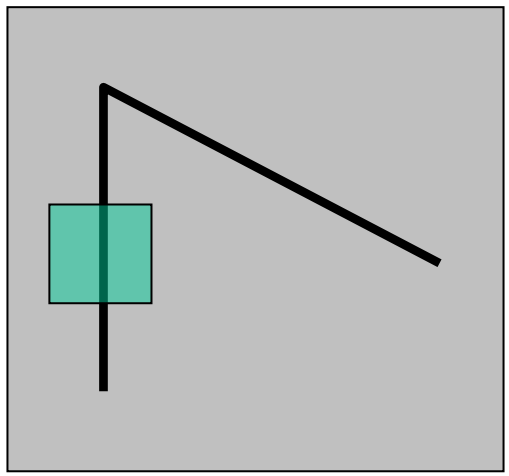
- Want uniqueness
  - Leads to unambiguous matches in other images
- Look for “interest points”: image regions that are unusual
- How to define “unusual”?

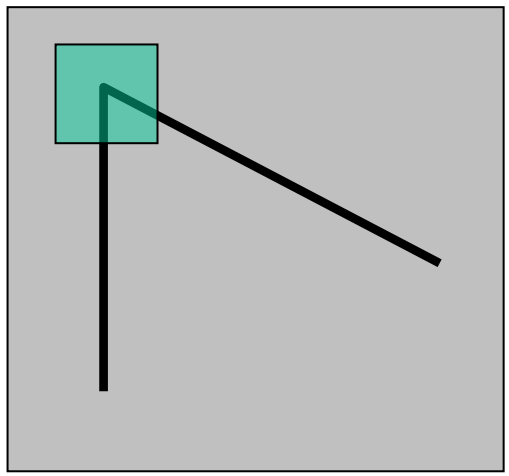
Finding interest points in an image

Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?



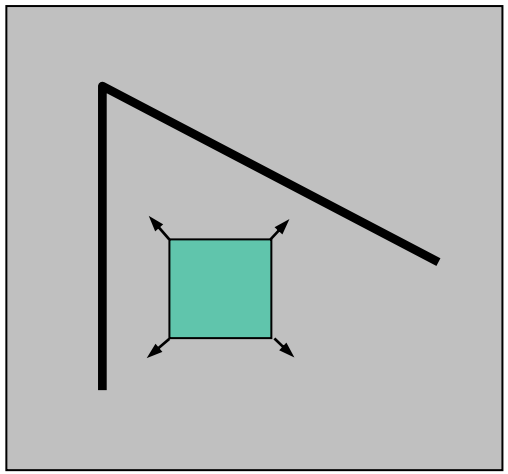




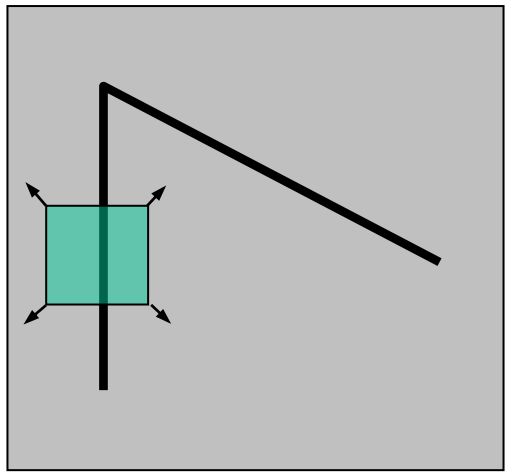
Slide adapted form Darya Frolova, Denis Simakov, Weizmann Institute.

Finding interest points in an image

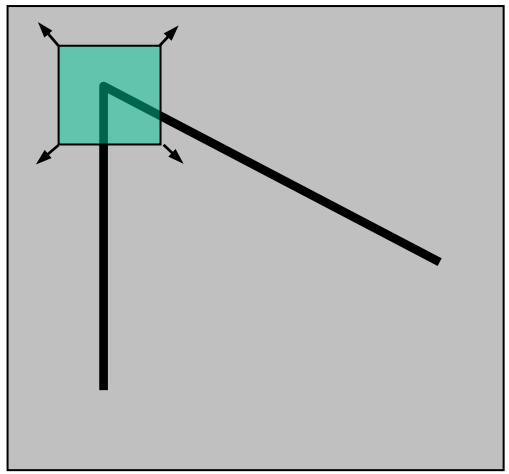
How does the window change when you shift it?



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions, i.e., even the minimum change is large

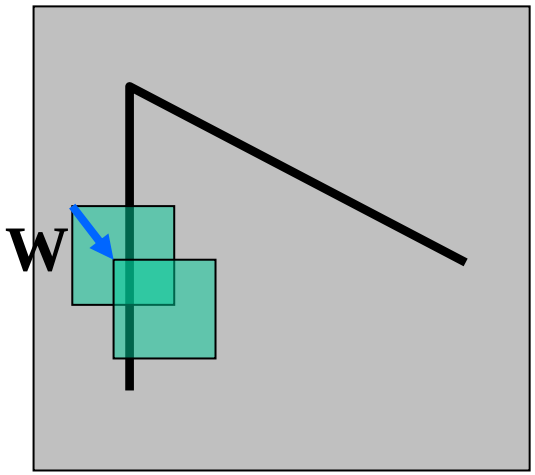
Find locations such that the minimum change caused by shifting the window in any direction is large

Slide adapted form Darya Frolova, Denis Simakov, Weizmann Institute.

Finding interest points (Feature Detection): the math

Consider shifting the window **W** by (u,v)

- how do the pixels in **W** change?
- compare each pixel before and after using the sum of squared differences (SSD)
- this defines an SSD "error" *E(u,v)*:



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

Small motion assumption

Taylor Series expansion of *I*:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (*u*,*v*) is small, then first order approx. is good

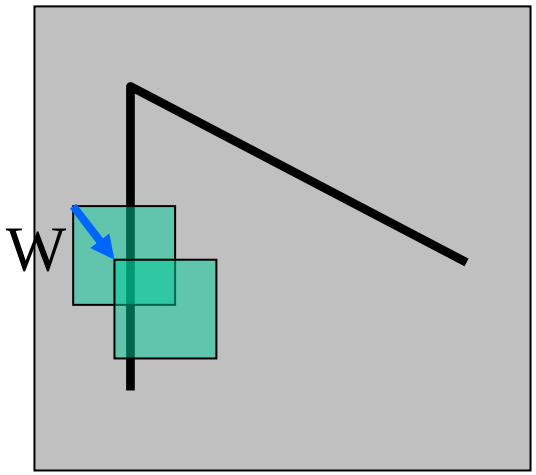
$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x,y) + [I_x \;\; I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

Plugging this into the formula on the previous slide…

Feature detection: the math



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2$$

$$\approx \sum_{(x,y)\in W} [I(x,y) + [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)]^2$$

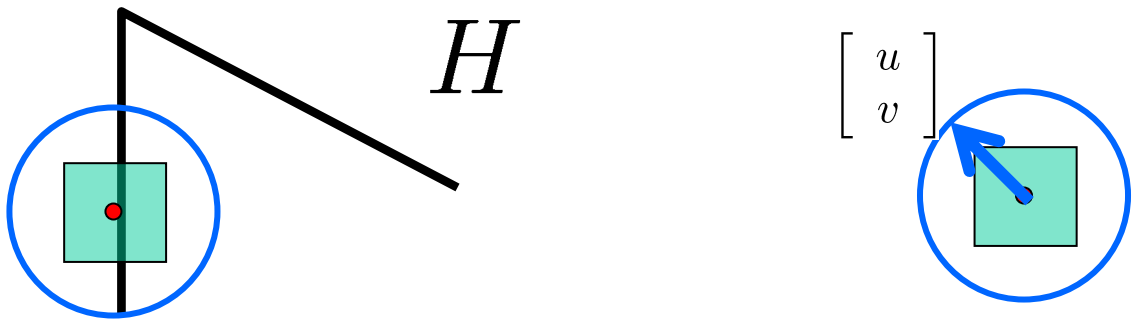
$$\approx \sum_{(x,y)\in W} \left[[I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$

Feature detection: the math

This can be rewritten:

$$E(u,v) \approx [u \;\; v] \left(\sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

$$H$$



For the example above:

- You can move the center of the green window to anywhere on the blue unit circle
- How do we find directions that will result in the largest and smallest E values?
- Find these directions by looking at the eigenvectors of ***H***

Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

$$Ax = \lambda x$$

The scalar λ is the **eigenvalue** corresponding to **x**

- The eigenvalues are found by solving:

$$det(A - \lambda I) = 0$$

- In our case, ***A*** = ***H*** is a 2x2 matrix, so we have

$$det \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

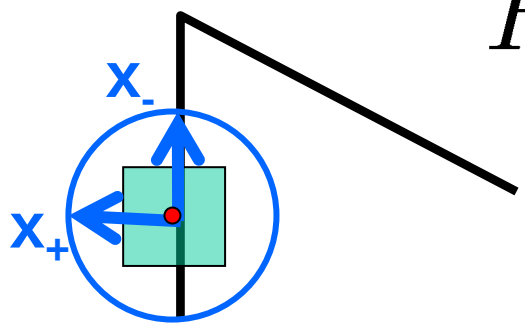
$$\lambda_{\pm} = \frac{1}{2} \left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2} \right]$$

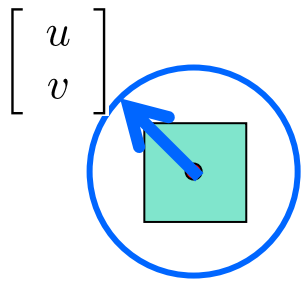
Once you know λ, you find **x** by solving

$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \left| \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right. = 0$$

Feature detection: the math

$$E(u,v) \approx [u \;\; v] \underbrace{\left(\sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right)}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$





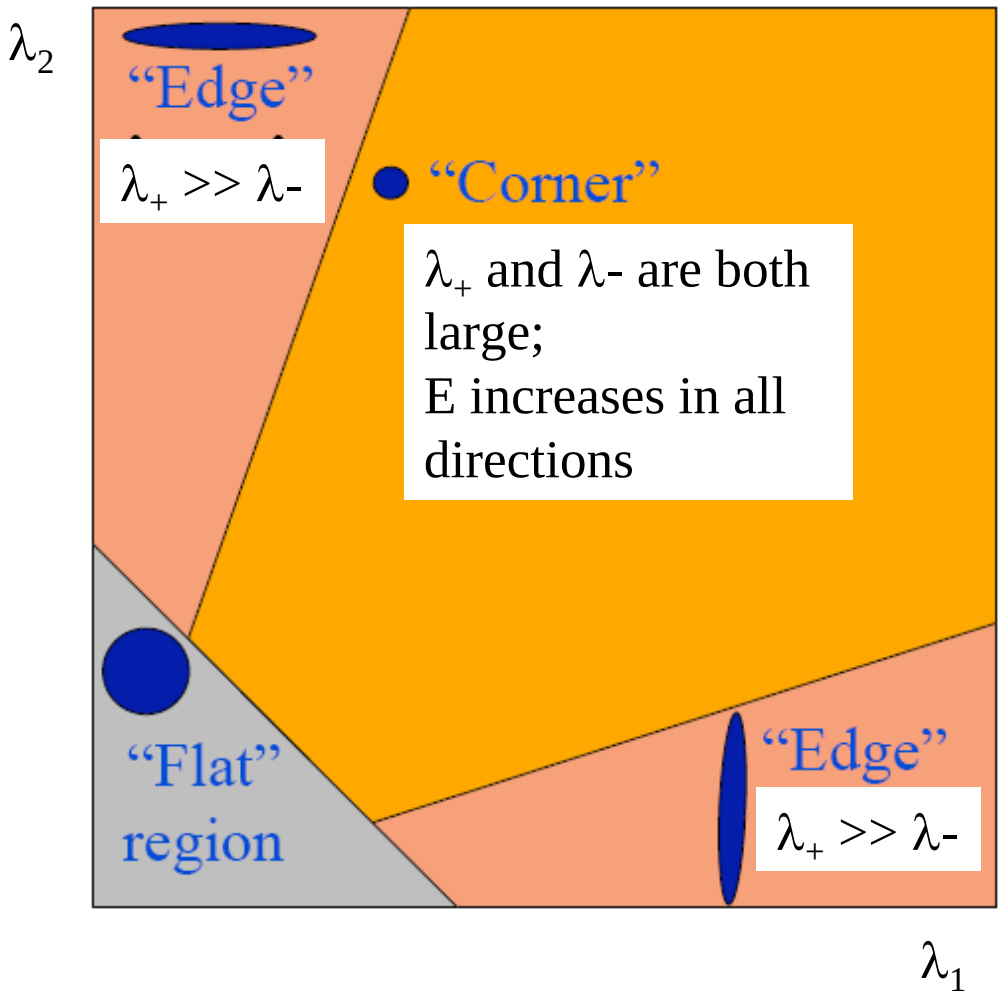
Eigenvalues and eigenvectors of H

- Capture shifts with the smallest and largest change (E value)
- x_+ = direction of **largest** increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of **smallest** increase in E.
- λ_- = amount of increase in direction x_-

