

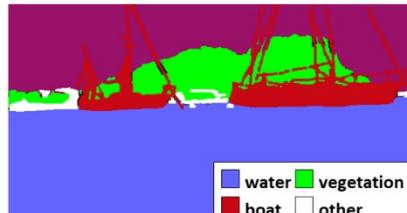
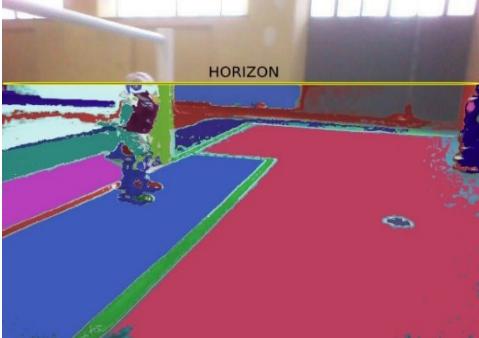
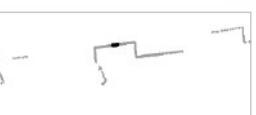
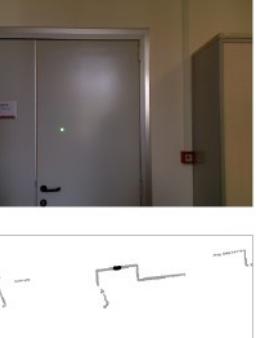


UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Corso di Sistemi Informativi
A.A. 2018/19

Features

Aprile 2019



■ water ■ vegetation
■ boat ■ other

Riferimenti

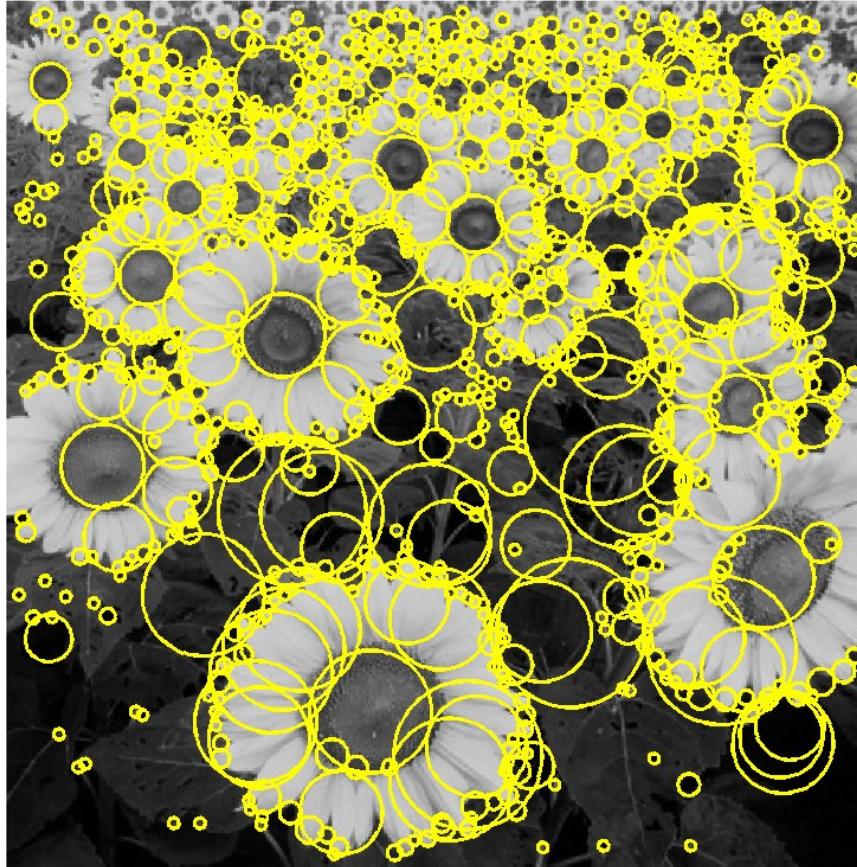
- Queste slide sono adattate da
Noah Snavely - CS5670: Computer Vision
["Lecture 4: Intro to local features and Harris corner detection"](#)
- I contenuti fanno riferimento al capitolo 4 del libro
"Computer Vision: Algorithms and Applications"
di Richard Szeliski, disponibile al seguente indirizzo
<http://szeliski.org/Book/>

Feature extraction

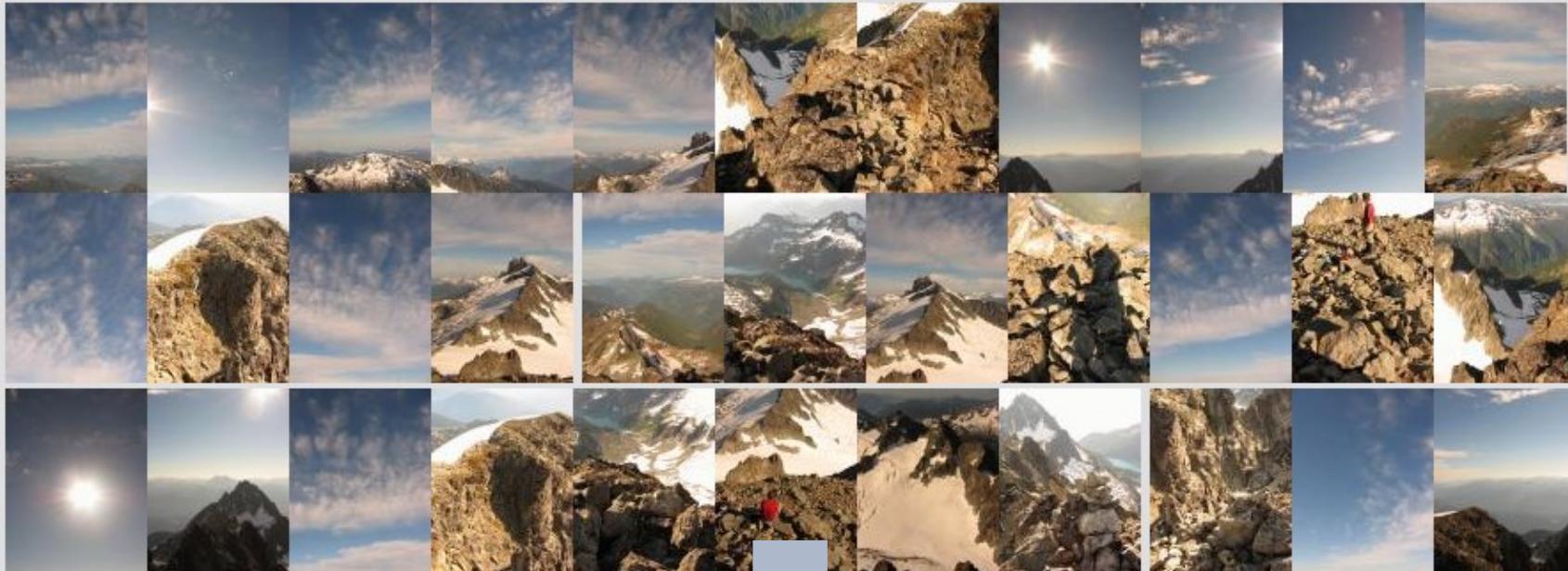
corners



blobs



Automatic panoramas



Credit: Matt Brown

Automatic panoramas



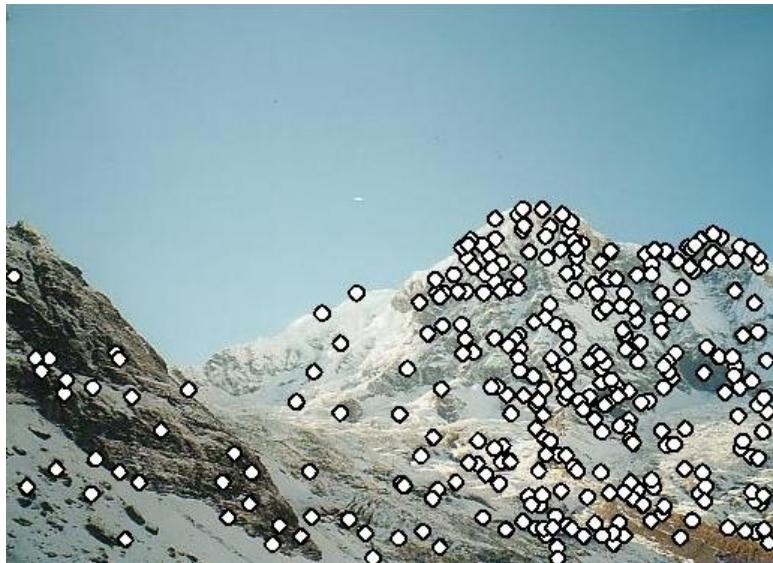
Feature matching

We have two images, how do we combine them?



Feature matching

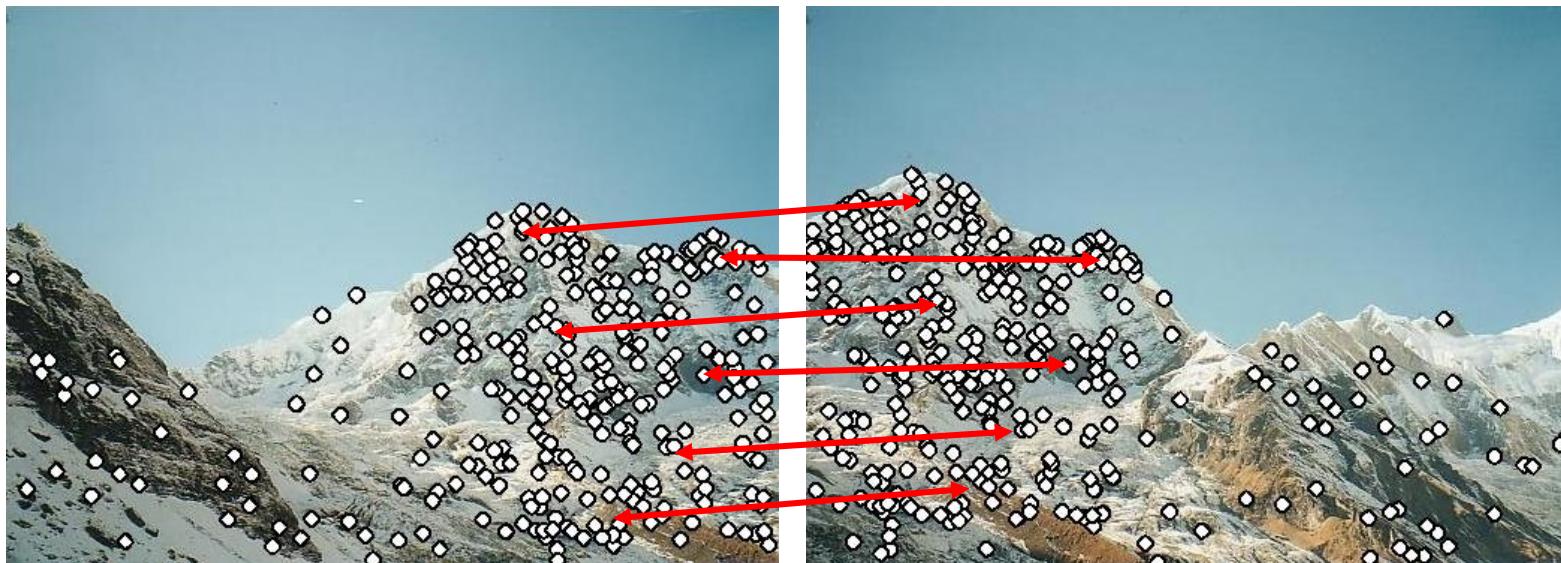
Step 1: extract features



Feature matching

Step 1: extract features

Step 2: match features



Feature matching

Step 1: extract features

Step 2: match features

Step 3: align images



Visual SLAM



<https://www.youtube.com/watch?v=DPLh6MoxPAk>

Image matching



by [Diva Sian](#)



by [swashford](#)

Harder case

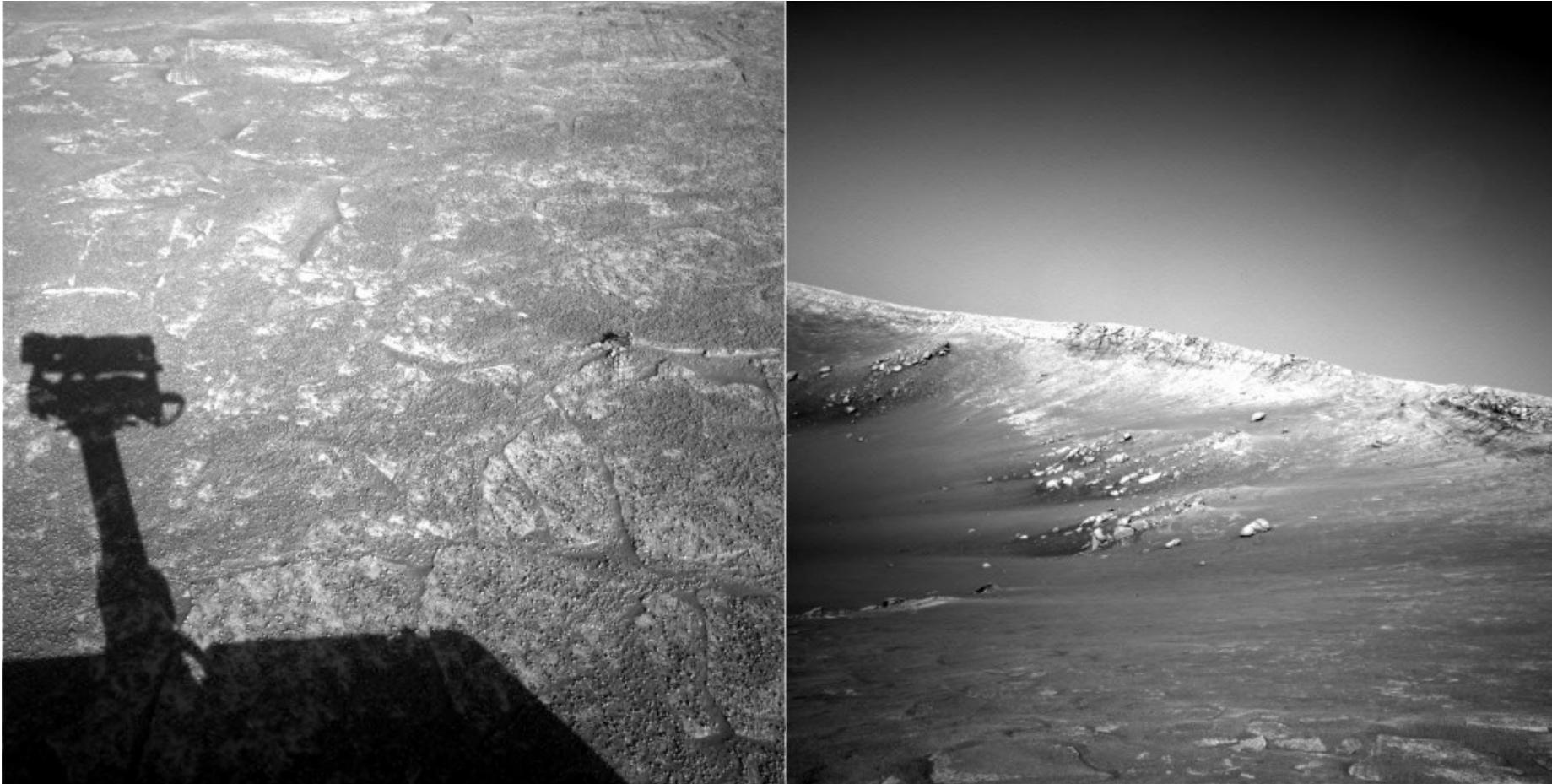


by [Diva Sian](#)