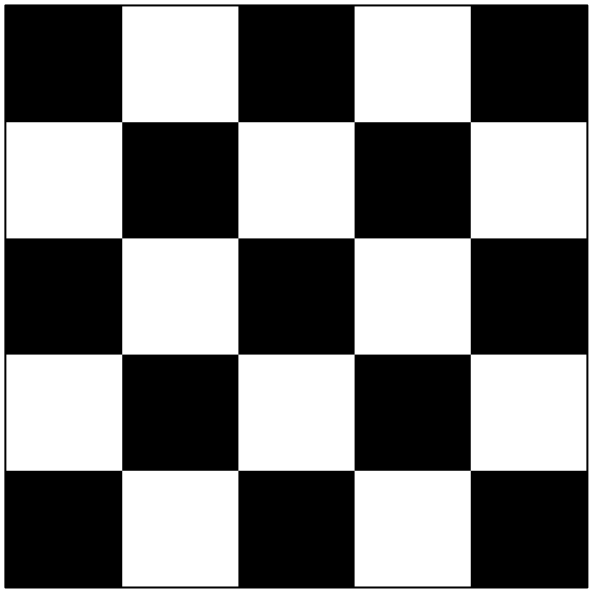
$$Hx_+ = \lambda_+ x_+$$
$$Hx_- = \lambda_- x_-$$

Feature detection: the math

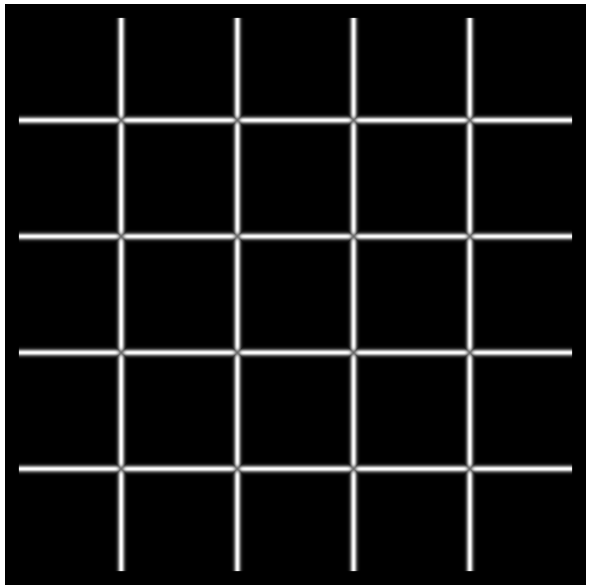
How are λ_+, $\mathbf{x}_+$, λ_-, and $\mathbf{x}_-$ relevant for feature detection?



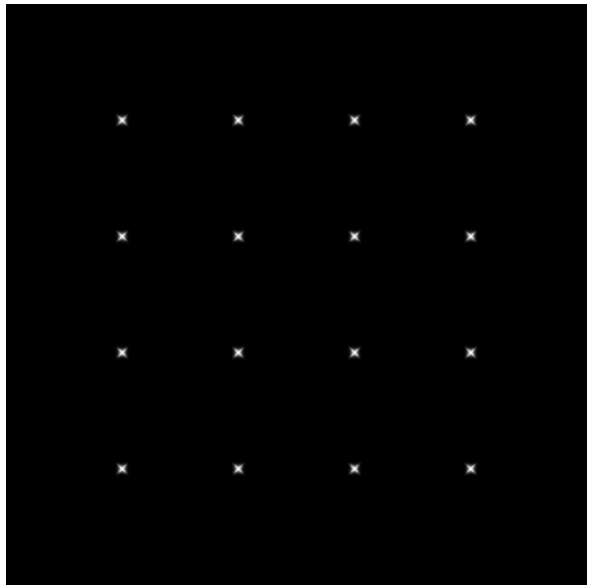
Example



I



λ_+

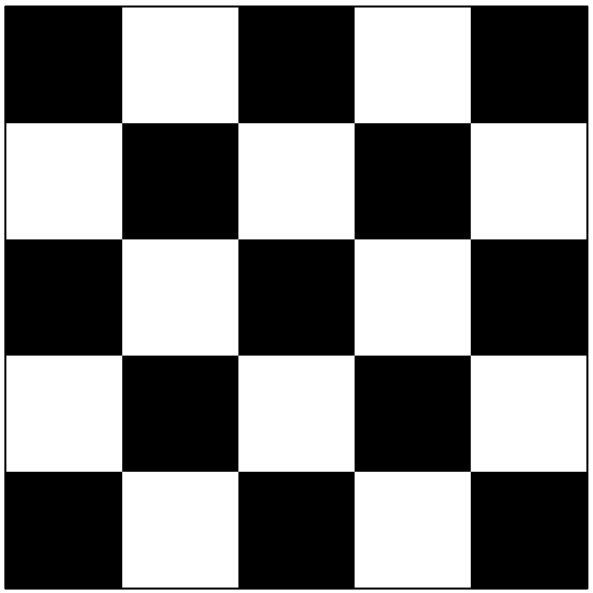


λ_-

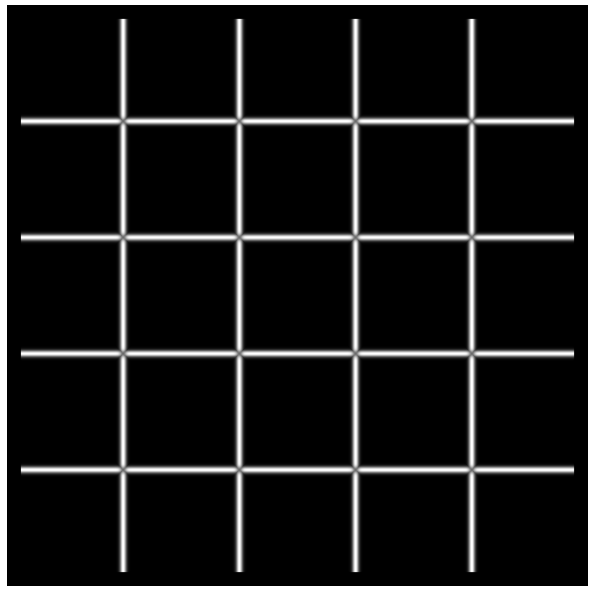
Feature detection: the math

Want *E(u,v)* to be ***large*** in ***all*** directions

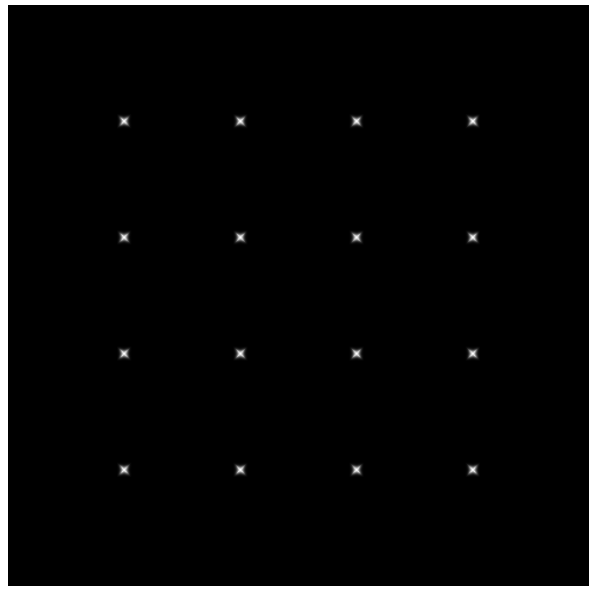
- the *minimum* of *E(u,v)* should be large over all unit vectors [u v]
- this minimum is given by the smaller eigenvalue λ_- of ***H***
- Look for large values of λ-



I



λ_+

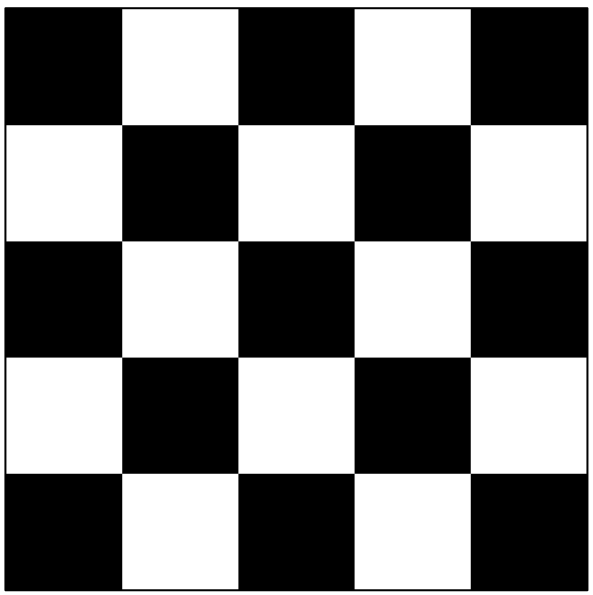


λ_-

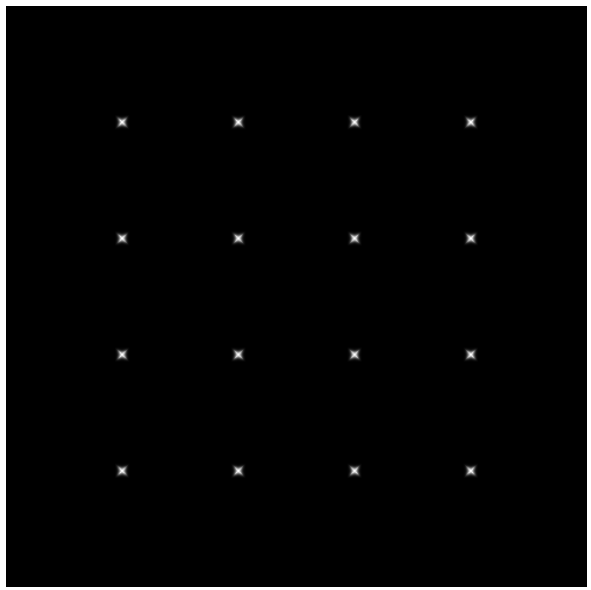
Feature detection (interest point detection) summary

Here's what you do

- Compute the gradient at each point in the image
- Create the ***H*** matrix from the entries in the gradient
- Compute the eigenvalues
- Find points with large λ_- (i.e., λ_- > threshold)
- Choose points where λ_- is a local maximum as interest points



I

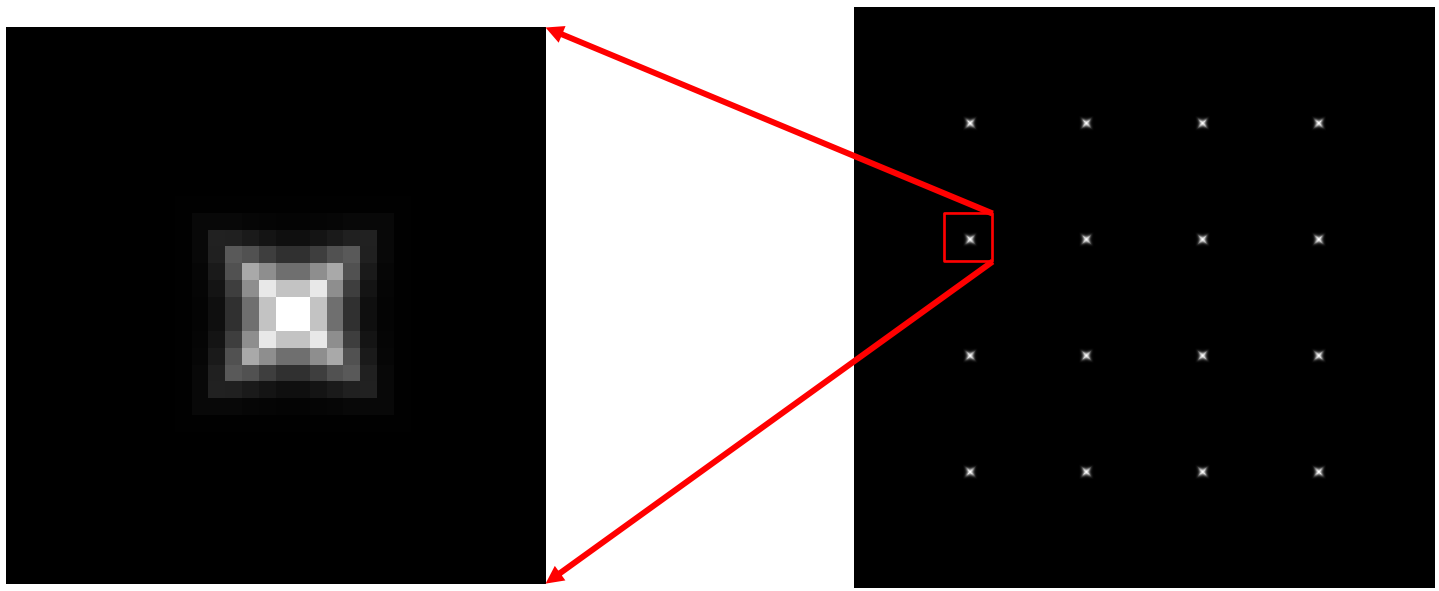


λ_-

Feature detection summary

Here's what you do

- Compute the gradient at each point in the image
- Create the ***H*** matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features (interest points)