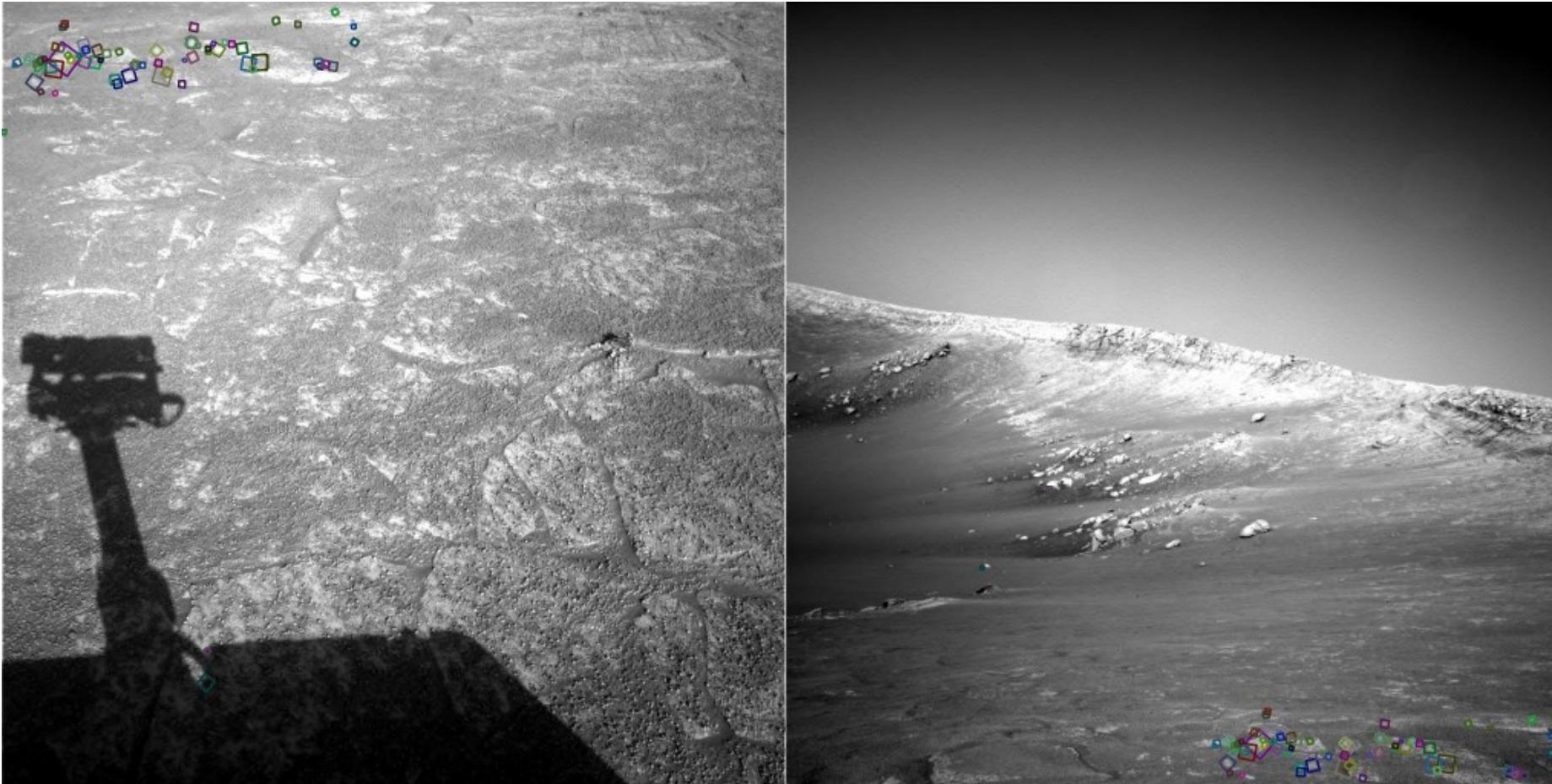


by [scgbt](#)

Harder still?

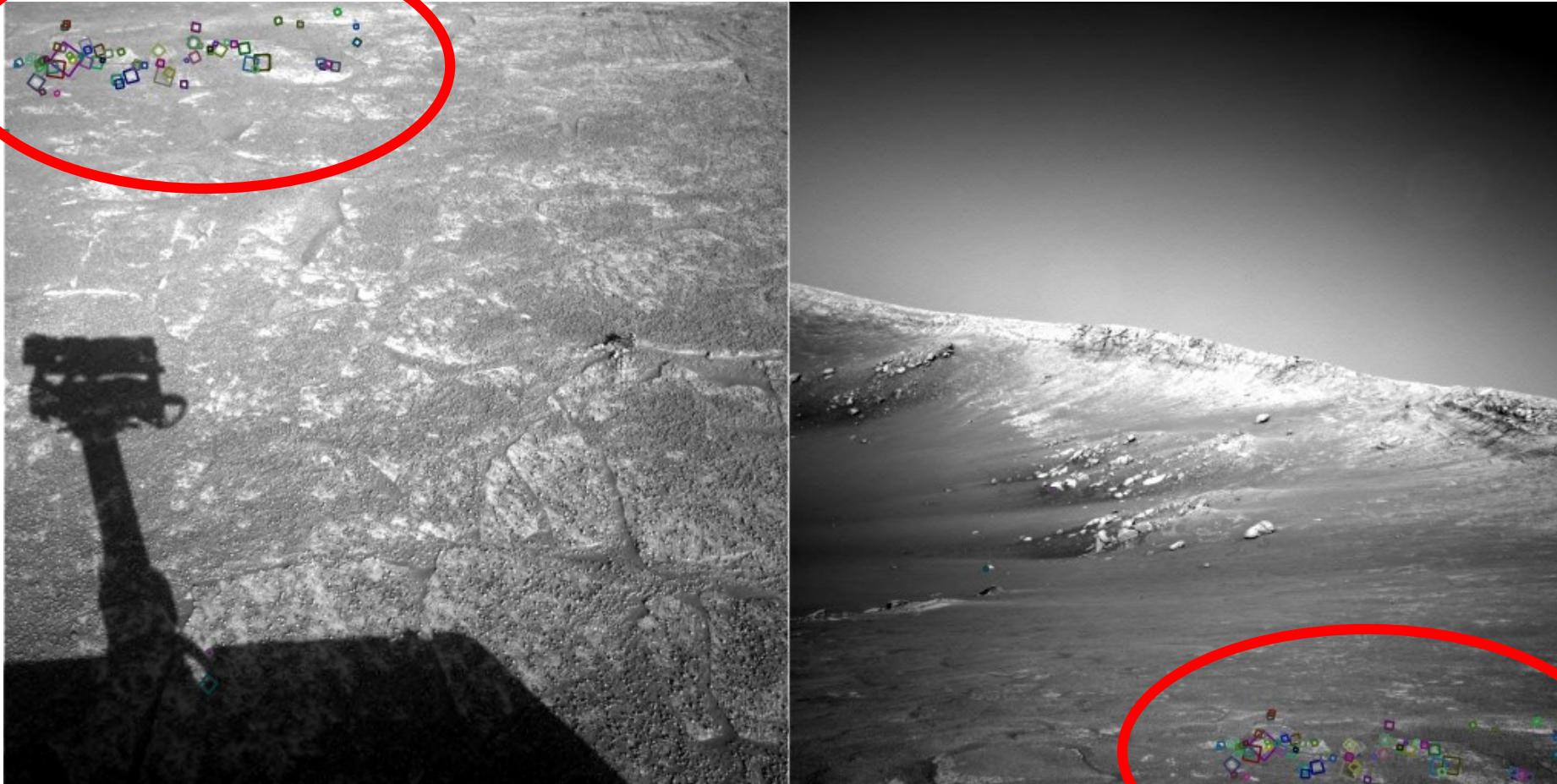


Harder still?



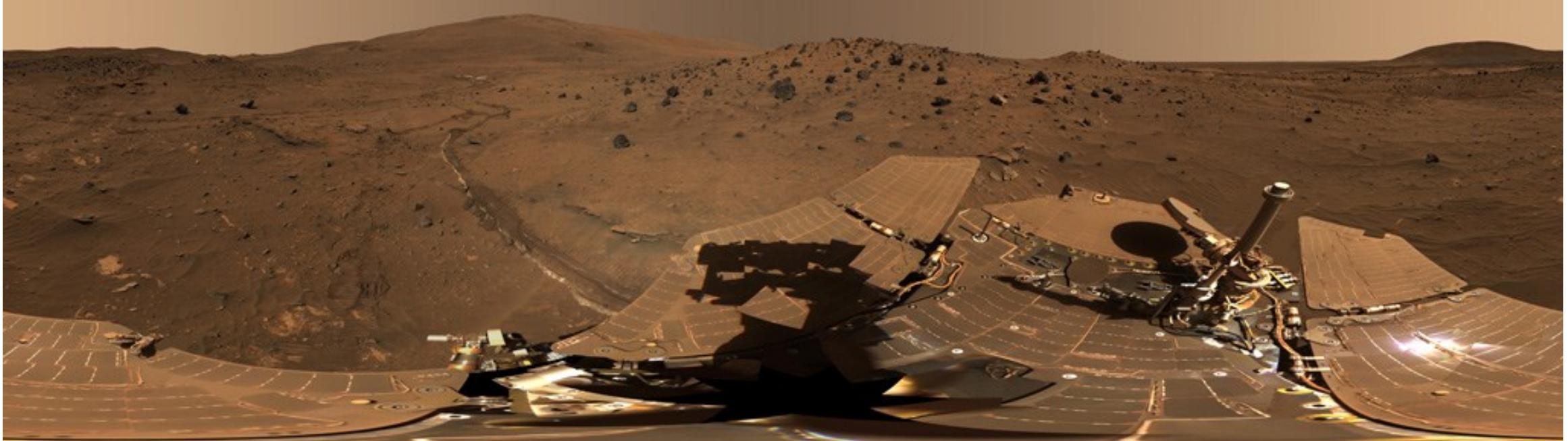
NASA Mars Rover images
with SIFT feature matches

Harder still?



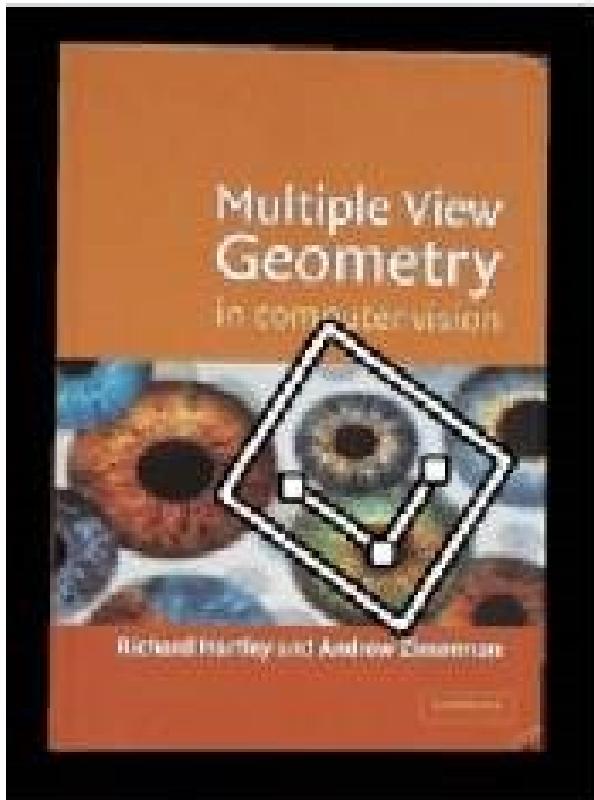
NASA Mars Rover images
with SIFT feature matches

Spirit Mars Rover Panorama

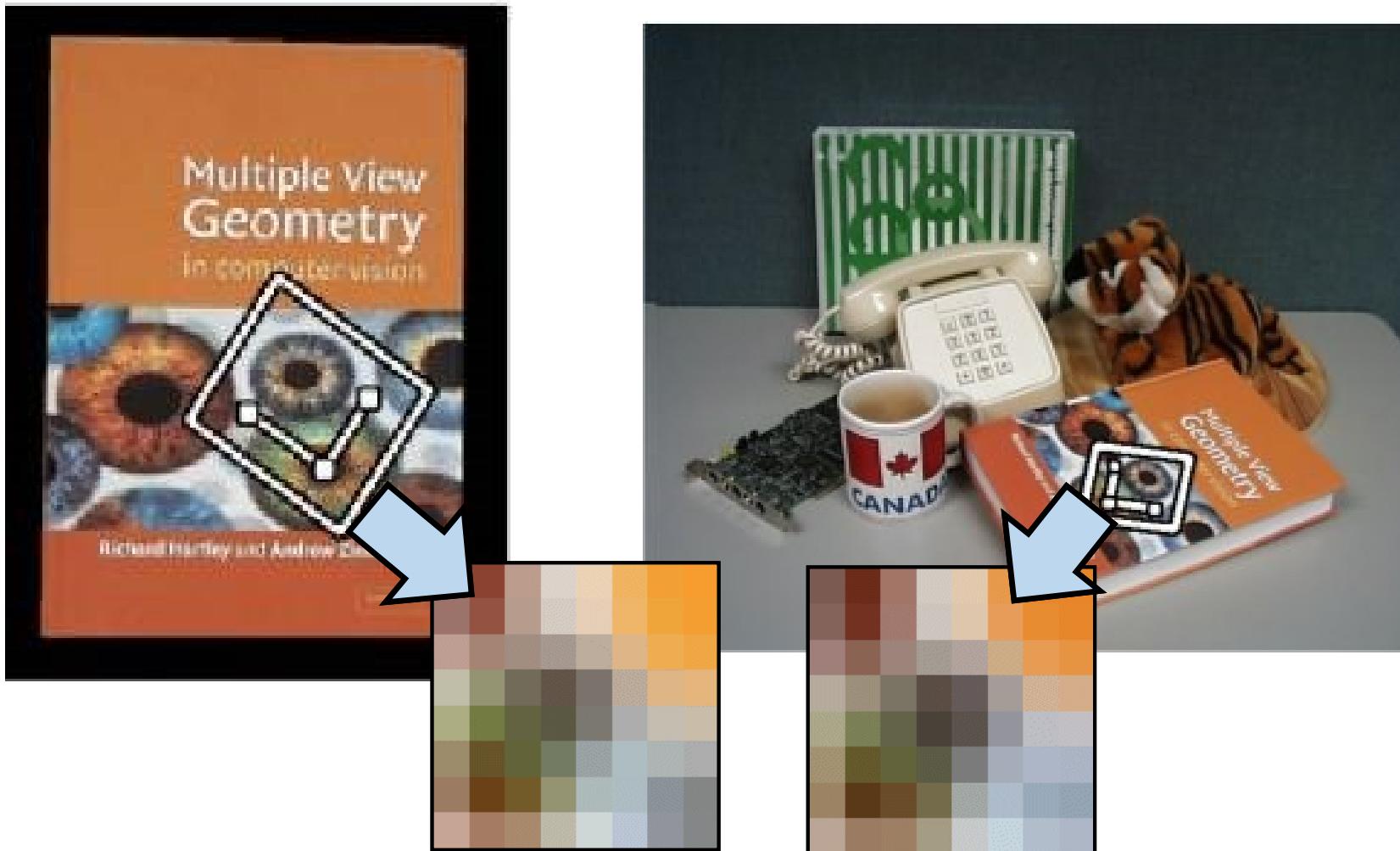


<https://mars.nasa.gov/mer/multimedia/panoramas/spirit/>

Feature Matching



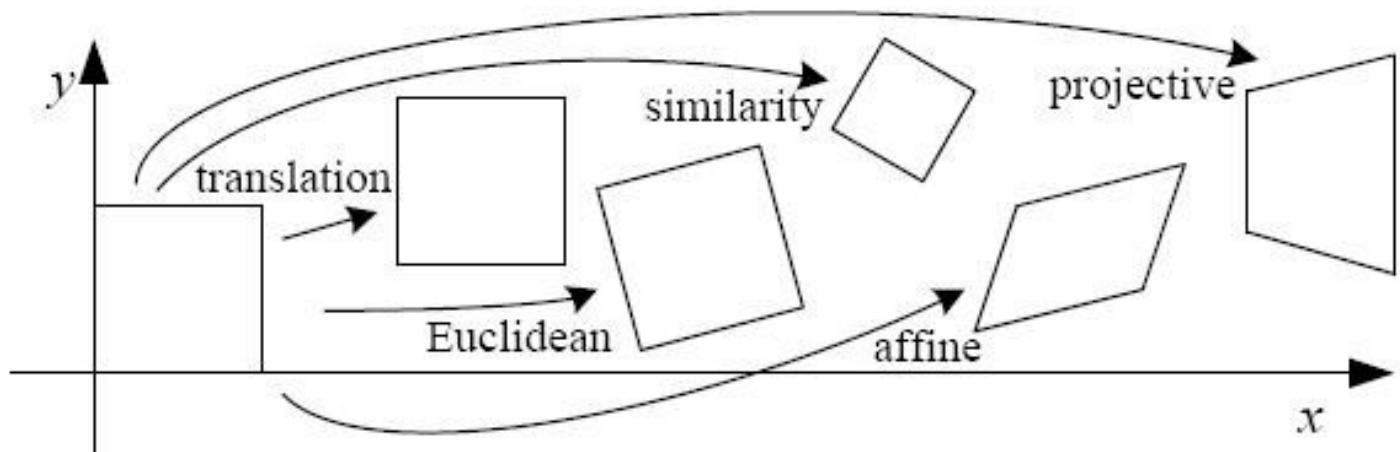
Feature Matching



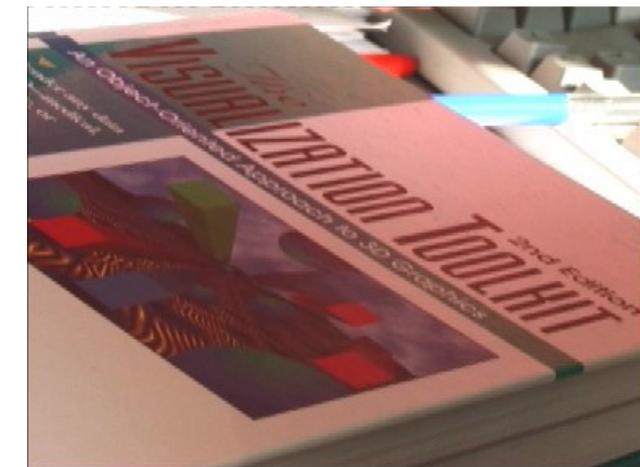
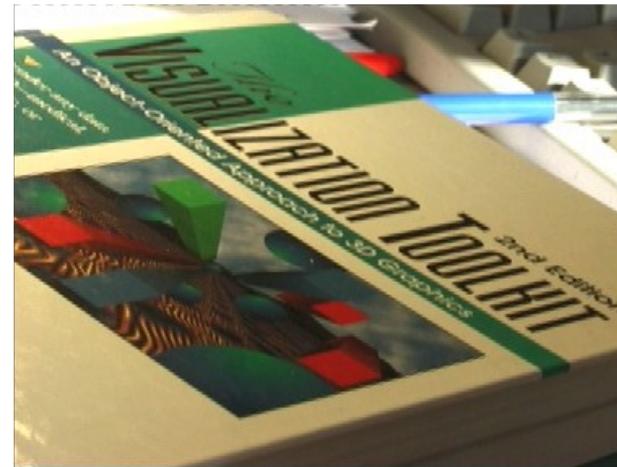
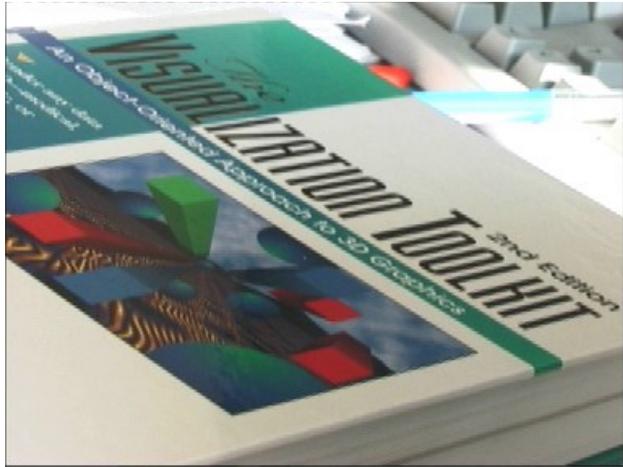
Geometric transformations

- Translation
- Euclidean (translation + rotation)
- Similarity (translation + rotation + scale)
- Affine transformations
- Projective transformations

Only holds
for planar patches



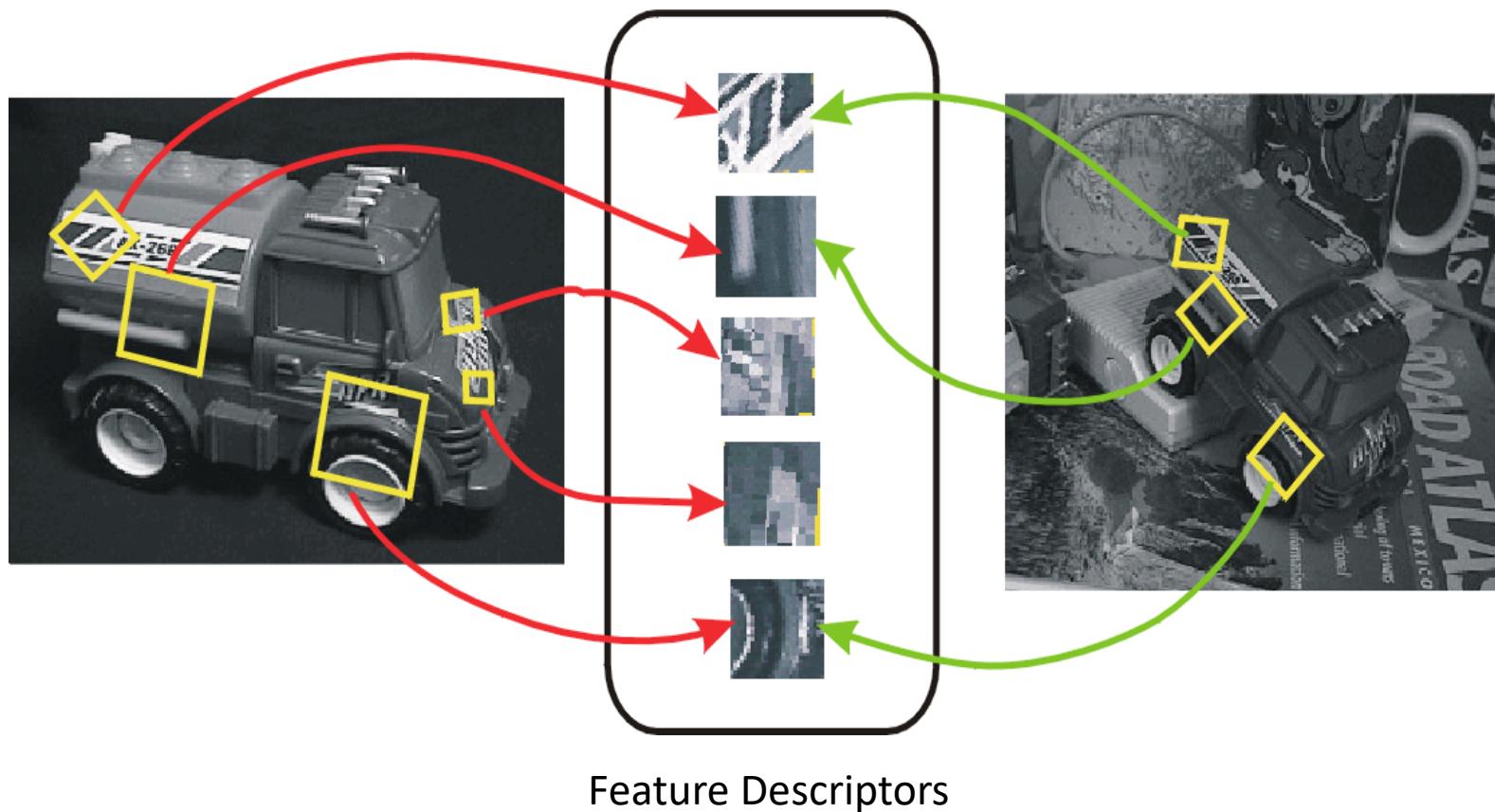
Photometric transformations



Invariant Local Features

Find features that are invariant to transformations

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



Advantages of local features

- **Locality**
 - features are local, so robust to occlusion and clutter
- **Quantity**
 - hundreds or thousands in a single image
- **Distinctiveness**
 - can differentiate a large database of objects
- **Efficiency**
 - real-time performance achievable

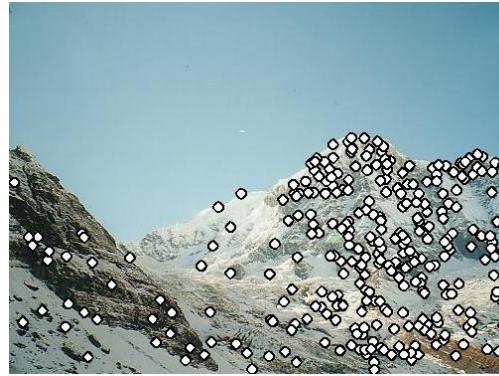
Approach

- 1. Feature detection:** find it
- 2. Feature descriptor:** represent it
- 3. Feature matching:** match it

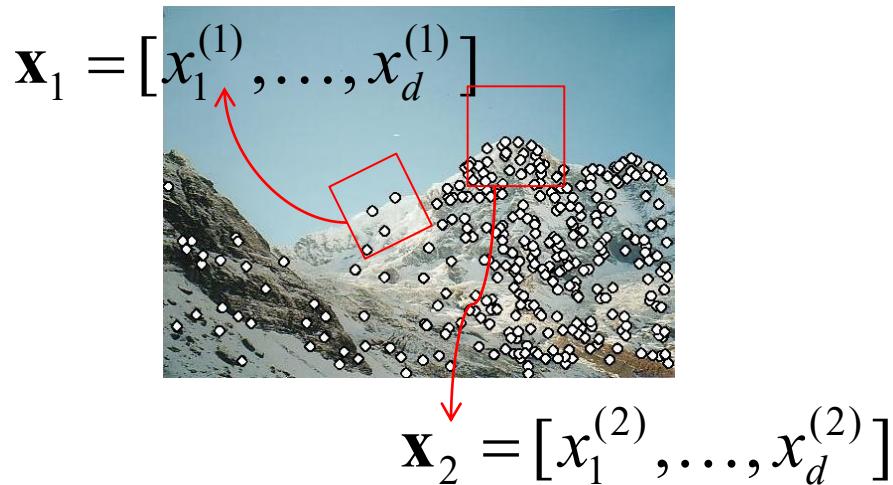
Feature tracking: track it, when motion

Local features: main components

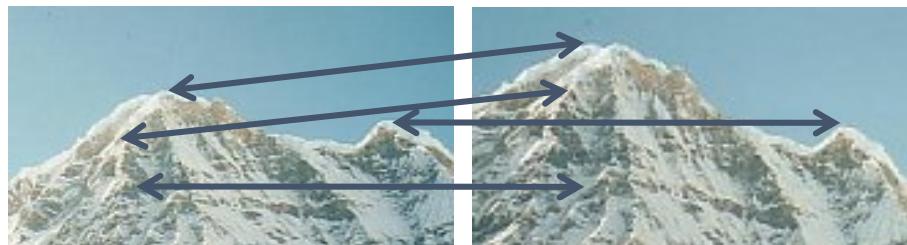
- 1) Detection: Identify the interest points



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



- 3) Matching: Determine correspondence between descriptors in two views



What points to choose?



Source: Derek Hoiem

Low-texture region?



Gradients have small magnitude

Edge?



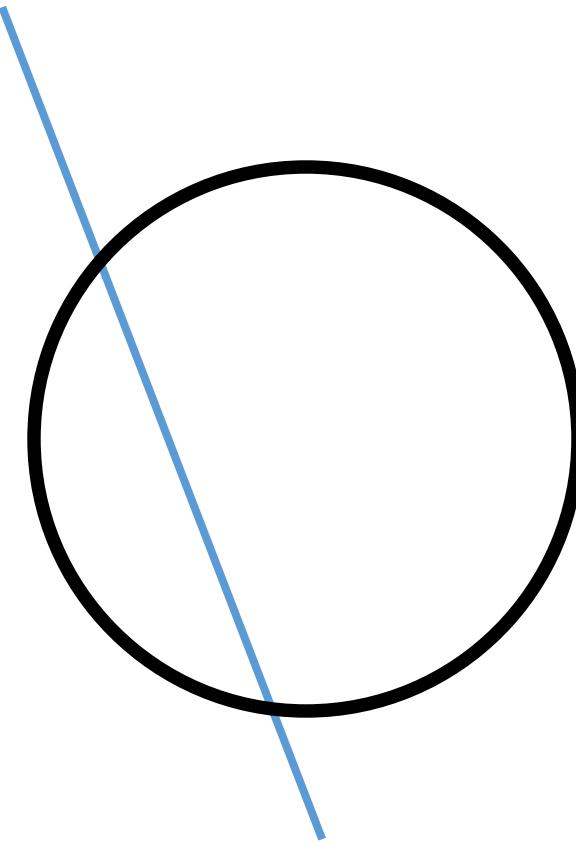
Gradients very large or very small

High-texture region?