λ_-

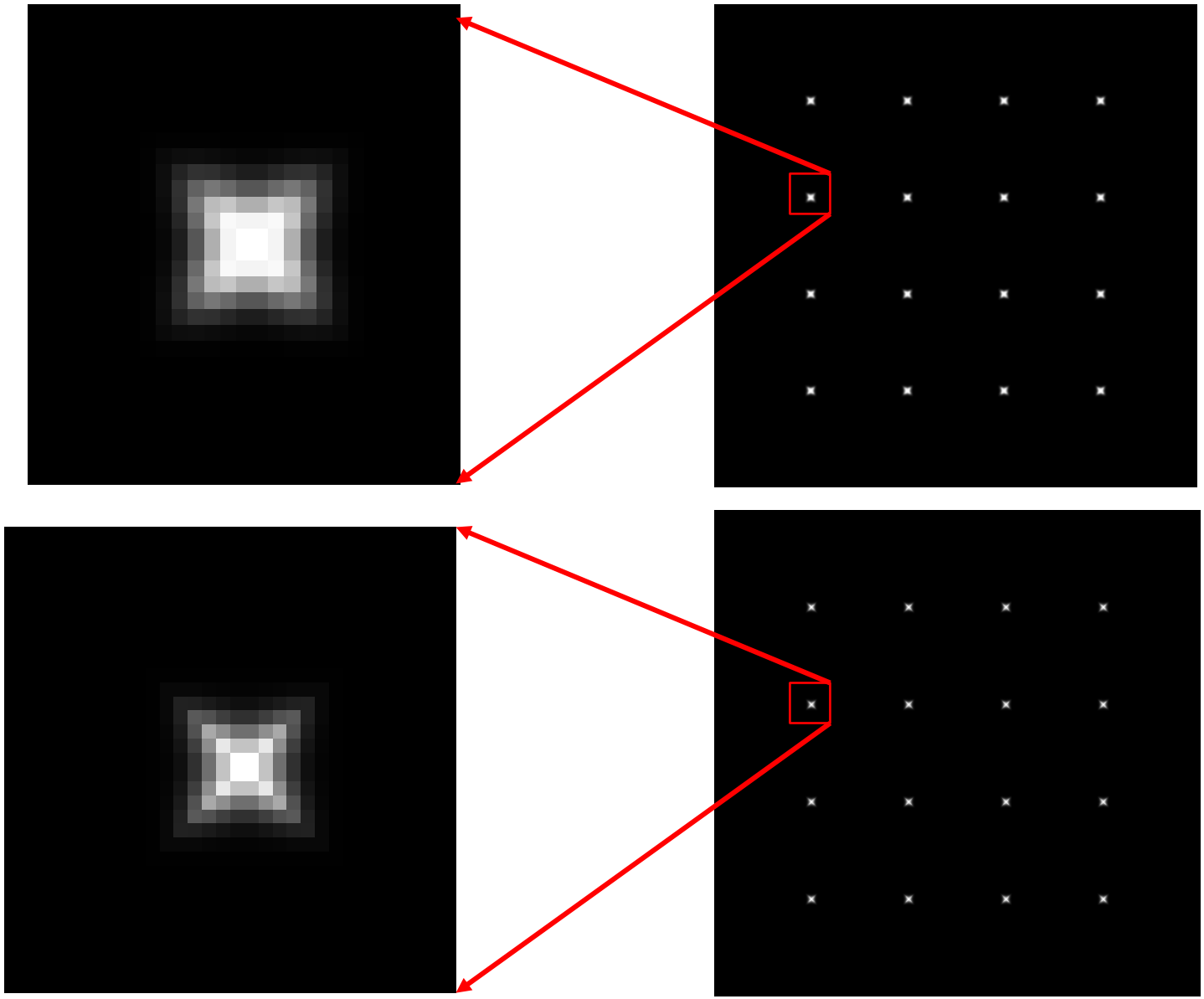
The Harris operator

λ_- is a variant of the "Harris operator" for feature detection

$$f_{Harris} = \lambda_+\lambda_- - k(\lambda_+ + \lambda_-)^2 = (h_{11}h_{22} - h_{12}h_{21}) - k(h_{11} + h_{22})^2$$
$$= \det(H) - k\ \mathrm{trace}(H)^2$$

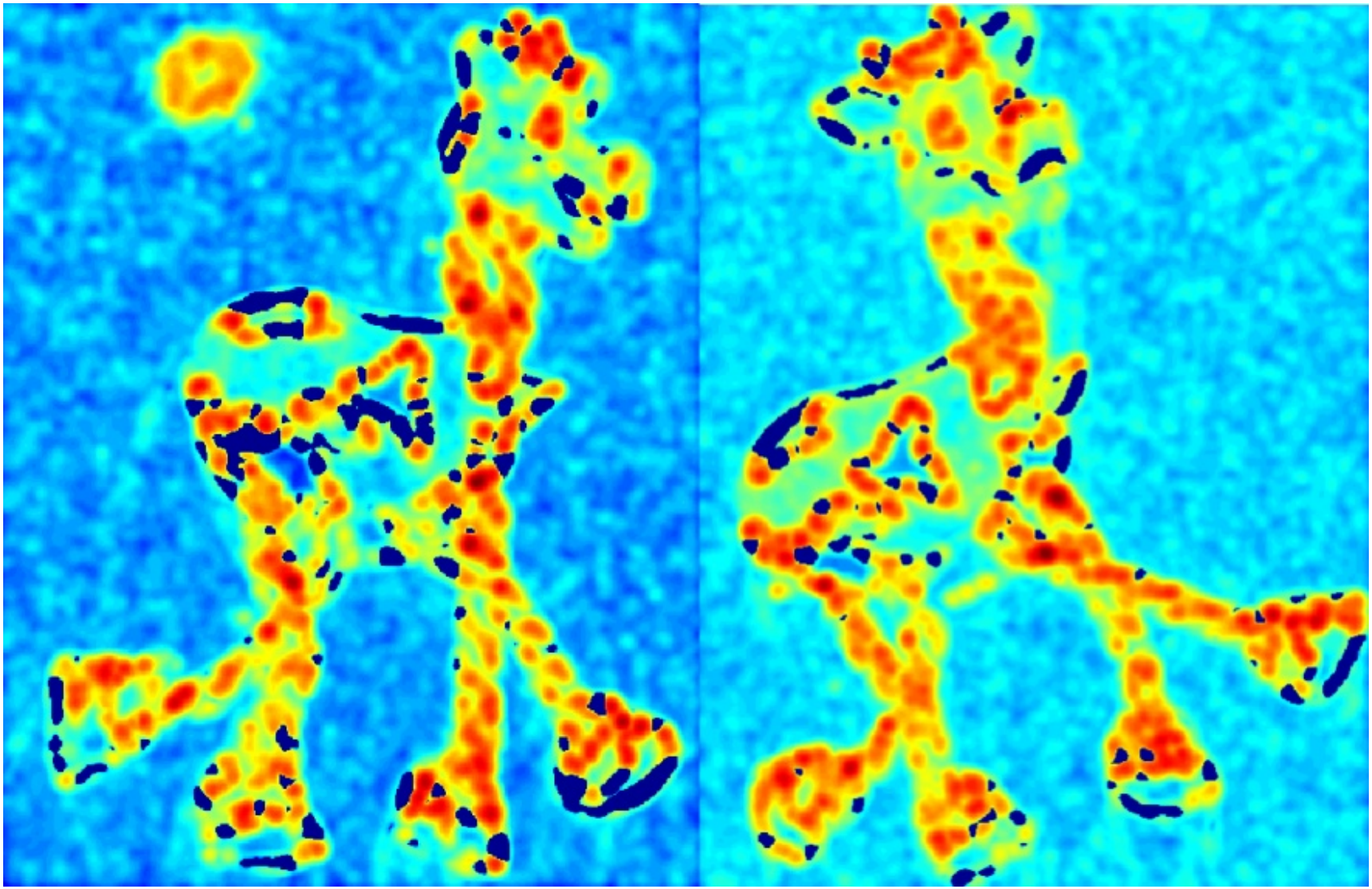
- det is the determinant; trace = sum of diagonal elements of a matrix
- Very similar to λ_- but less expensive (no eigenvalue computation)
- Called the "Harris Corner Detector" or "Harris Operator"
- Most popular among all detectors