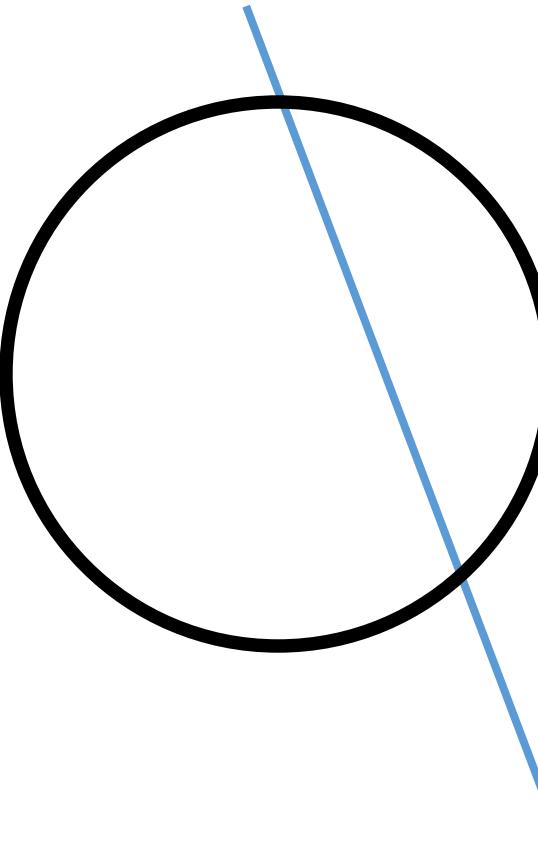


Gradients are different, large magnitudes

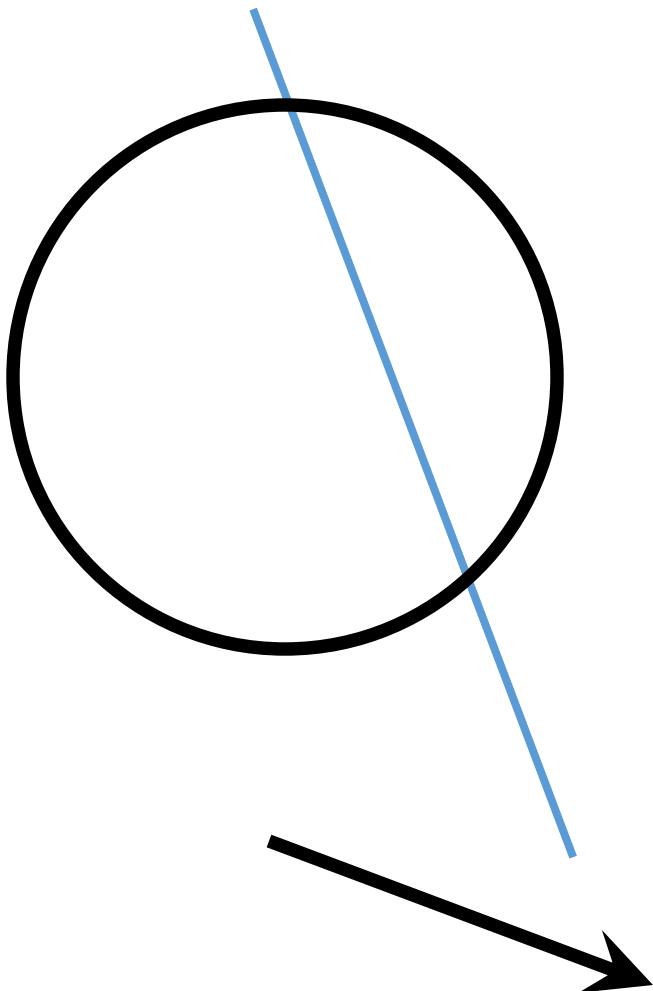
The aperture problem



The aperture problem



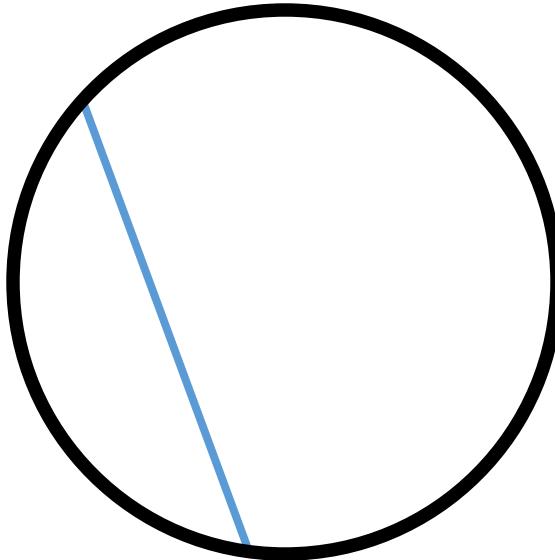
The aperture problem



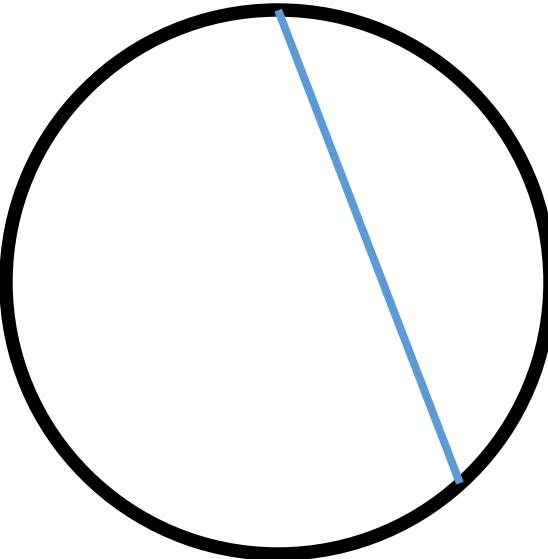
Actual motion

Source: Derek Hoiem

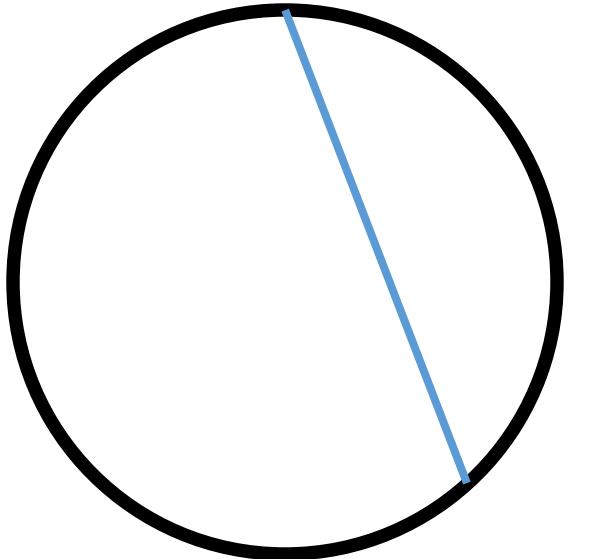
The aperture problem



The aperture problem

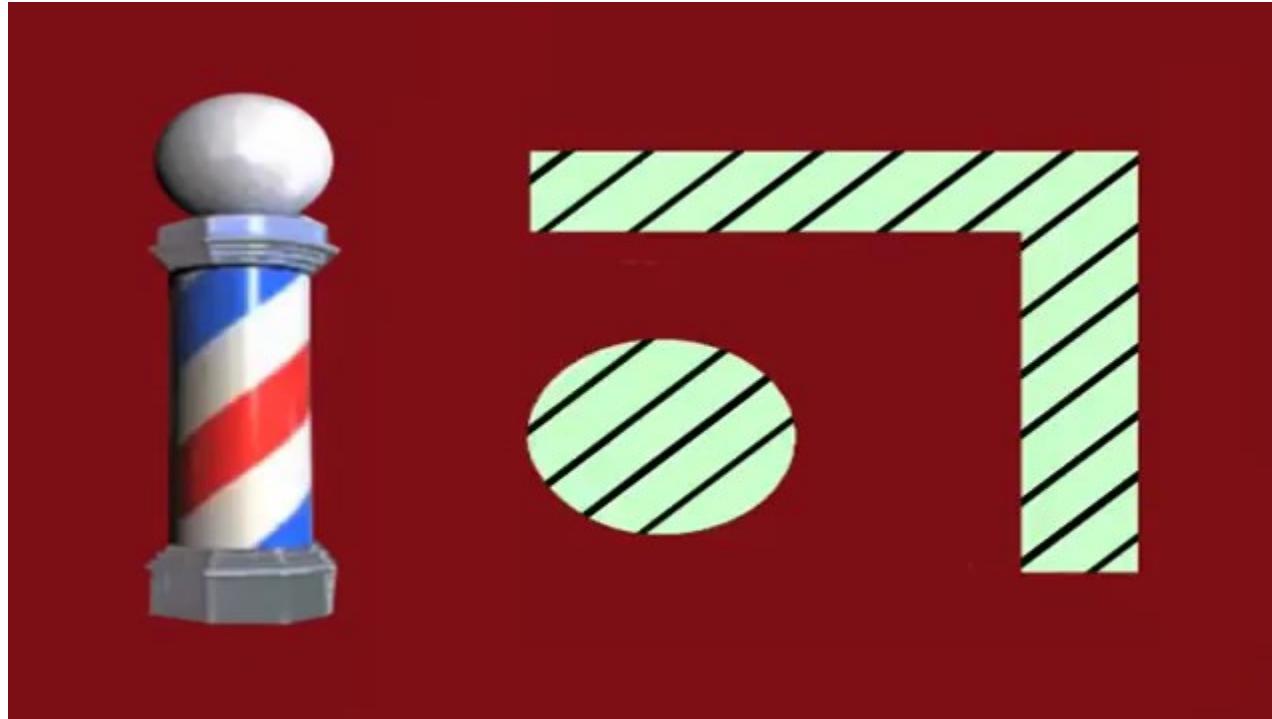


The aperture problem



Perceived motion

The barber pole illusion



http://www.opticalillusion.net/wp-content/uploads/2013/07/WedBarb4BB_F8_HQ.mp4?_=1

So, what makes a good feature?



Want uniqueness

Look for image regions that are unusual

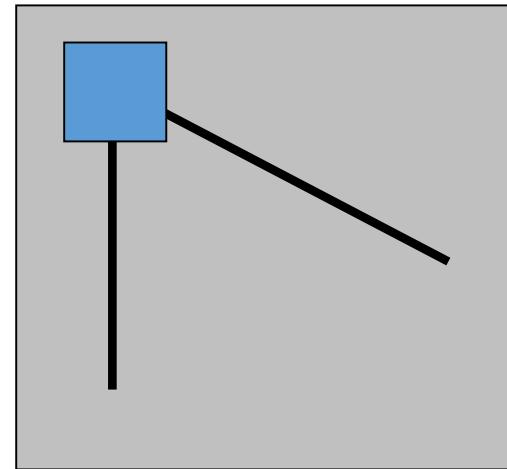
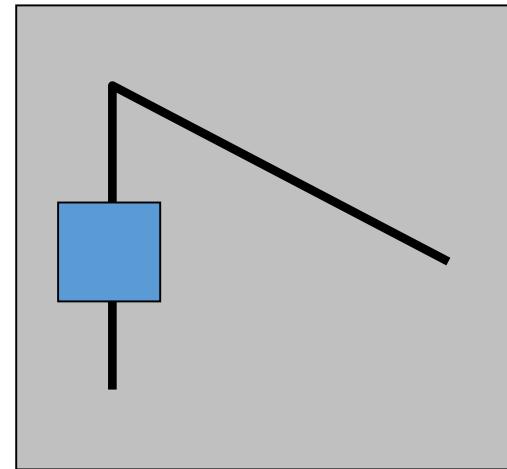
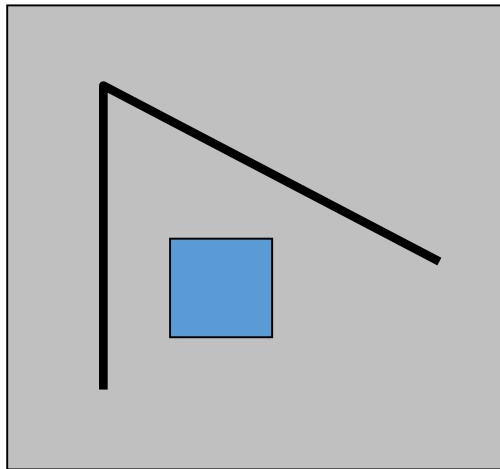
- Lead to unambiguous matches in other images

How to define “unusual”?

Want uniqueness

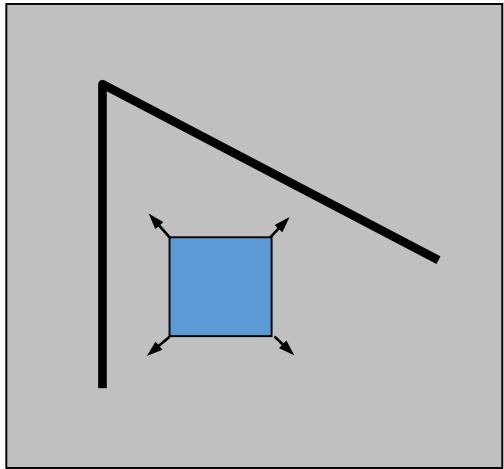
Suppose we only consider a small window of pixels

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

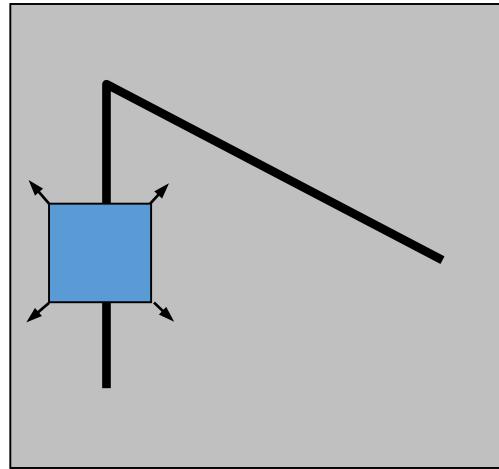


Want uniqueness

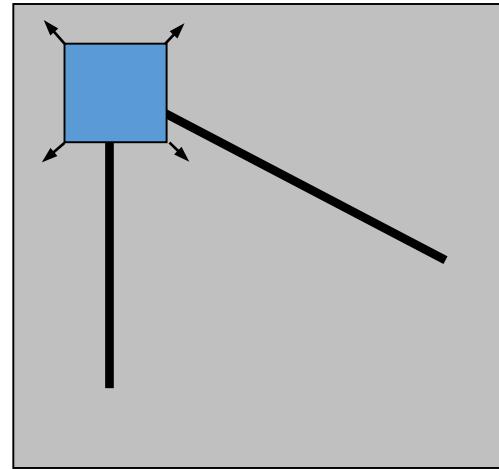
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



“flat” region:
no change in all
directions



“edge”:
no change along the
edge direction

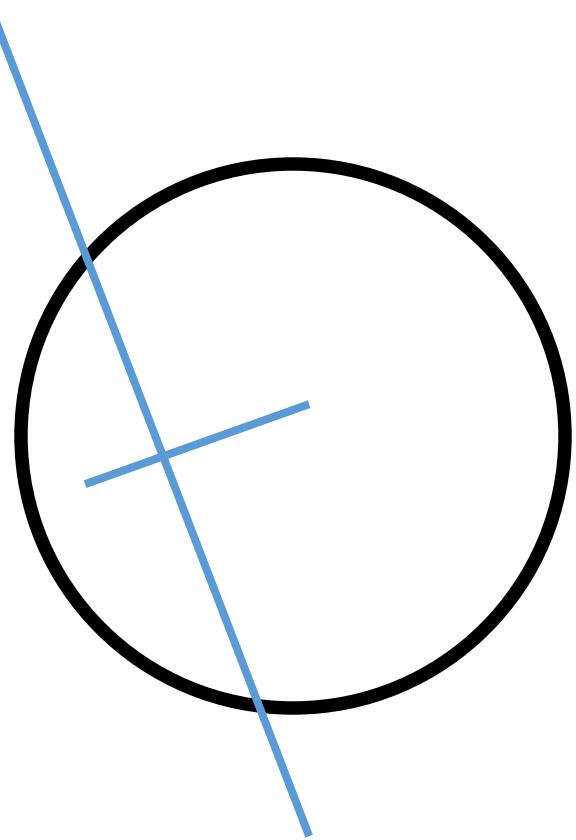


“corner”:
significant change in
all directions

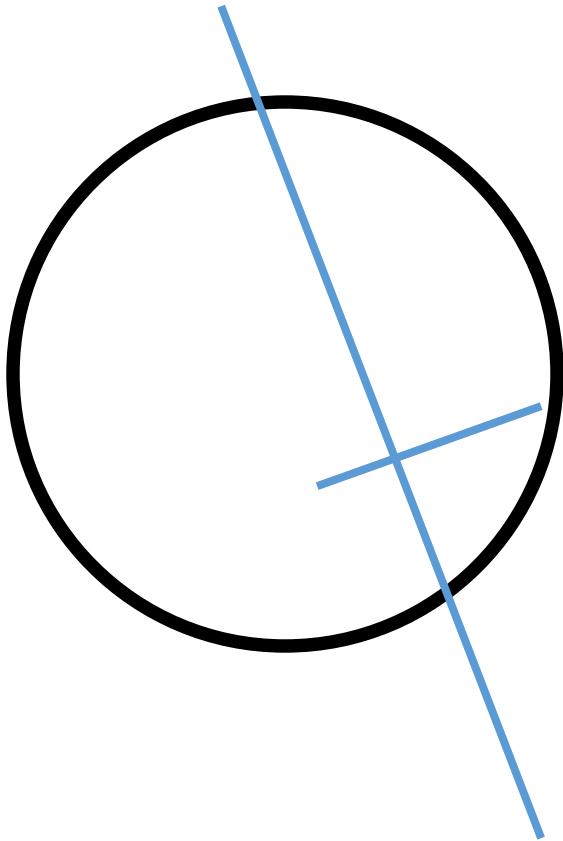
Corners

- Textureless patches are nearly impossible to localize
- Patches with large contrast changes (gradients) are easier to localize
- But straight line segments at a single orientation suffer from the aperture problem, i.e., it is only possible to align the patches along the direction normal to the edge direction
- Gradients in at least two (significantly) different orientations are the easiest, e.g., **corners**

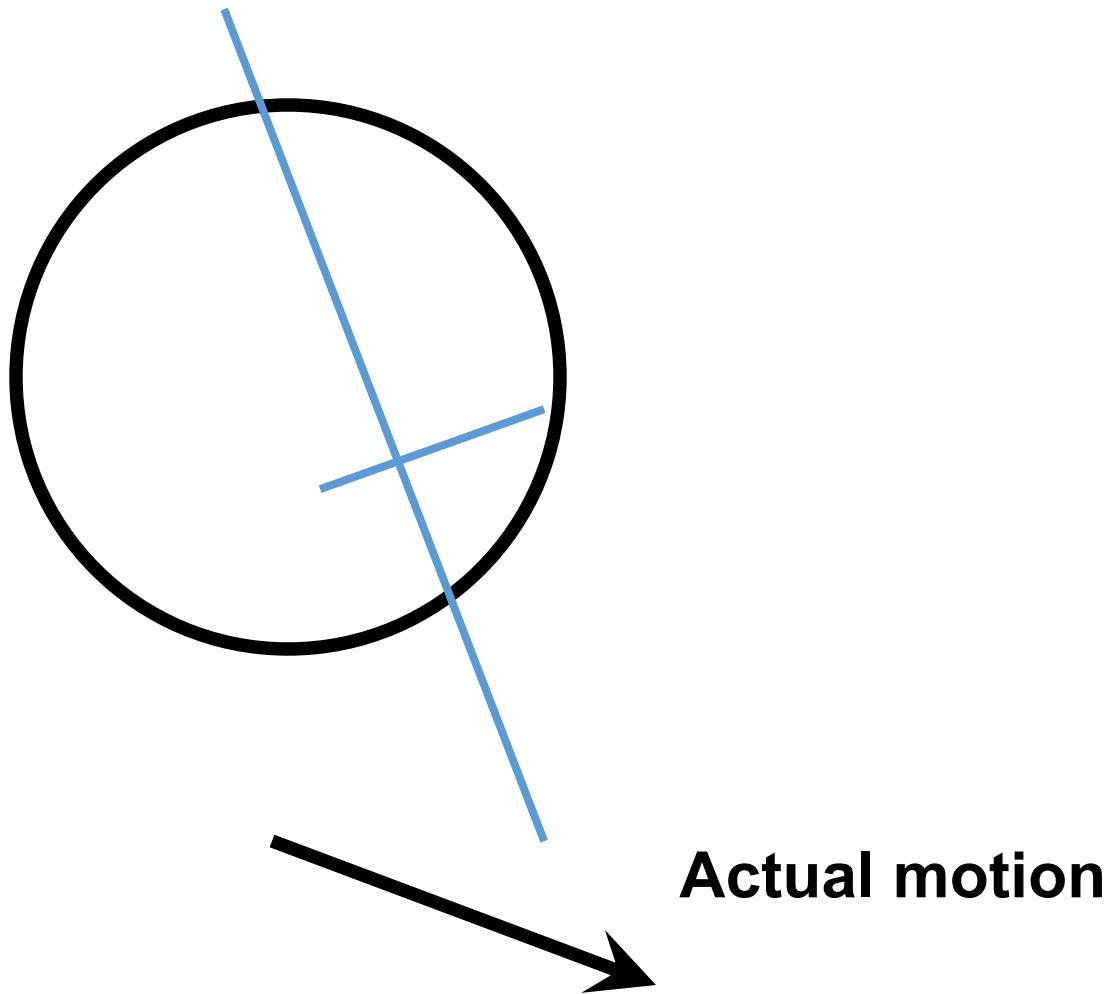
The aperture problem resolved



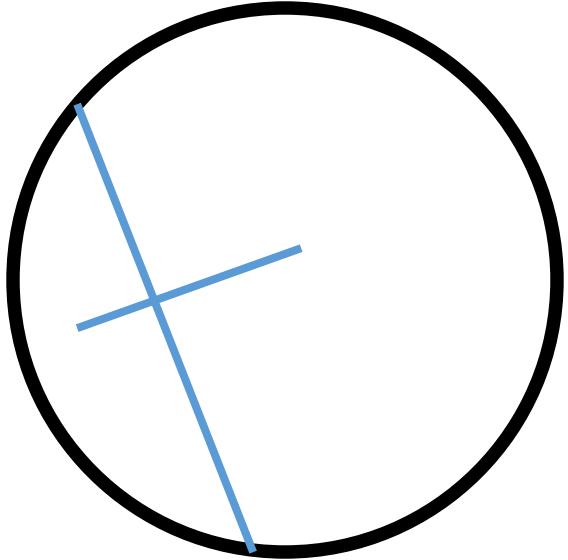
The aperture problem resolved



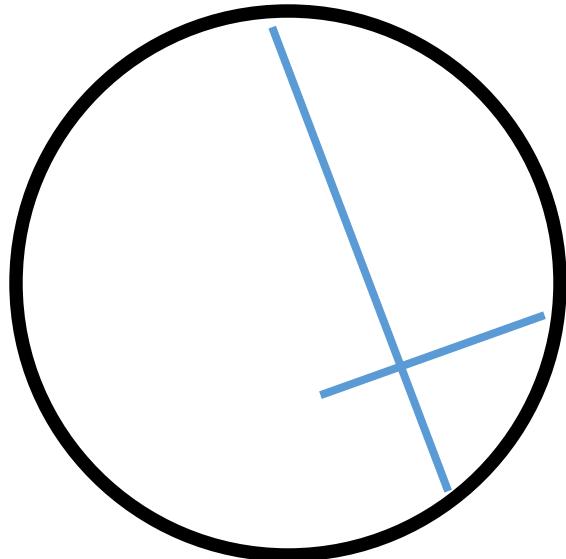
The aperture problem resolved



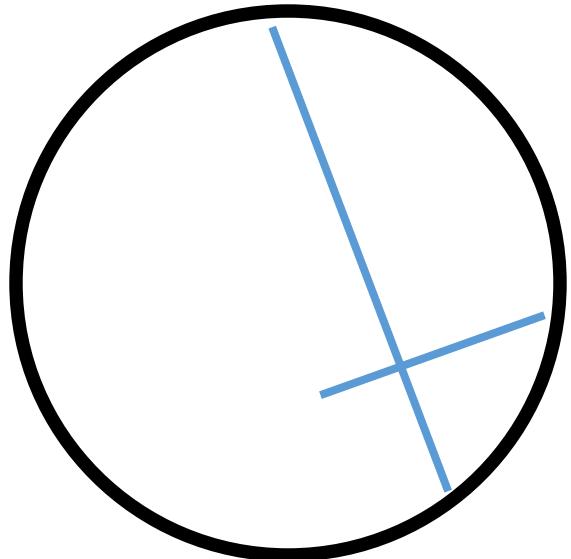
The aperture problem resolved



The aperture problem resolved



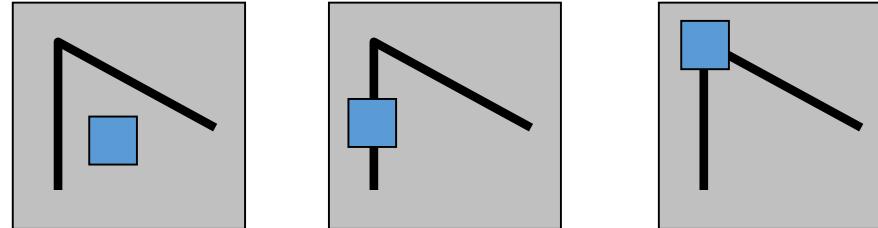
The aperture problem resolved



Perceived motion

Harris corner detection

- Generate a **cornerness score** for each image window



- Find points whose surrounding window gave large corner response ($f > \text{threshold}$)
- Take the points of local maxima, i.e., perform non-maximum suppression