The Harris operator

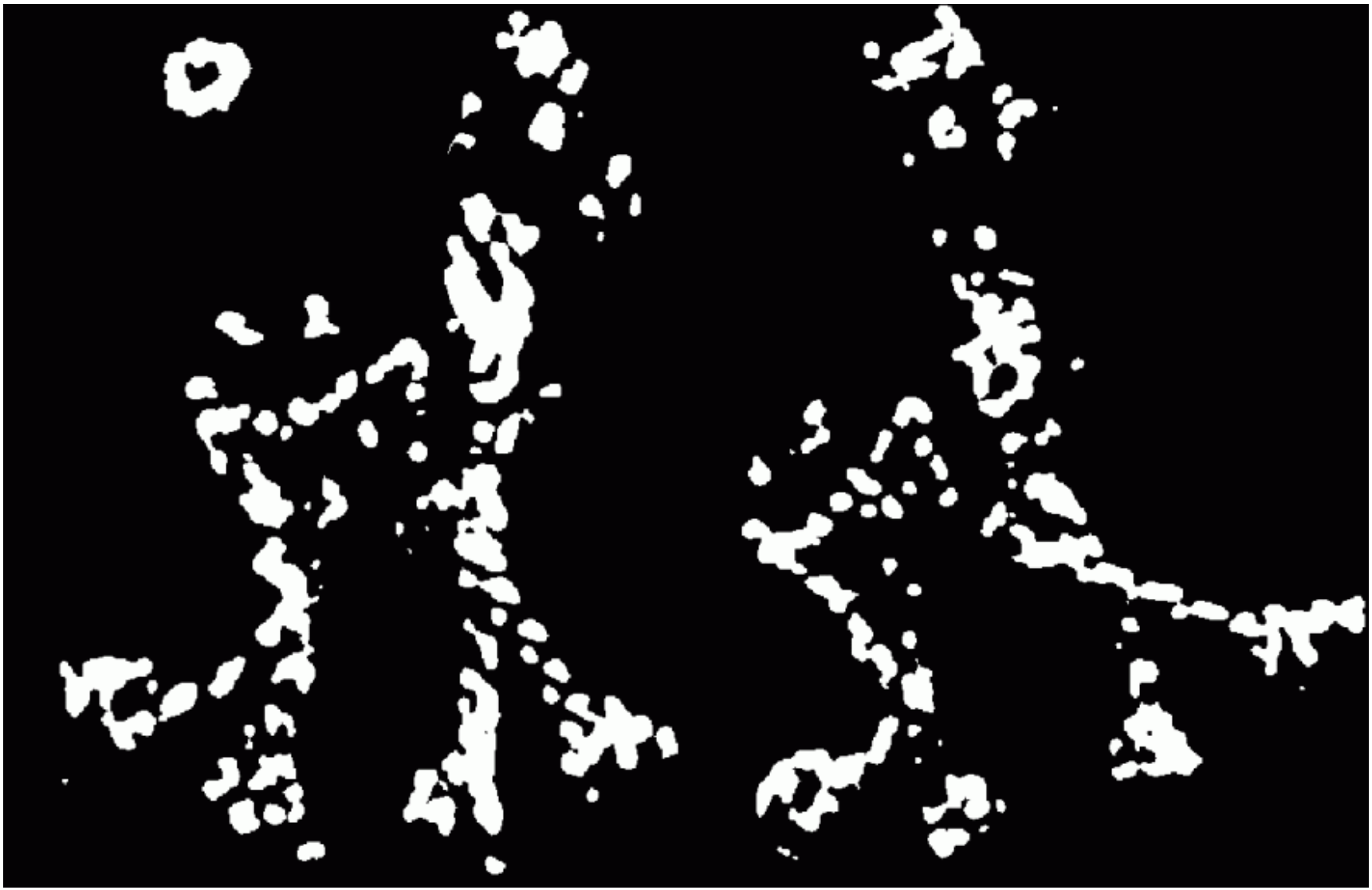


Harris detector example

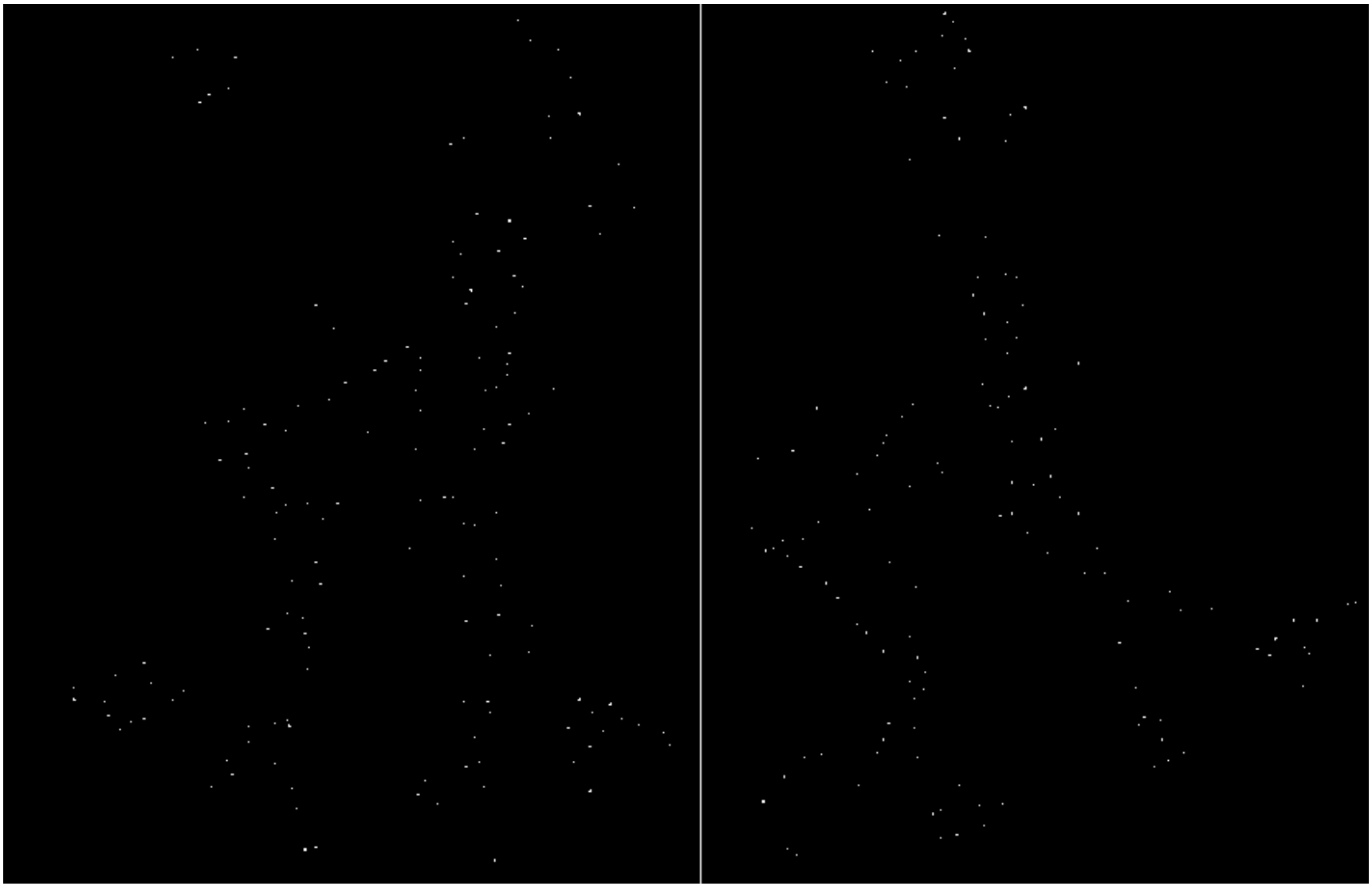
f_{Harris} value (red high, blue low)



Threshold (f_{Harris} > threshold value)



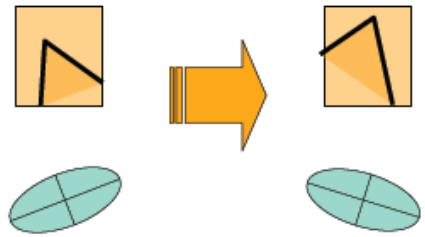
Find local maxima of f_{Harris}



Harris features (in red)

Invariance of Eigenvalue-based feature detectors

Suppose you **rotate** the image by some angle



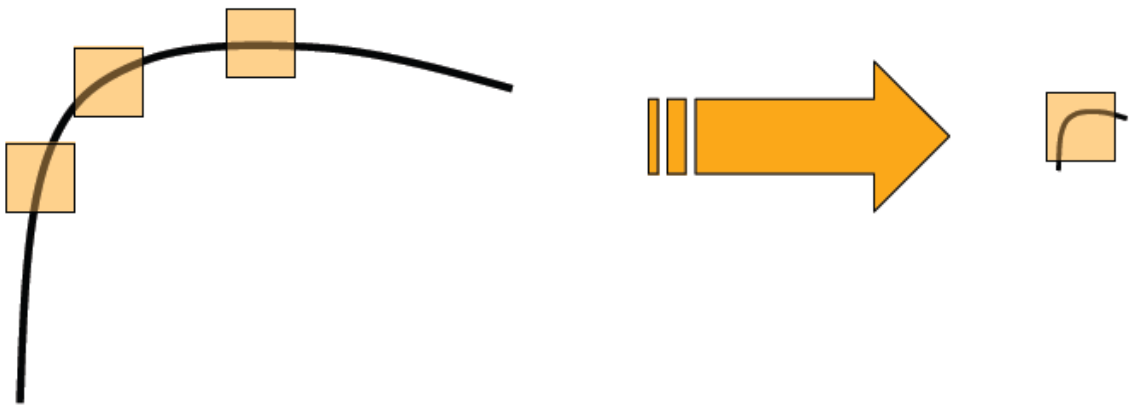
- Will you still pick up the same feature points?
- Yes (since eigenvalues remain the same)

What if you change the brightness?

- Will you still pick up the same feature points?
- Mostly yes (uses gradients which involve pixel differences)

Scale?

- No!

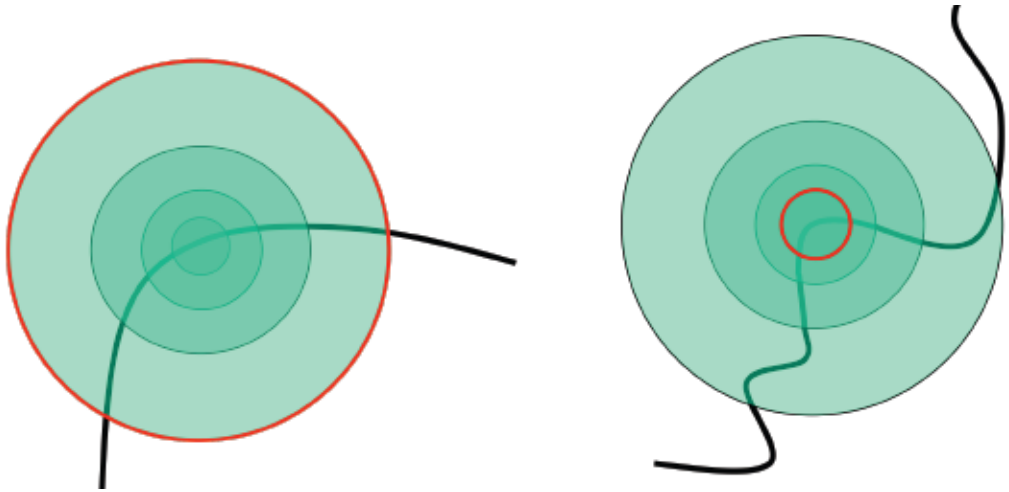


All points will be classified as edges

Corner !

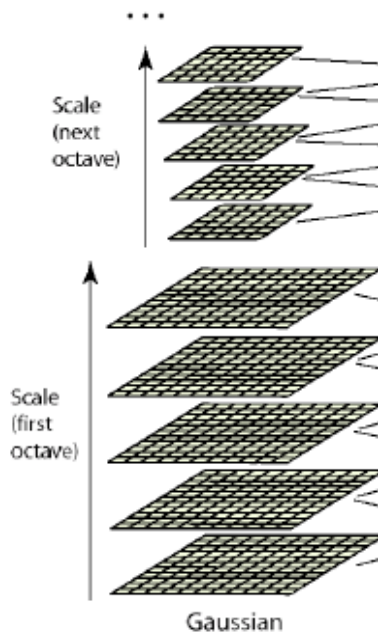
Scale invariant interest point detection

Suppose you're looking for corners using f_{Harris}



Key idea: Find scale that gives local maximum of f_{Harris}

- Generate copies of image at multiple scales by convolving with Gaussians of different σ and using a pyramid
- Find local maxima points by comparing to neighboring points at current and adjacent scales
- Each interest point is a local maximum in both position and scale



See SIFT paper on line for details

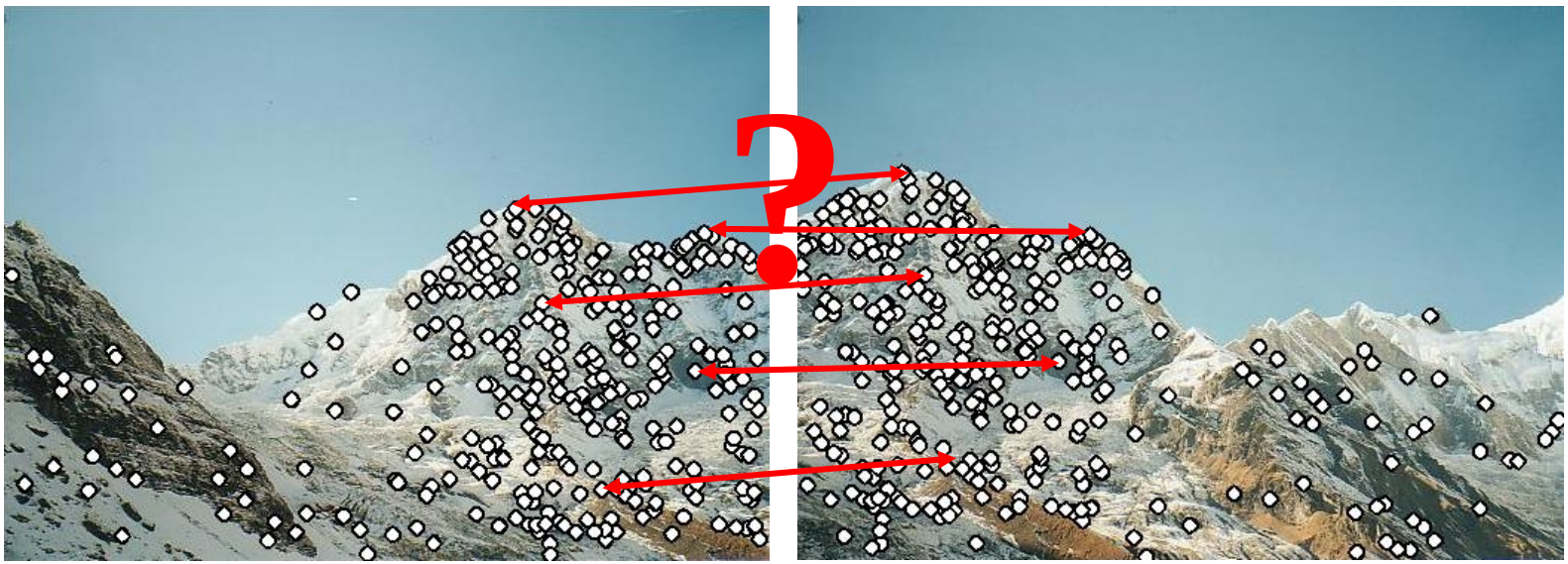
Feature descriptors

We know how to detect good interest points

Next question:

How to match image regions around interest points?

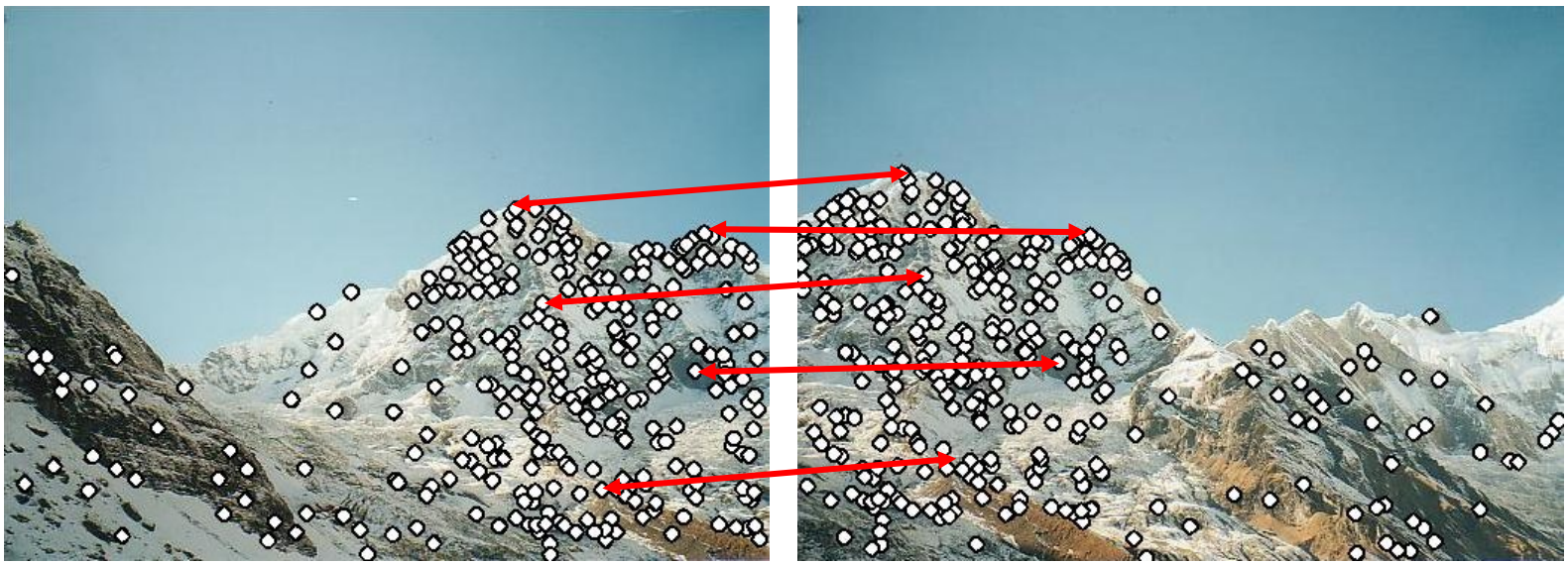
Answer: Need feature descriptors



Feature descriptors

Lots of possibilities (this is a popular research area)