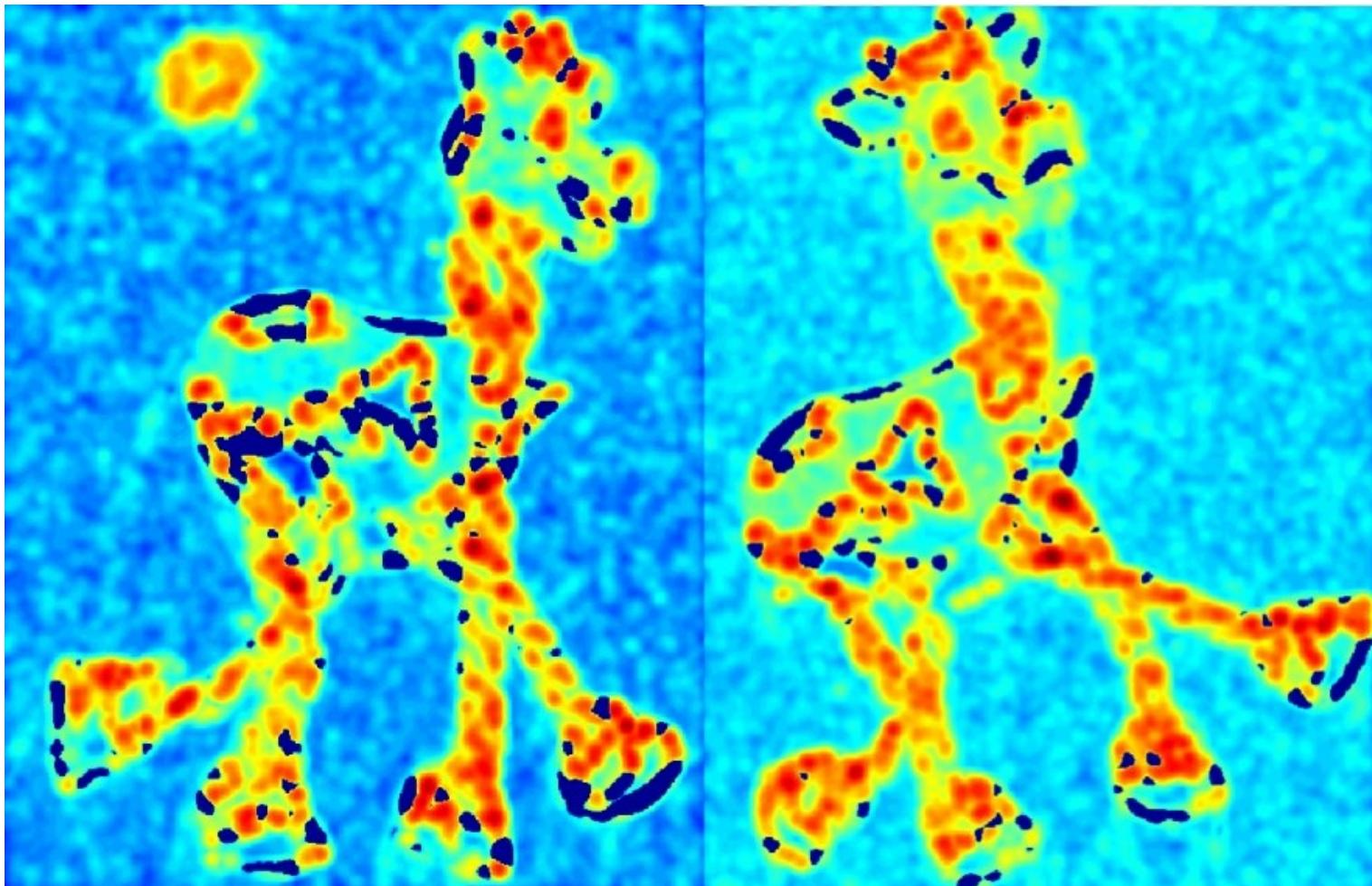
Harris detector steps



Source: James Hays

Harris detector steps

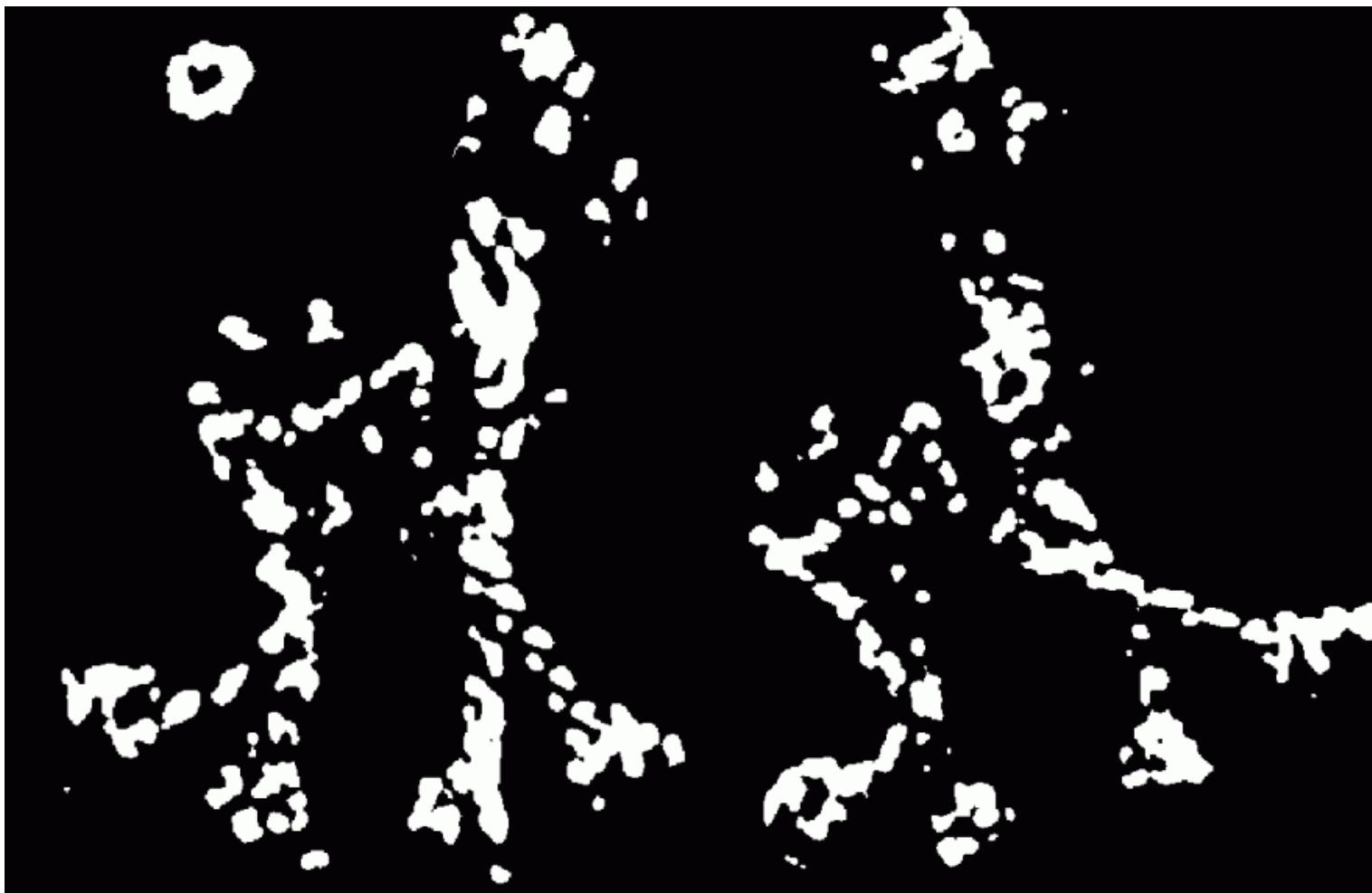
Compute corner response R



Source: James Hays

Harris detector steps

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



Source: James Hays

Harris detector steps

Take only the points of local maxima of R



Source: James Hays

Harris detector steps



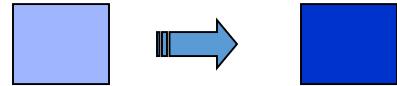
Source: James Hays

Invariance and covariance

We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations

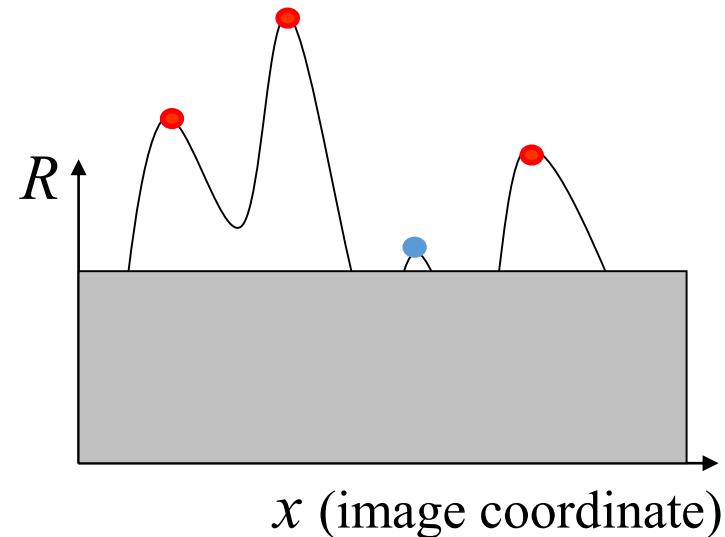
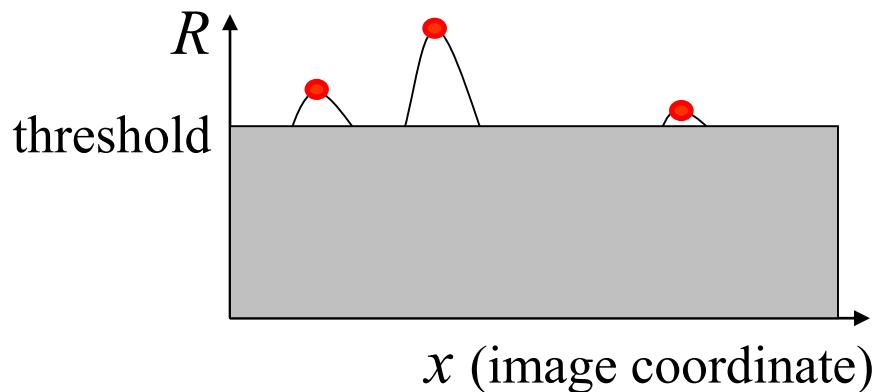
- **Invariance:** image is transformed and corner locations do not change
- **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

Affine intensity change



$$I \rightarrow a I + b$$

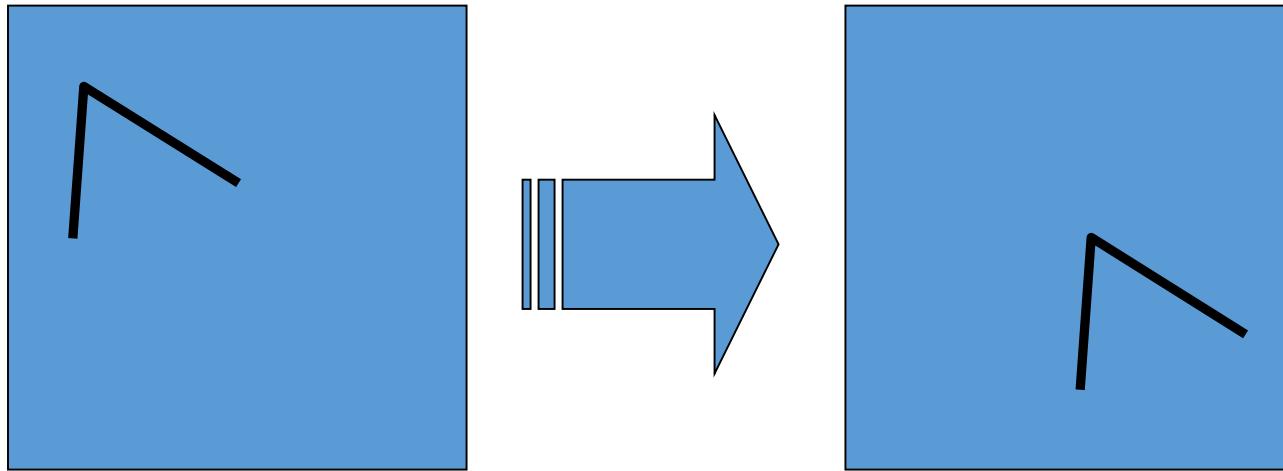
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a I$



Partially invariant to affine intensity change

Source: James Hays

Translation

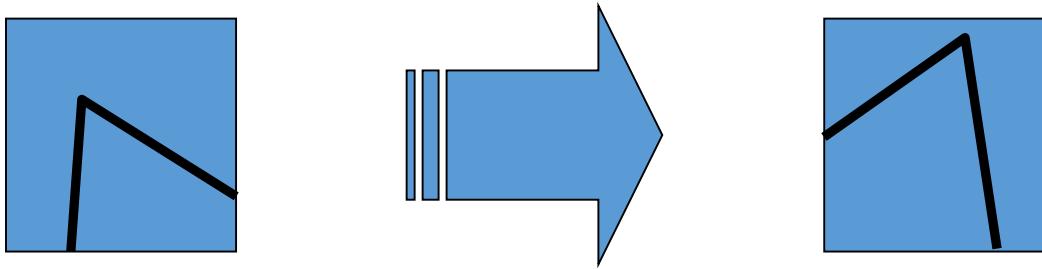


Derivatives and window function are shift-invariant

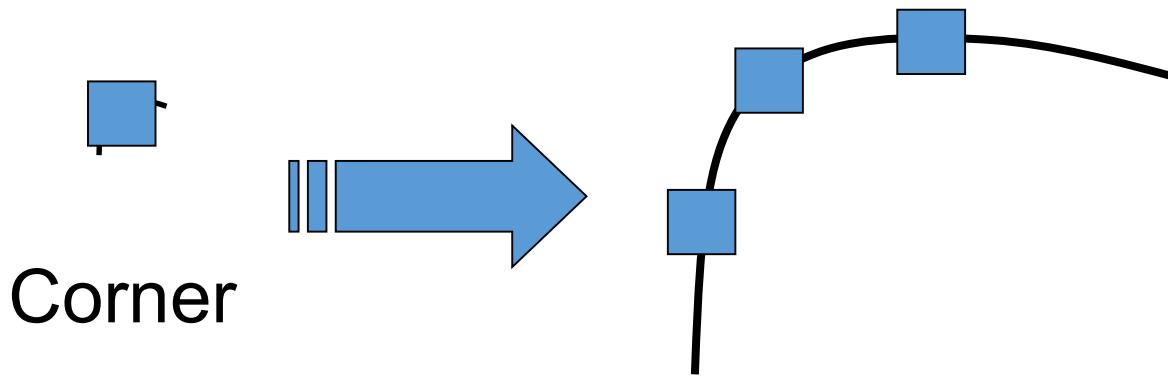
Corner location is covariant w.r.t. translation

Source: James Hays

Rotation and Scaling



Corner location is covariant w.r.t. rotation

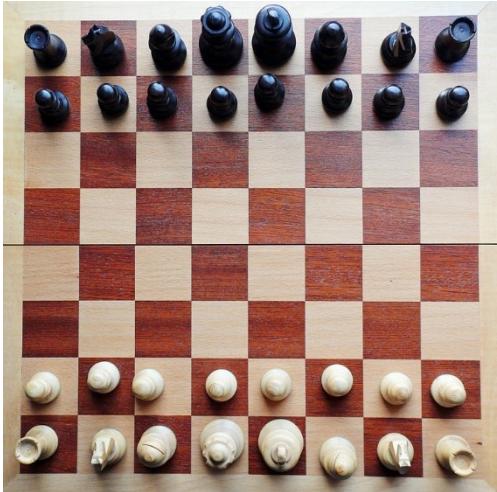


All points will be
classified as
edges

Corner location is not covariant to scaling!