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



How to achieve invariance in image matching

Two steps:

1. Make sure your feature *detector* is invariant
   - Harris is invariant to translation and rotation
   - Scale is trickier
     - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
     - More sophisticated methods find “the best scale” to represent each feature (e.g., SIFT)
2. Design an invariant feature *descriptor*
   - A descriptor captures the intensity 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…

Rotation invariance for feature descriptors

Find dominant orientation of the image patch

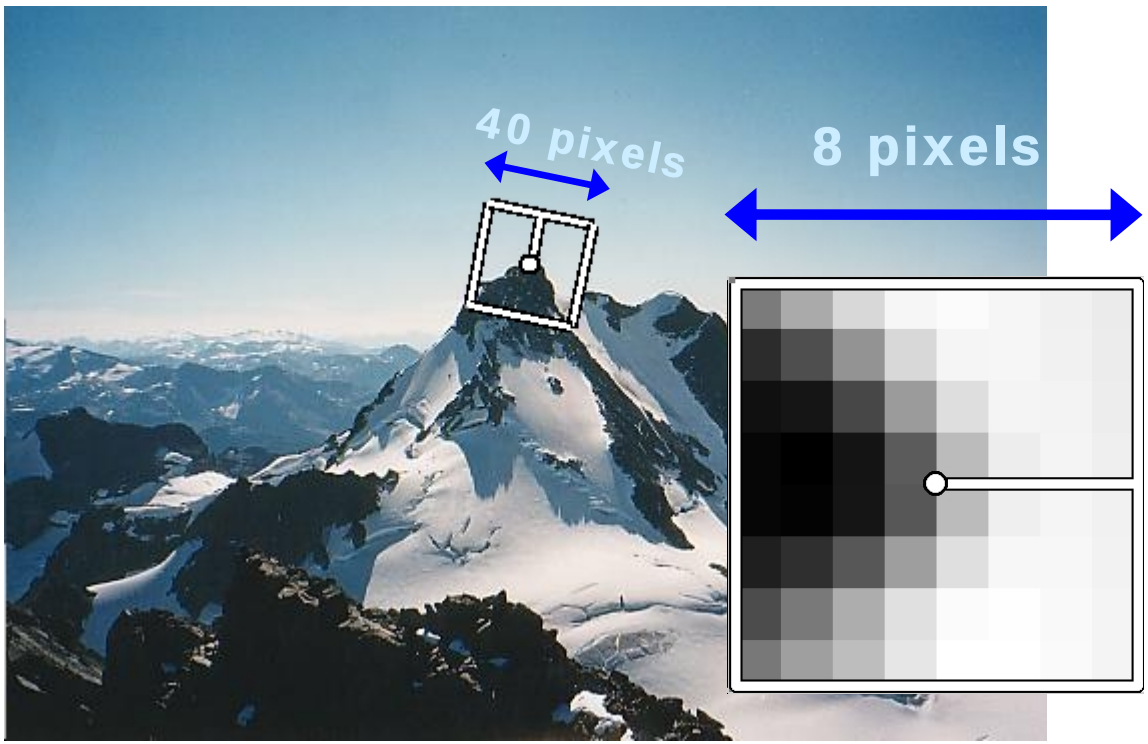
- This is given by $\mathbf{x}_+$, the eigenvector of $\mathbf{H}$ corresponding to λ_+
  - λ_+ is the *larger* eigenvalue
- Rotate the patch according to this angle

Figure by Matthew Brown

Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering) to get 8x8 square window
- Rotate to horizontal
- 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

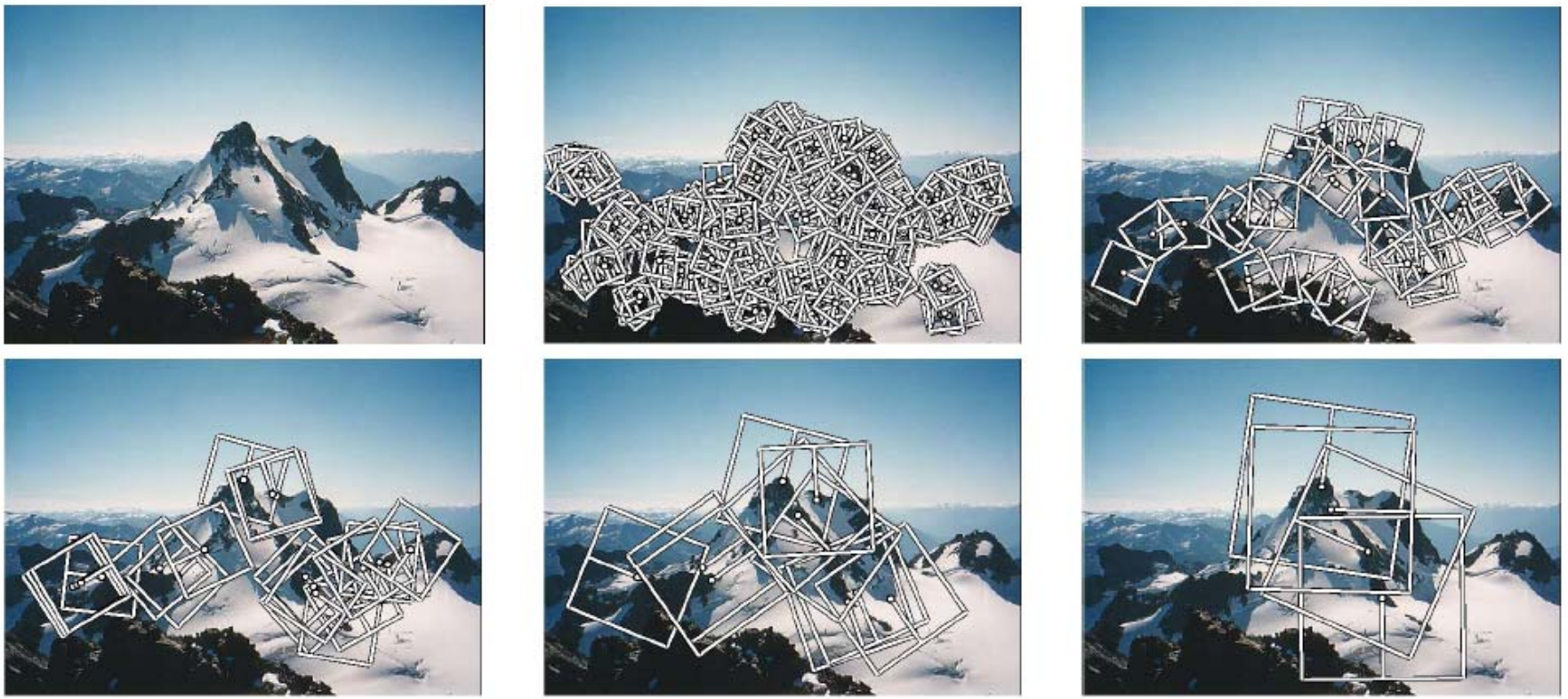
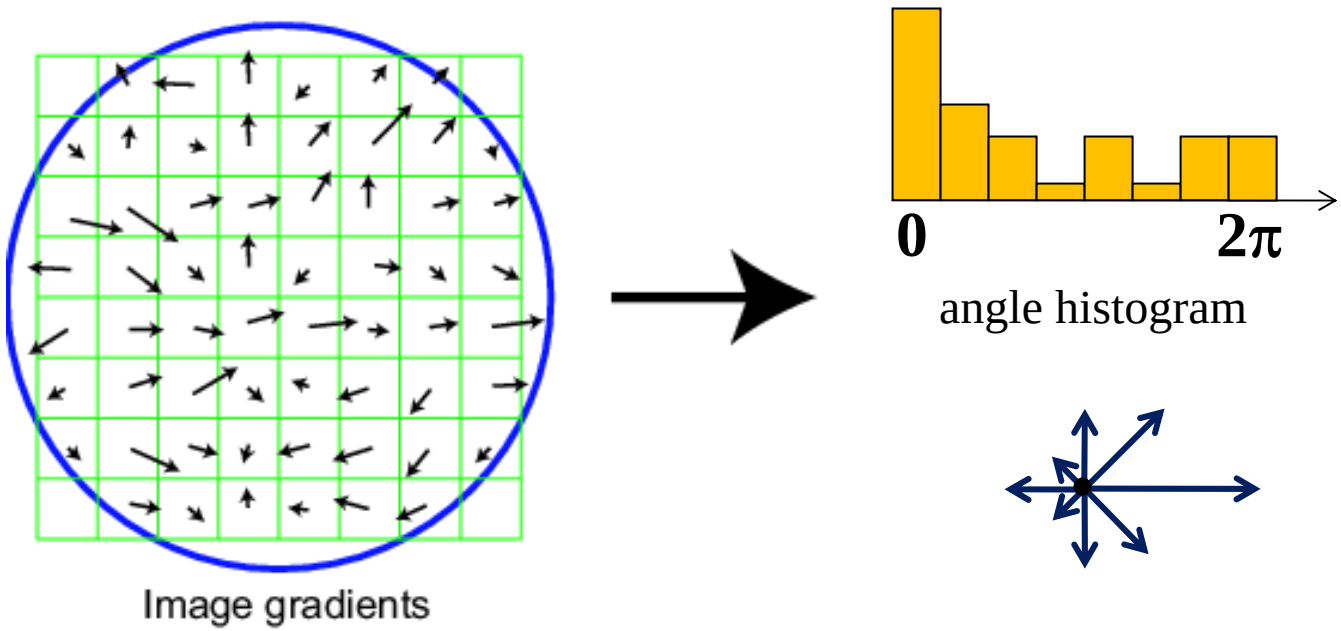


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected interest point (8x8 shown below)
- Compute edge orientation (angle of the gradient minus 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations (8 bins)

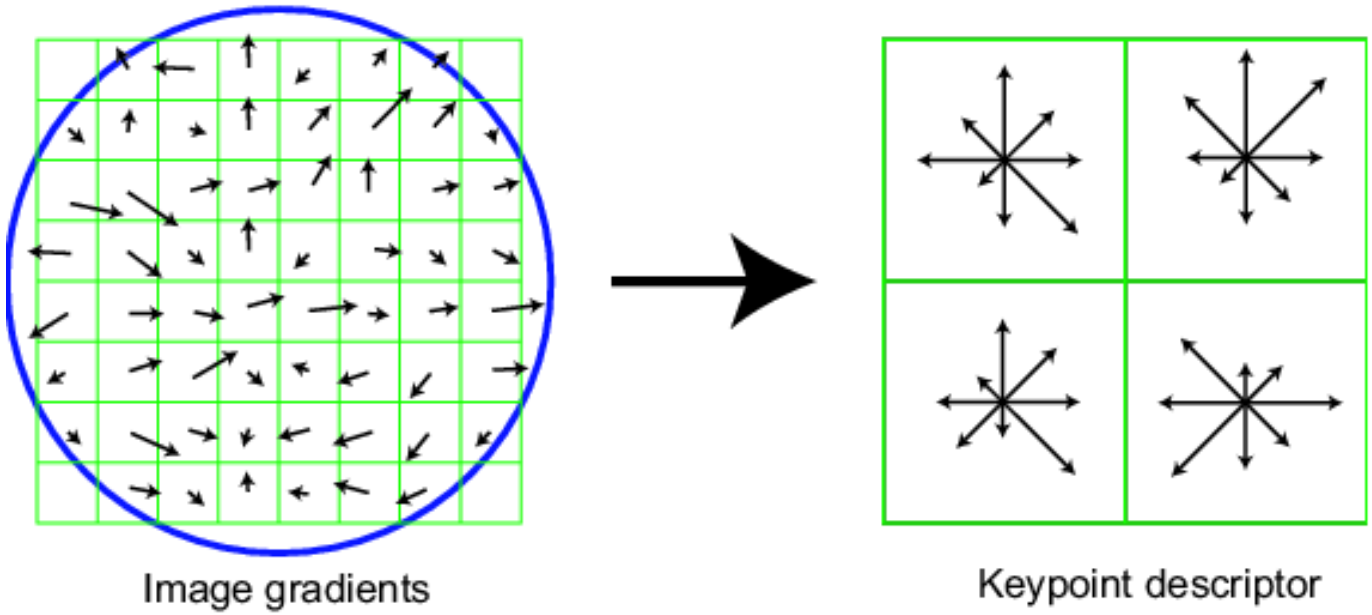


Adapted from slide by David Lowe

SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells
  (8x8 window and 2x2 grid shown below for simplicity)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