Source: James Hays

Harris corner detection: esempi



Harris corner detection: esempi

```
from matplotlib import pyplot as plt
import urllib.request

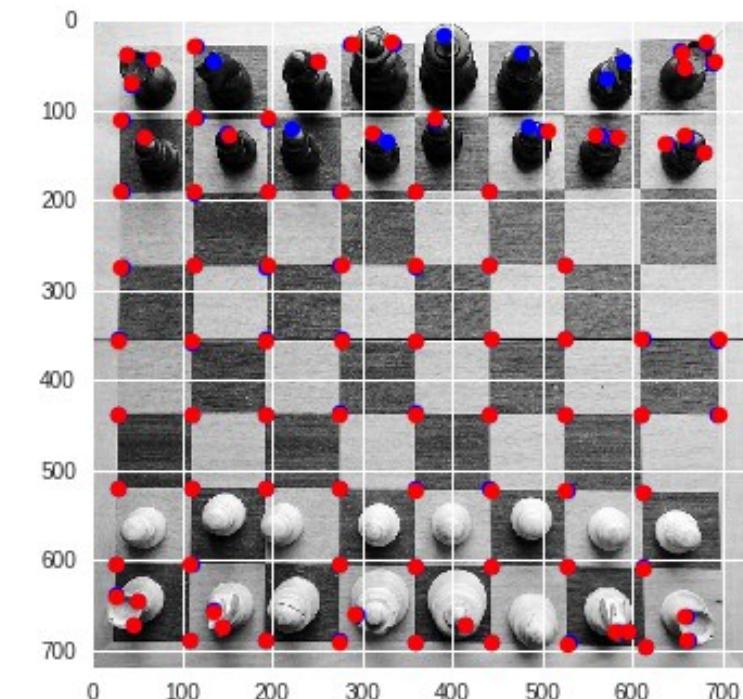
from skimage.io import imread
from skimage.color import rgb2gray
from skimage.feature import corner_harris, corner_subpix, corner_peaks

url = "https://dbloisi.github.io/corsi/images/chessboard-1.jpg"

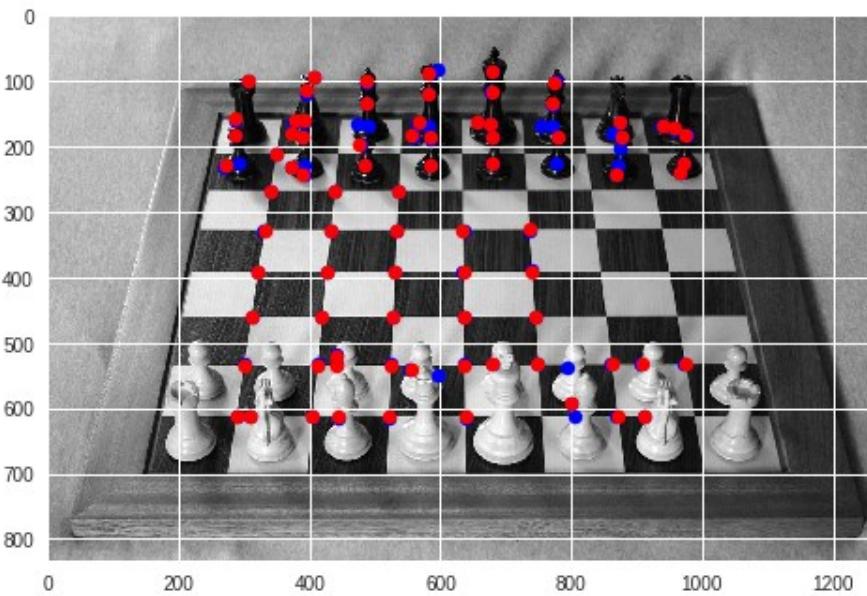
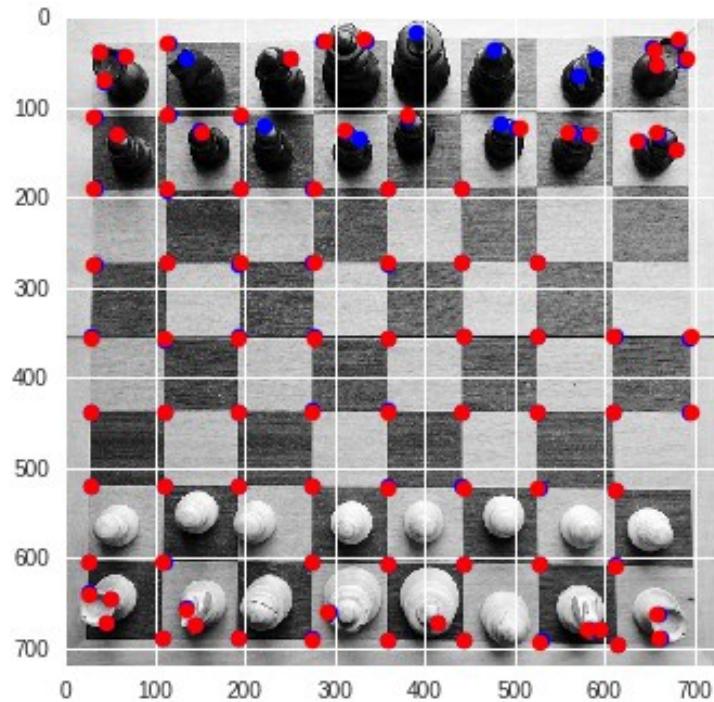
image = imread(urllib.request.urlopen(url))
image = rgb2gray(image)

coords = corner_peaks(corner_harris(image), min_distance=10)
coords_subpix = corner_subpix(image, coords, window_size=13)

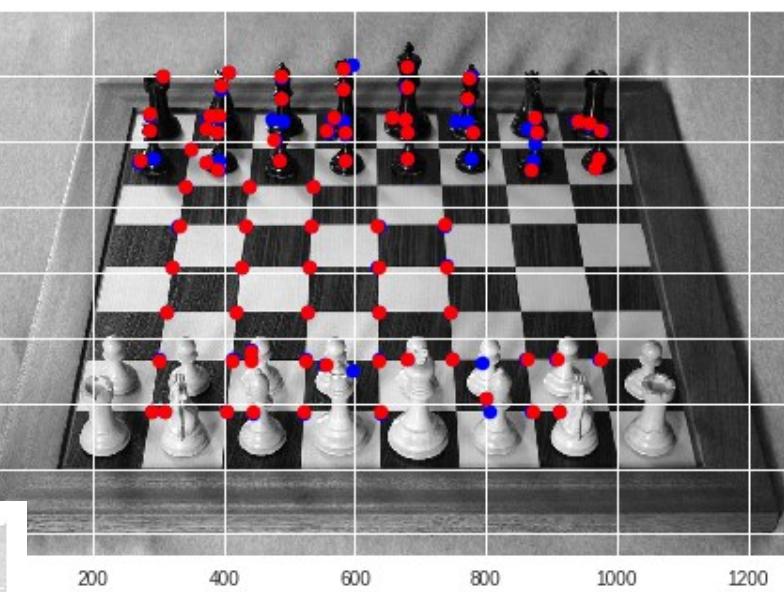
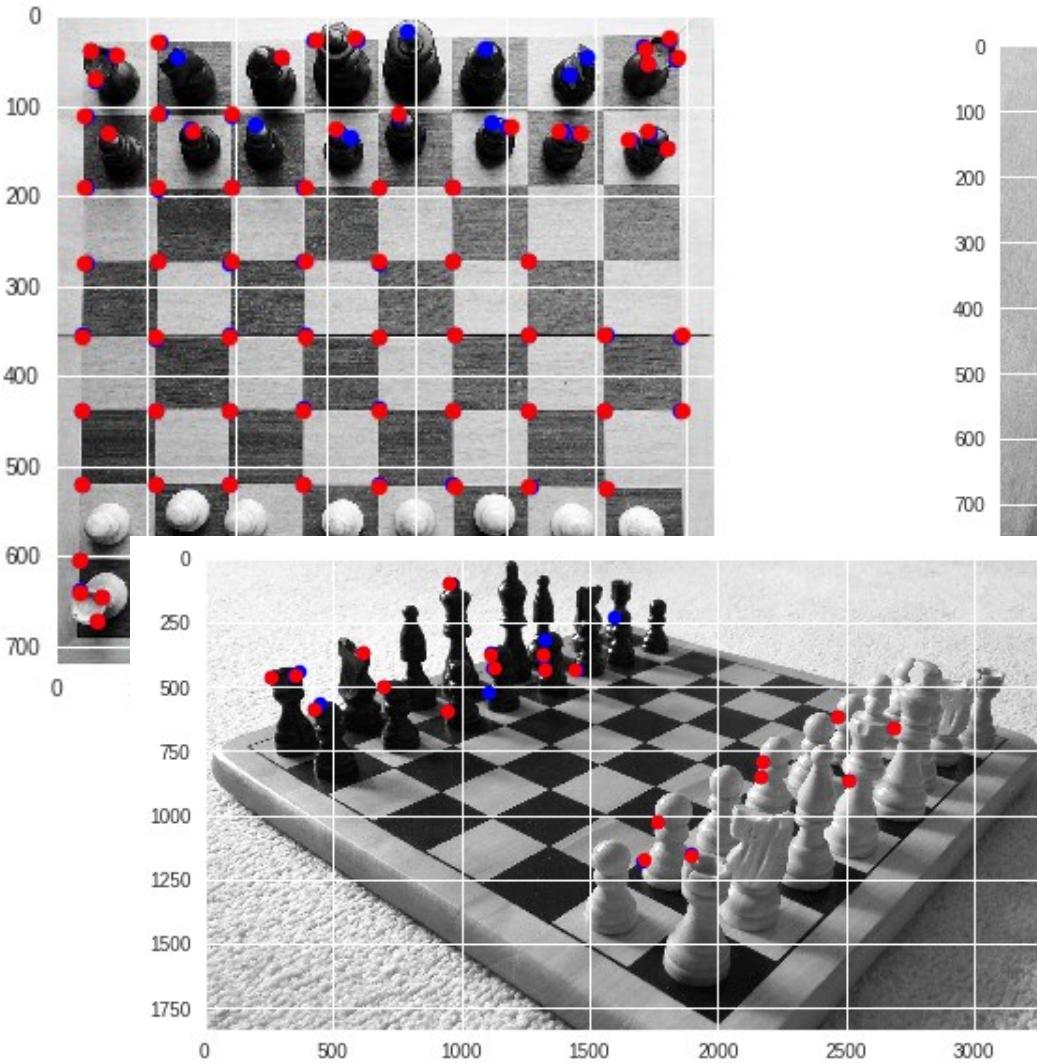
fig, ax = plt.subplots()
ax.imshow(image, interpolation='nearest', cmap=plt.cm.gray)
ax.plot(coords[:, 1], coords[:, 0], '.b', markersize=15)
ax.plot(coords_subpix[:, 1], coords_subpix[:, 0], '.r', markersize=15)
plt.show()
```



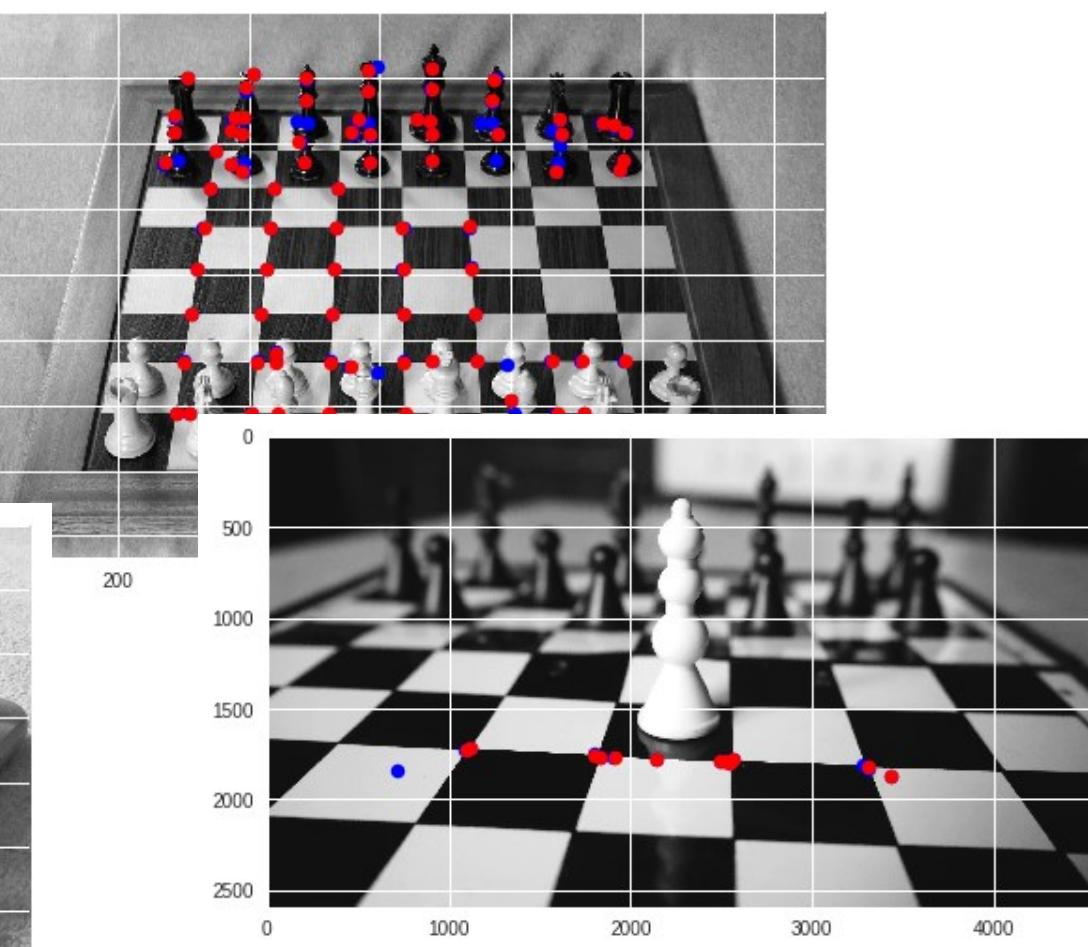
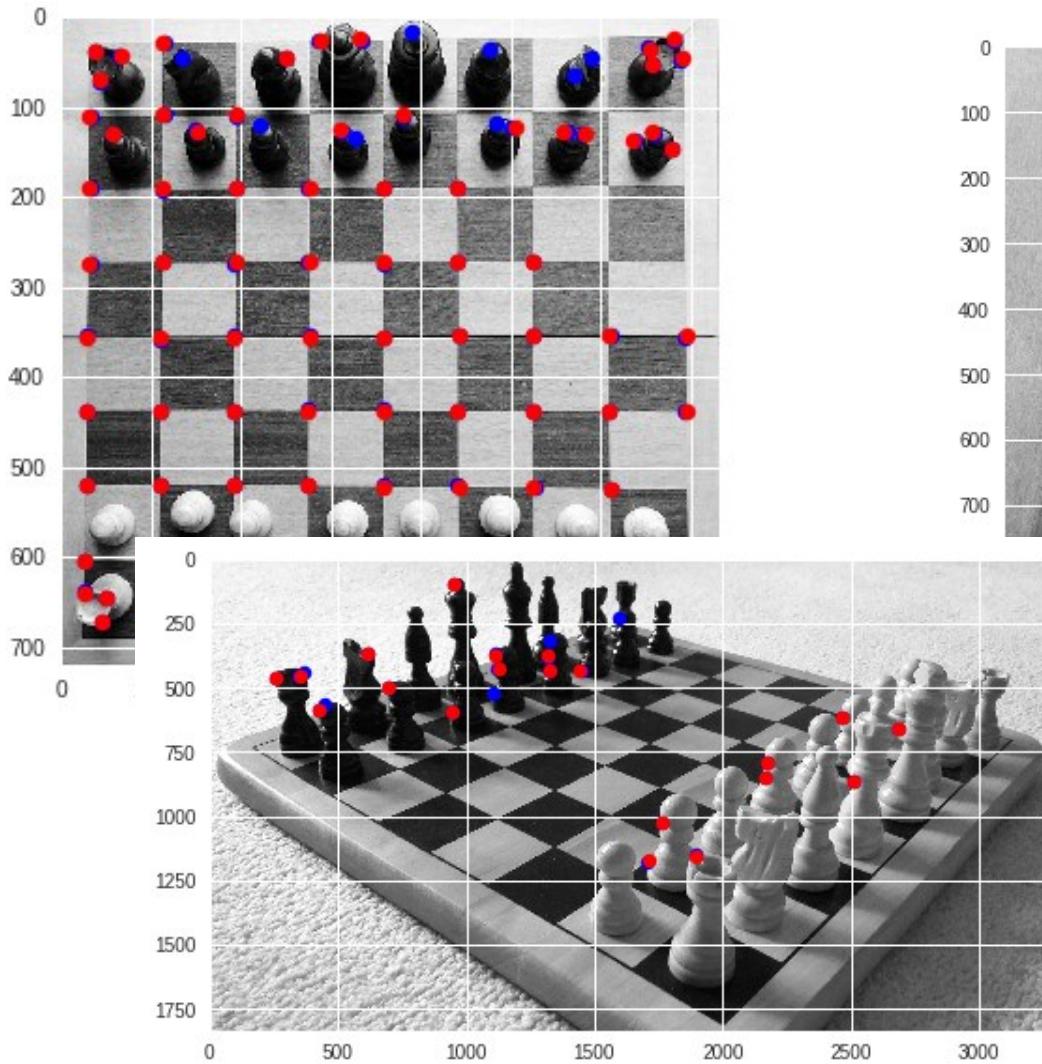
Harris corner detection: esempi