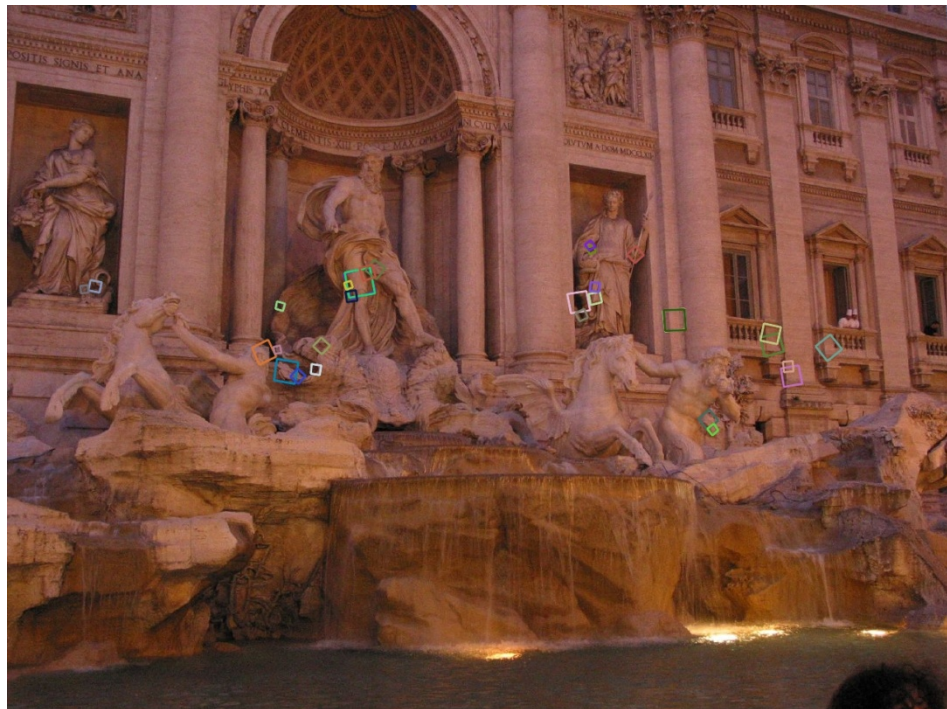


Adapted from slide by David Lowe

Properties of SIFT-based matching

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



Feature matching

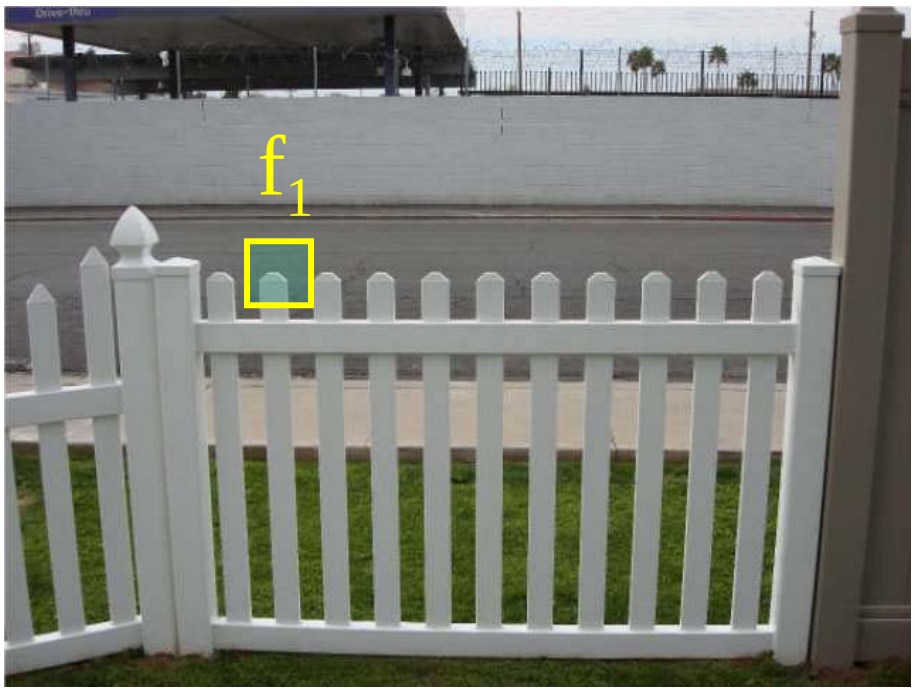
Given a feature in I_1, how to find the best match in I_2?

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

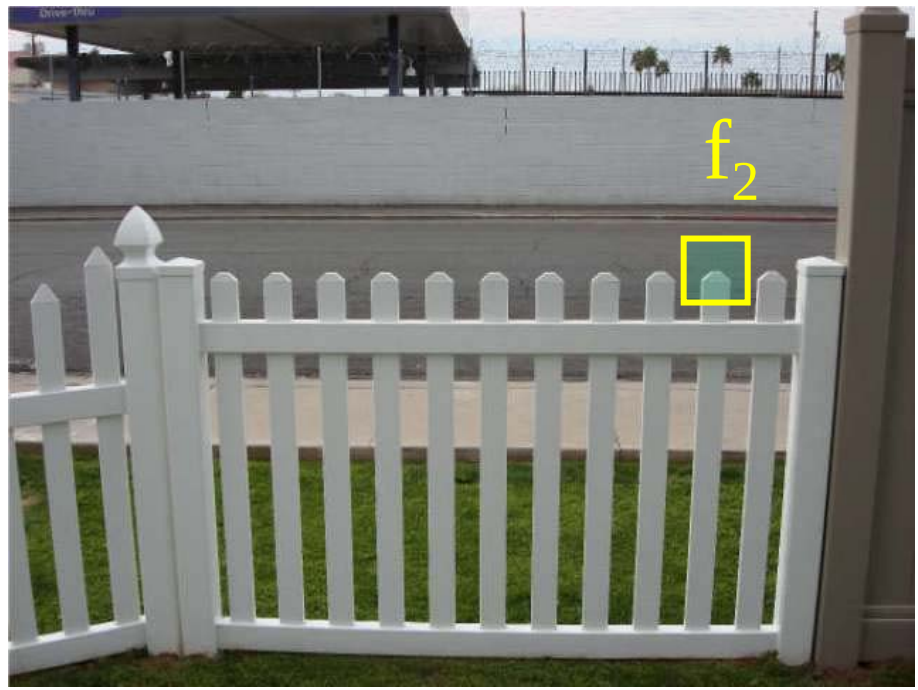
Feature distance: SSD

How to define the similarity 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
  - Doesn't provide a way to discard ambiguous (bad) matches



I_1

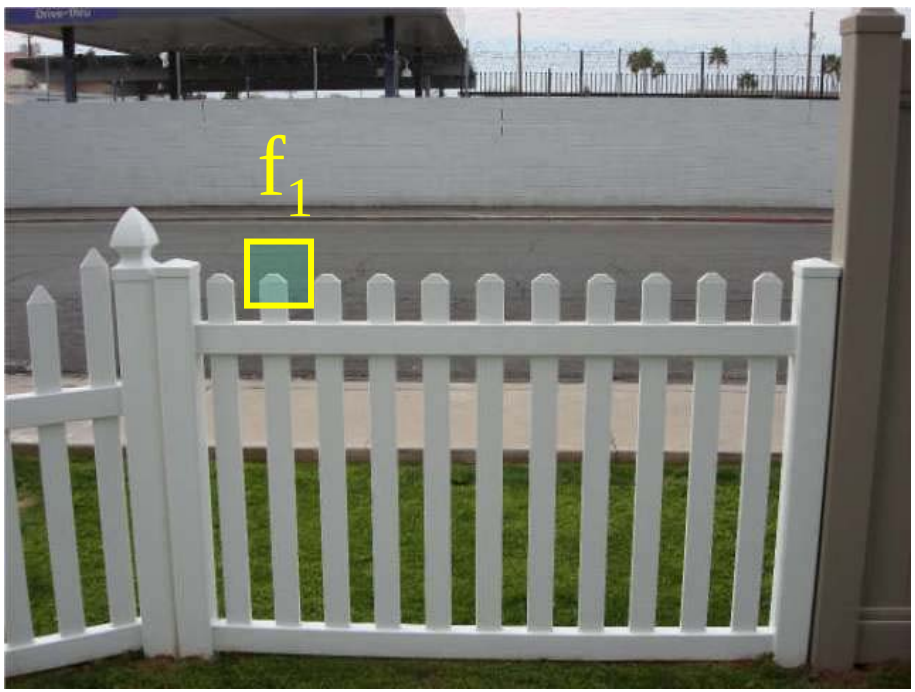


I_2

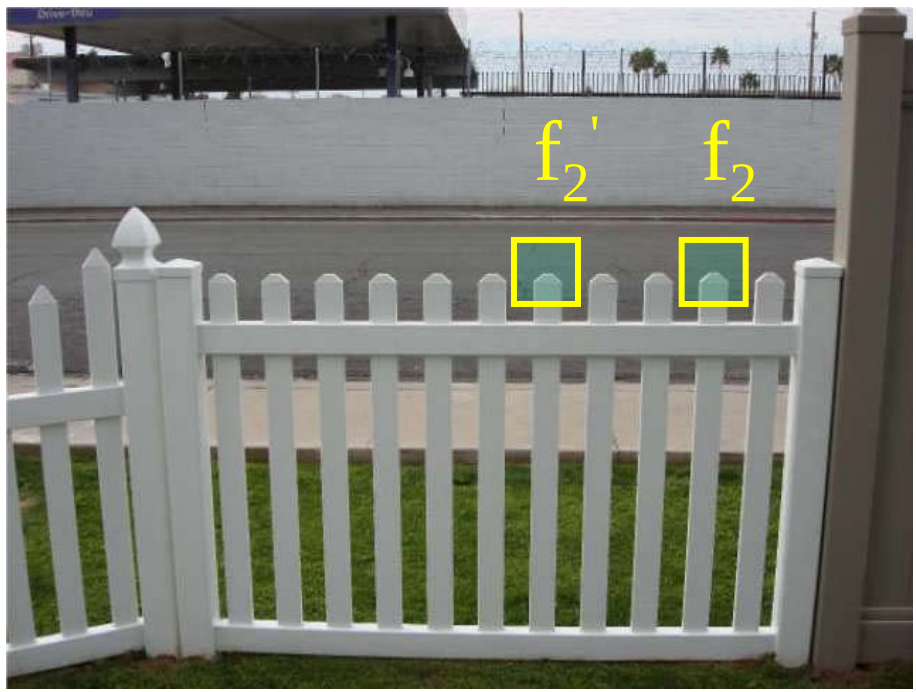
Feature distance: Ratio of SSDs

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
  - An ambiguous/bad match will have ratio close to 1
  - Look for unique matches which have low ratio