Harris corner detection: esempi

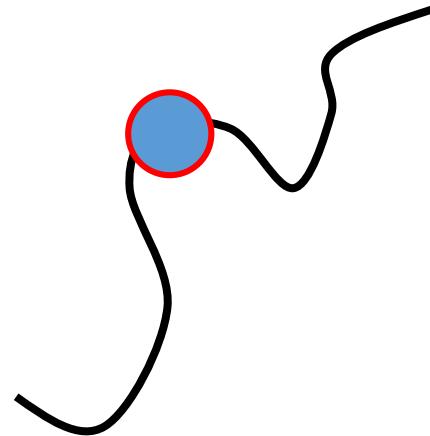


Harris corner detection: esempi



Scale invariant detection

Suppose you're looking for corners

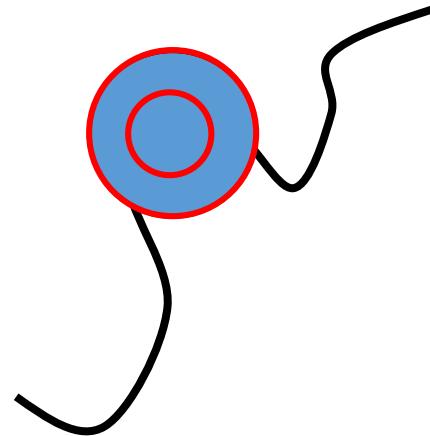


Key idea: find scale that gives local maximum of f

- in both position and scale
- One definition of f : the Harris operator

Scale invariant detection

Suppose you're looking for corners

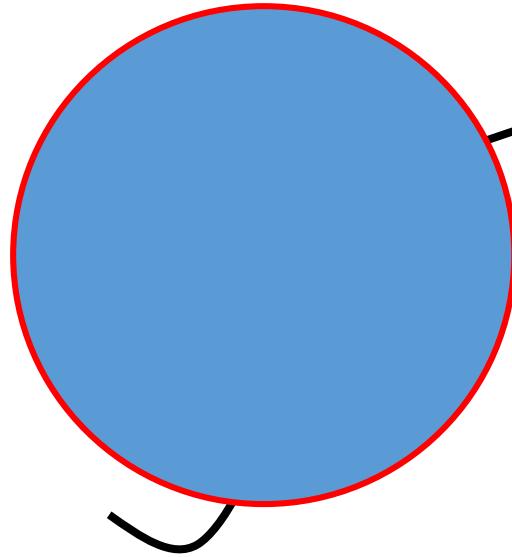


Key idea: find scale that gives local maximum of f

- in both position and scale
- One definition of f : the Harris operator

Scale invariant detection

Suppose you're looking for corners

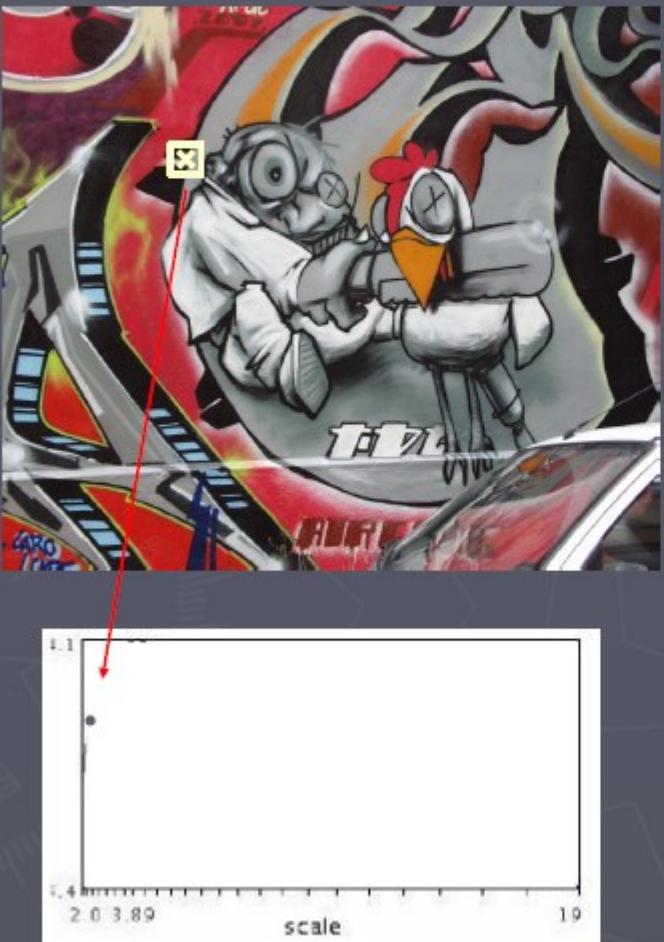


Key idea: find scale that gives local maximum of f

- in both position and scale
- One definition of f : the Harris operator

Automatic scale selection

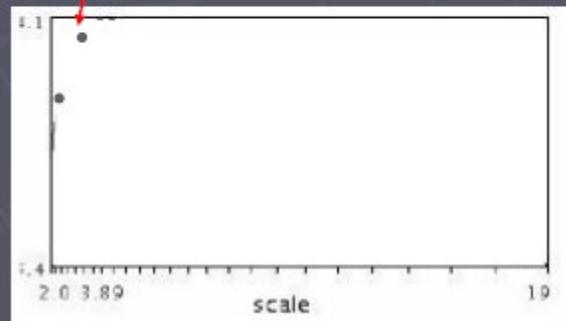
Lindeberg et al., 1996



$$f(I_{i_1 \dots i_m}(x, \sigma))$$

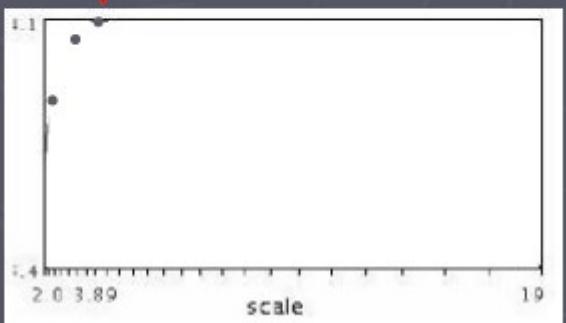
Slide from Tinne Tuytelaars

Automatic scale selection



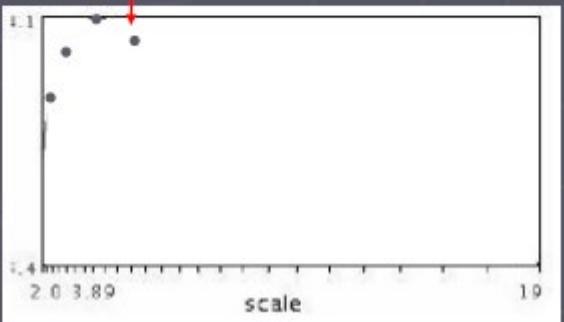
$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Automatic scale selection



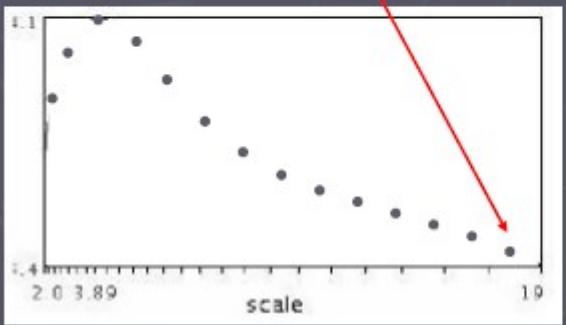
$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Automatic scale selection



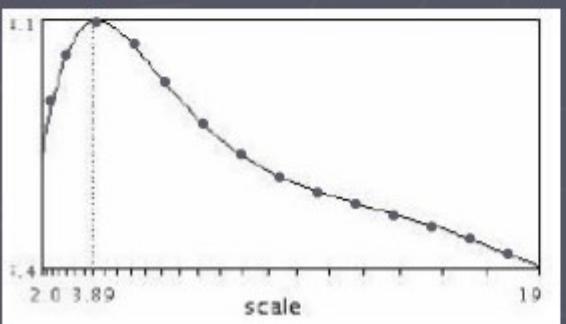
$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Automatic scale selection



$$f(I_{i_1\dots i_m}(x, \sigma))$$

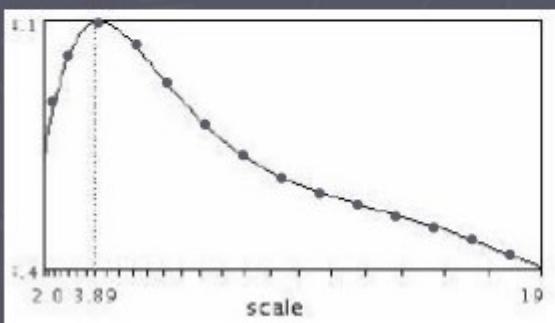
Automatic scale selection



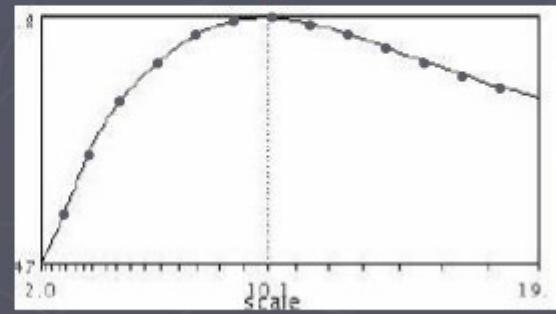
$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Automatic scale selection

Function responses for increasing scale



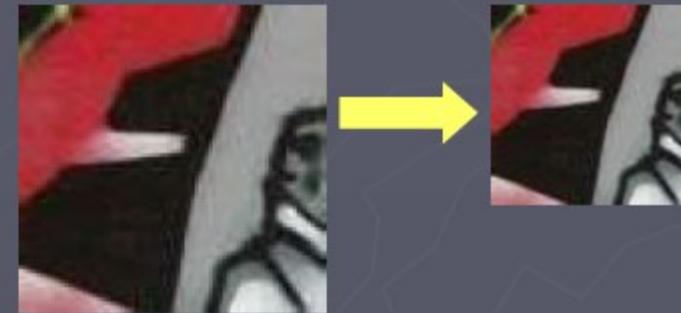
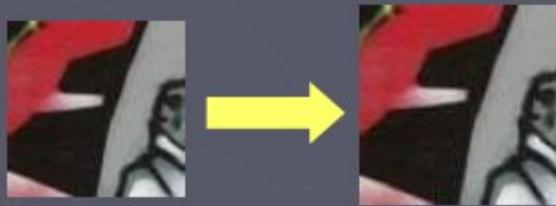
$$f(I_{i_1\dots i_m}(x, \sigma))$$



$$f(I_{i_1\dots i_m}(x', \sigma'))$$

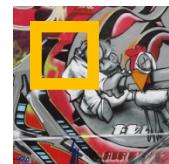
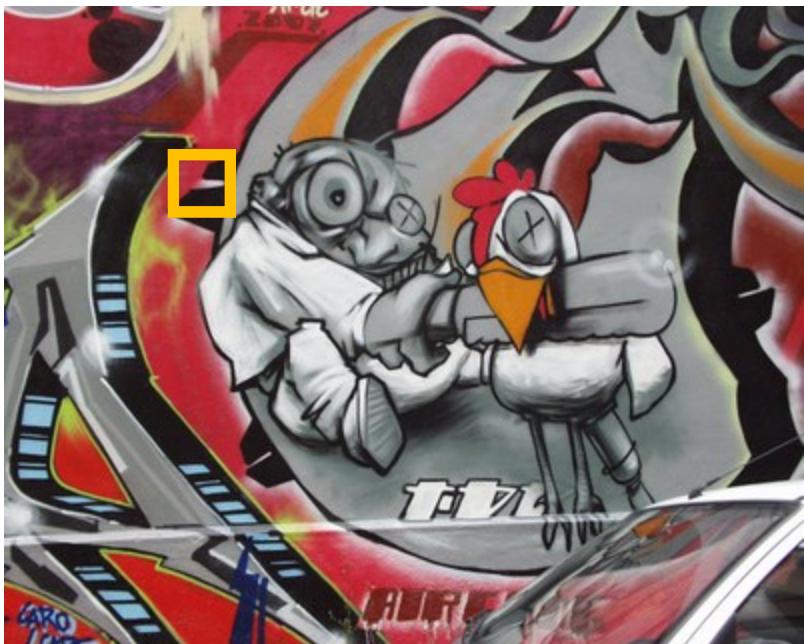
Automatic scale selection

Normalize: rescale to fixed size



Implementation

- Instead of computing f for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid



(sometimes need to create in-between levels, e.g. a $\frac{3}{4}$ -size image)

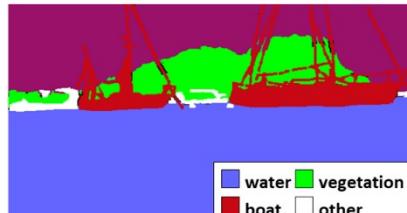
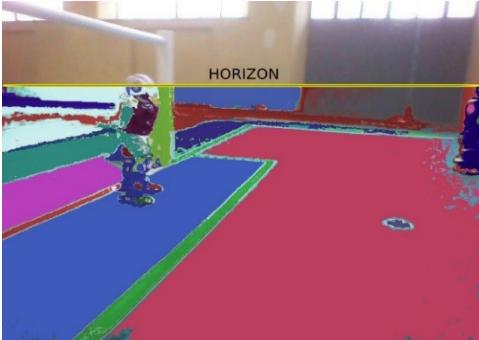
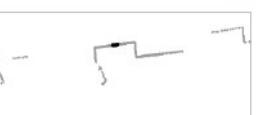


UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Corso di Sistemi Informativi
A.A. 2018/19

Features

Aprile 2019



Docente
Domenico Daniele Bloisi