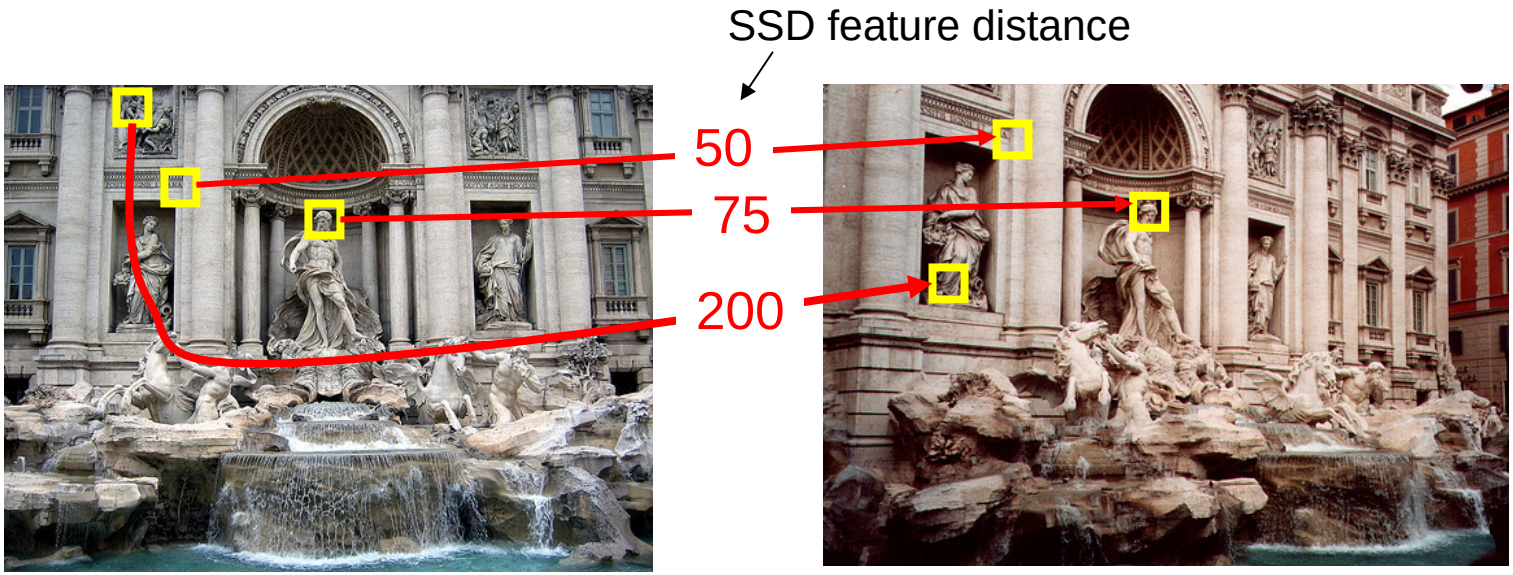


I_1



I_2

Image matching

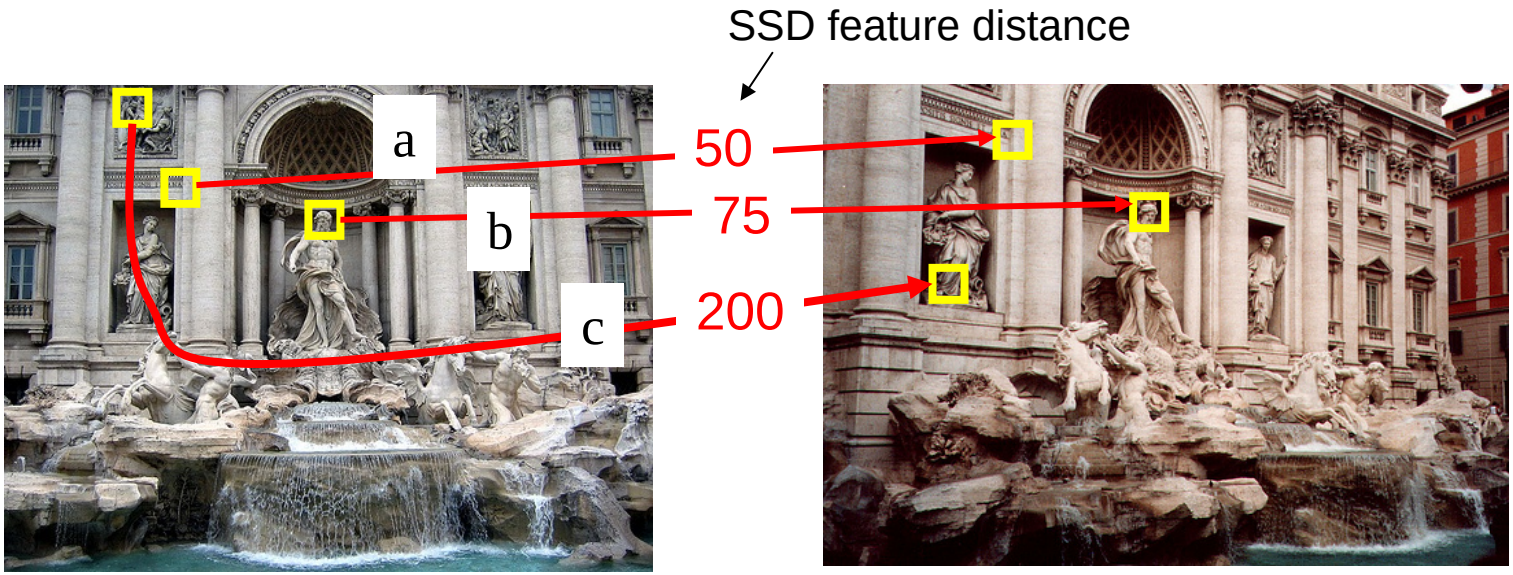


Suppose we use SSD
Small values are possible matches but how small?

Decision rule: Accept match if SSD < T
where T is a threshold

What is the effect of choosing a particular T?

Effect of threshold T



Decision rule: Accept match if SSD < T

Example: **Large T**

T = 250 ⇒ a, b, c are all accepted as matches

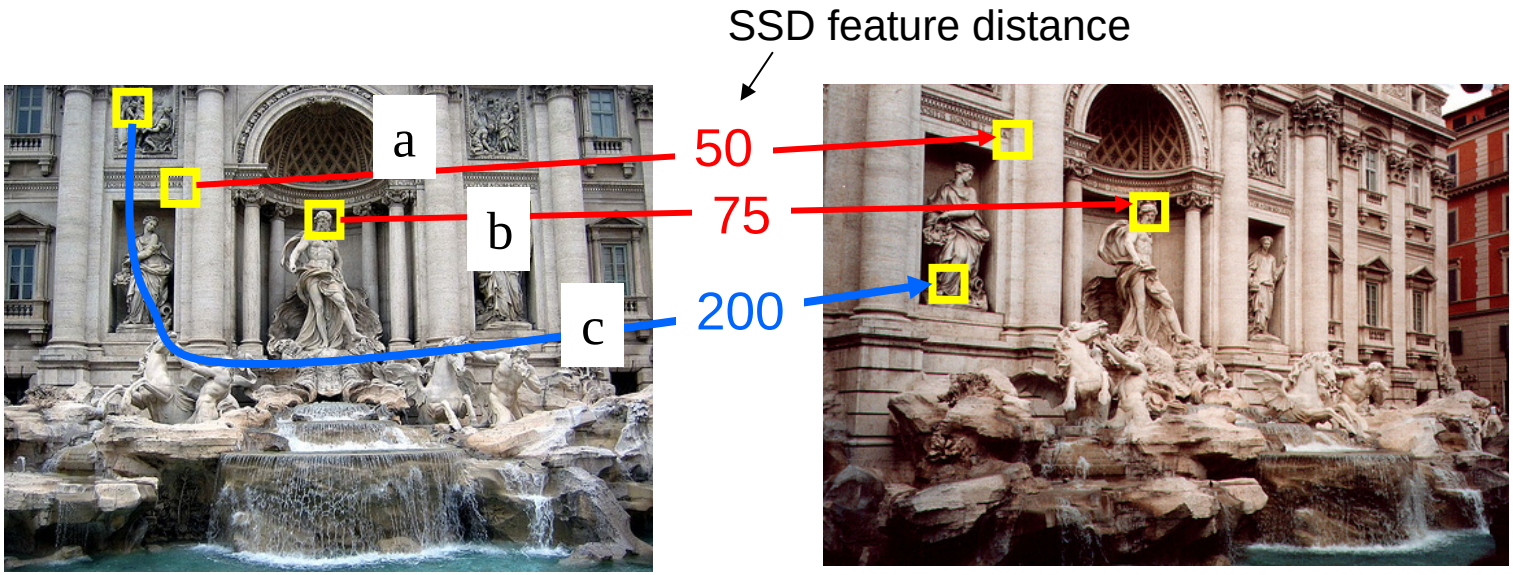
a and b are true matches (“**true positives**”)

– they are actually matches

c is a false match (“**false positive**”)

– actually not a match

Effect of threshold T



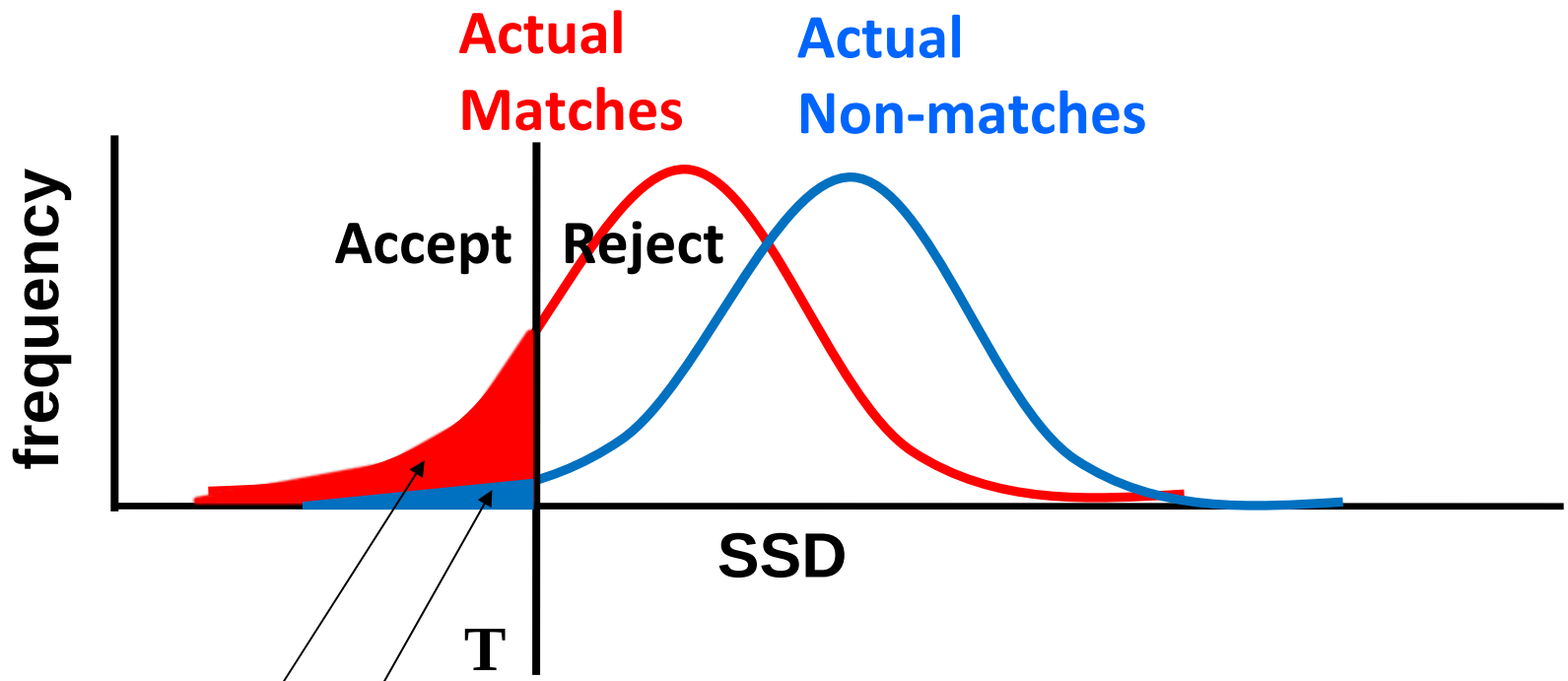
Decision rule: Accept match if SSD < T

Example: **Smaller T**

T = 100 ⇒ only a and b are accepted as matches

a and b are true matches (“true positives”)
c is no longer a “false positive” (it is a “true negative”)

True positives and false positives



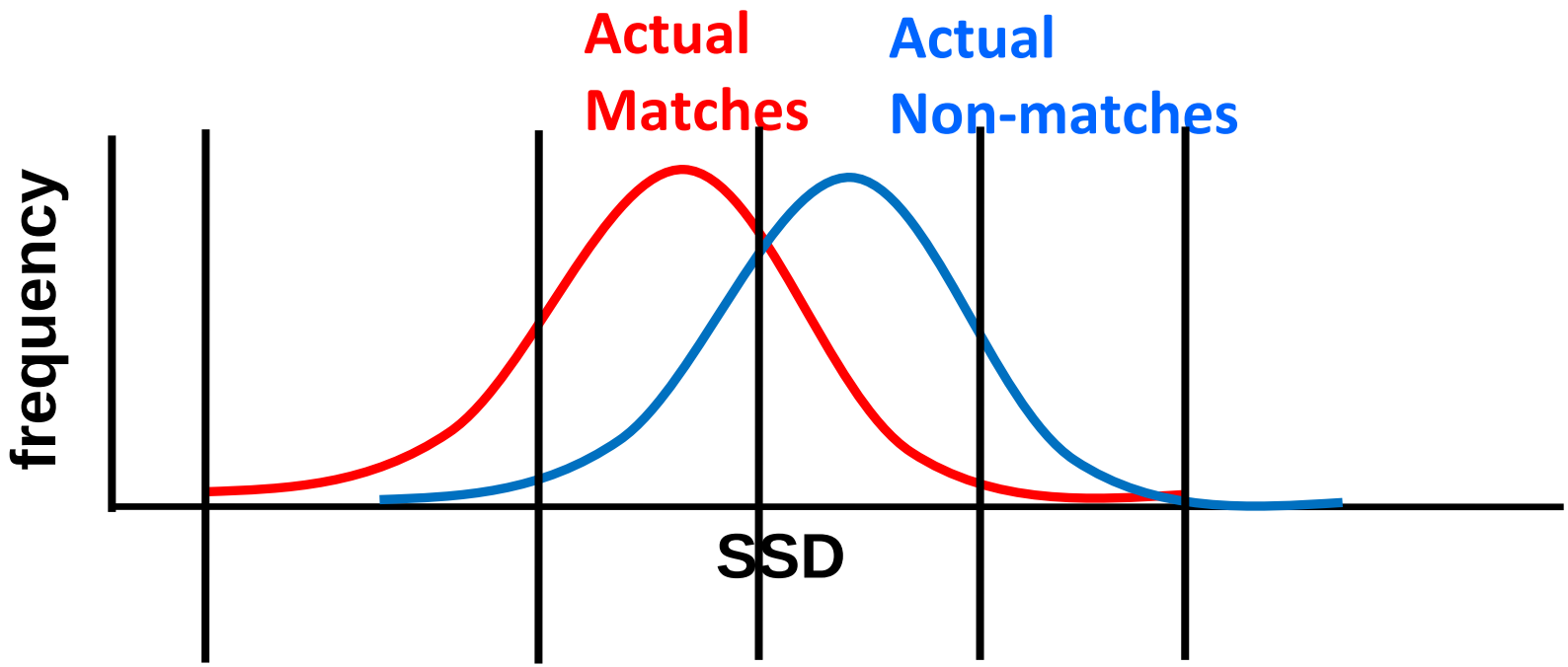
True positives

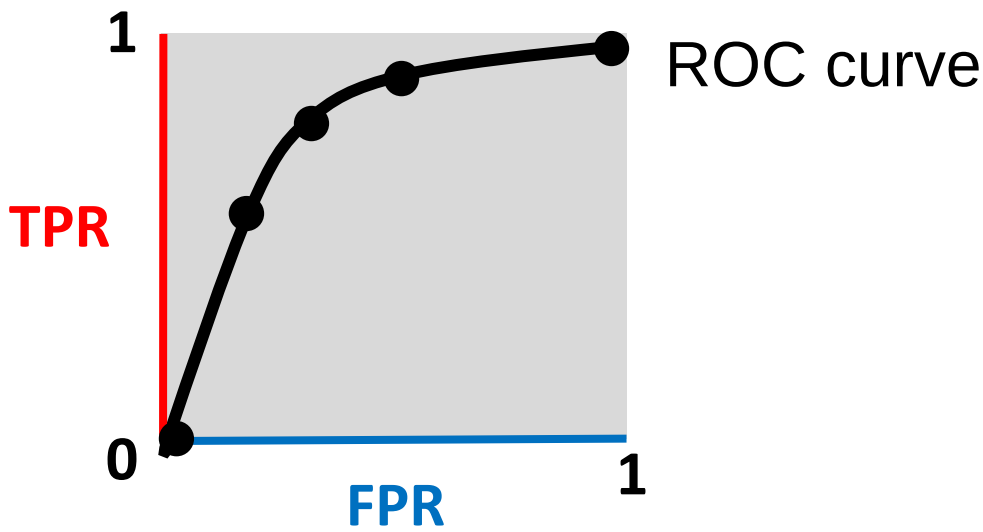
False positives

True positive rate = (TPR) $\frac{\#\text{ true positives}}{\#\text{ actual matches}}$

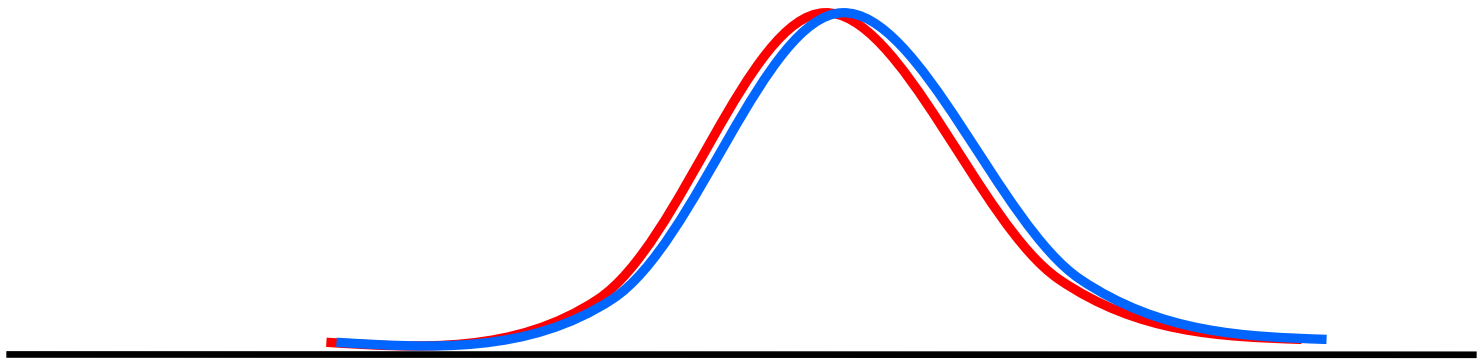
False positive rate = (FPR) $\frac{\#\text{ false positives}}{\#\text{ actual nonmatches}}$

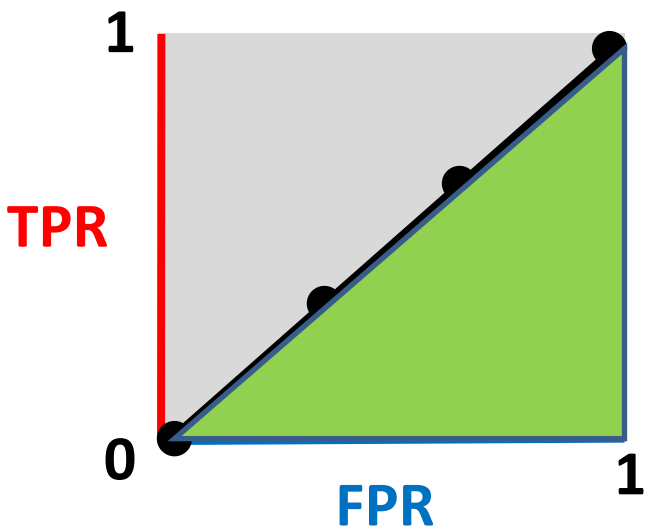
Receiver Operating Characteristic (ROC) curve





If the features selected were bad…

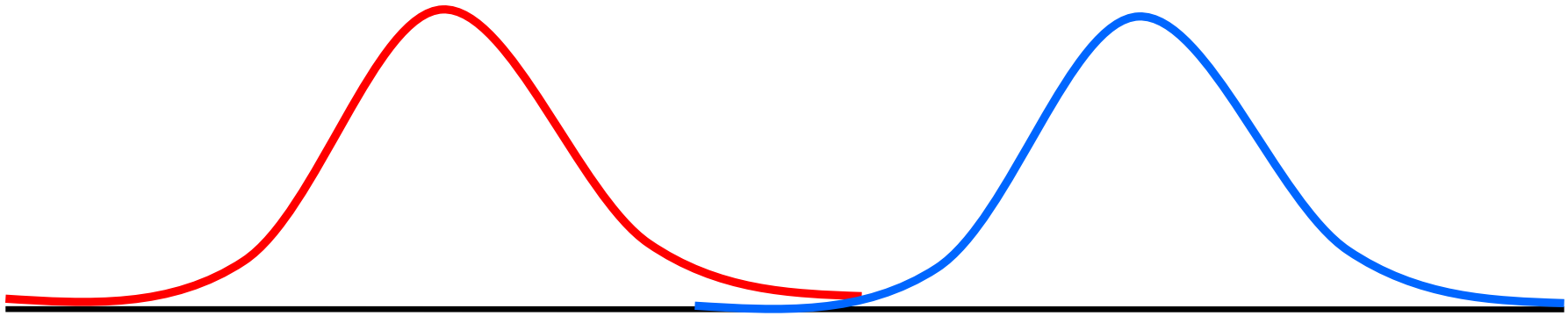


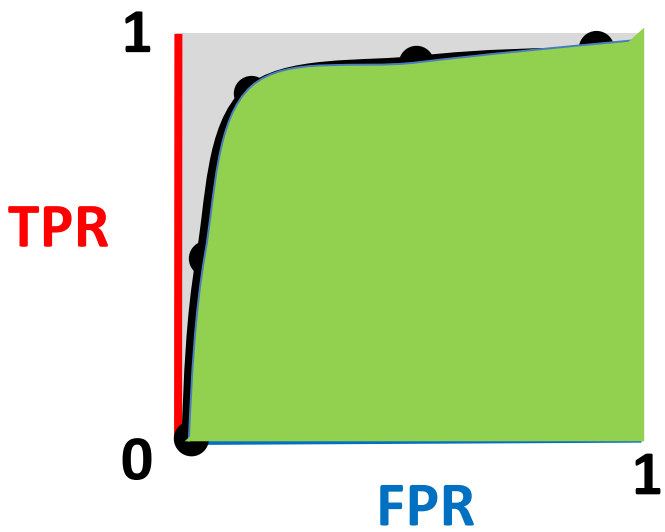


Area ≈ 0.5

Adapted from a slide by Shin Kira

If features selected were good…



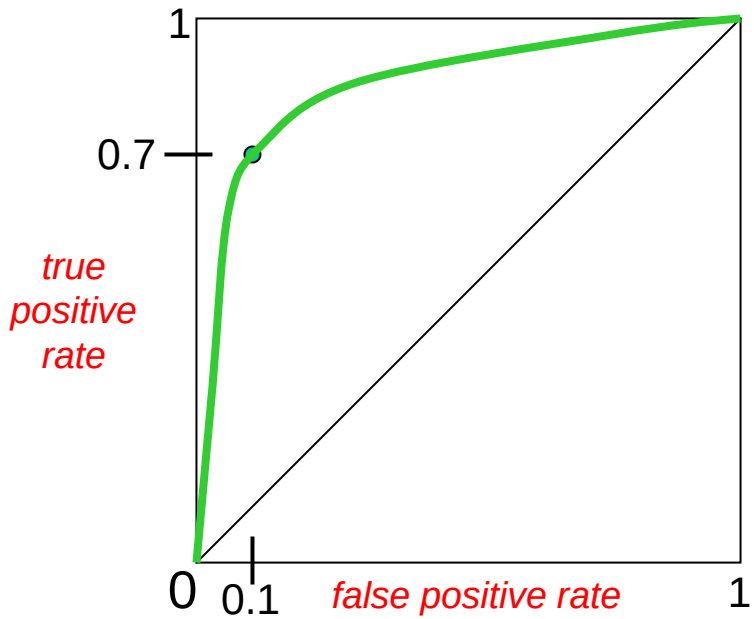


Area ≈ 1.0

Adapted from a slide by Shin Kira

Using ROC curves

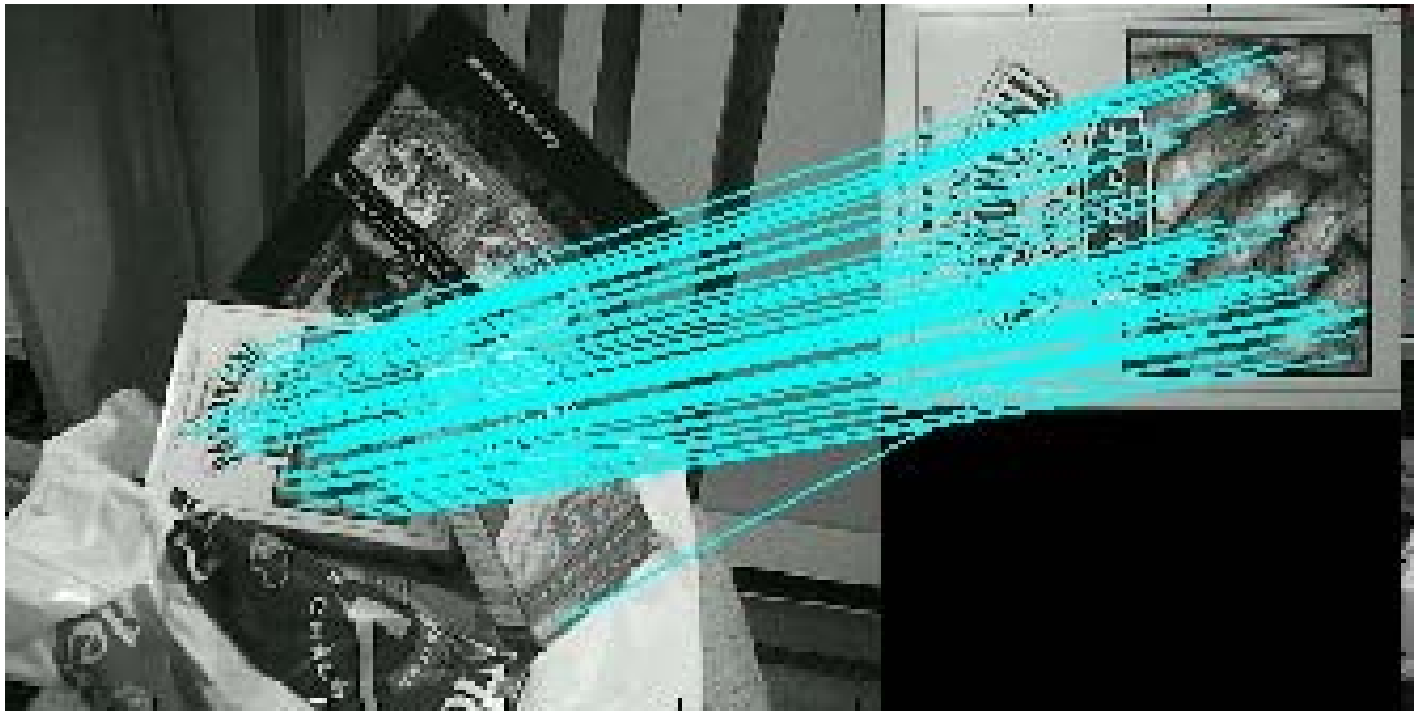
- Useful for comparing different feature matching methods
- Pick method that maximizes area under the curve
- More info: http://en.wikipedia.org/wiki/Receiver_operating_characteristic



Applications of Features

- Image alignment (e.g., mosaics): Project #2
- Object recognition
- 3D reconstruction
- Motion tracking
- Indexing and database retrieval
- Robot navigation
- …

Object recognition using SIFT



http://www.cs.ubc.ca/~lowe/keypoints/

Object recognition using SIFT

(From Lowe, 2004)

Sony Aibo

SIFT usage:

- Recognize charging station
- Communicate with visual cards
- Teach object recognition

Next Time: Panoramic mosaics and image stitching

Things to do:

- Work on Project 1
- Read online readings on mosaics