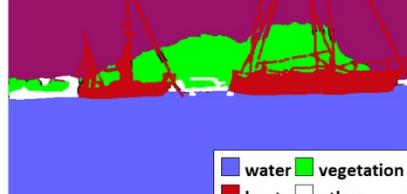
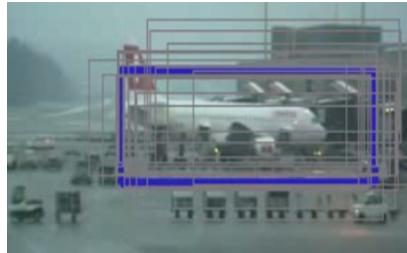
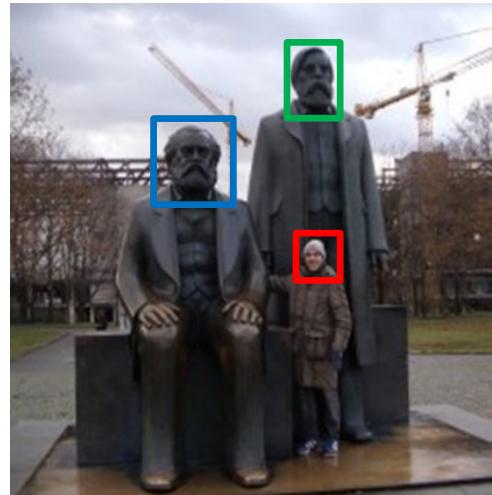
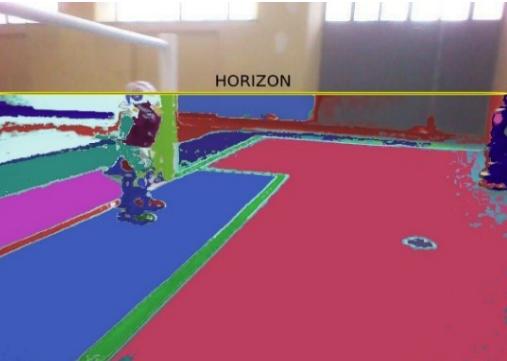




**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Visione e Percezione

Features



Docente
Domenico D. Bloisi

Domenico Daniele Bloisi

- Ricercatore RTD B

Dipartimento di Matematica, Informatica
ed Economia

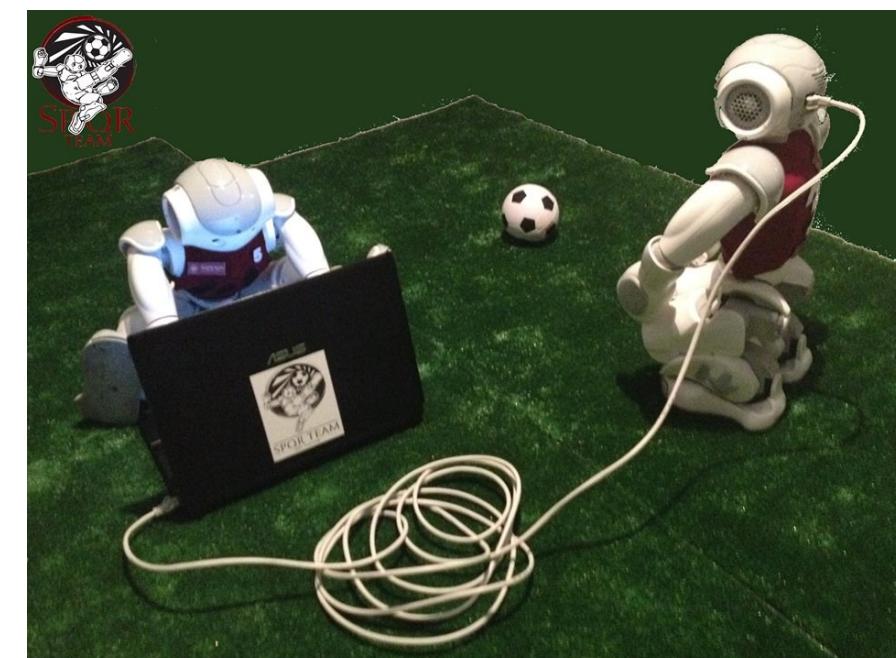
Università degli studi della Basilicata

<http://web.unibas.it/bloisi>

- SPQR Robot Soccer Team

Dipartimento di Informatica, Automatica
e Gestionale Università degli studi di
Roma “La Sapienza”

<http://spqr.diag.uniroma1.it>

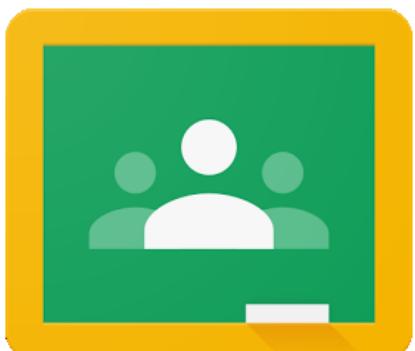


Informazioni sul corso

- Home page del corso
<http://web.unibas.it/bloisi/corsi/visione-e-percezione.html>
- Docente: Domenico Daniele Bloisi
- Periodo: **Il semestre** marzo 2021 – giugno 2021

Martedì 17:00-19:00 (Aula COPERNICO)

Mercoledì 8:30-10:30 (Aula COPERNICO)



Codice corso Google Classroom:

<https://classroom.google.com/c/Njl2MjA4MzgzNDFa?cjc=xgolays>

Ricevimento

- Su appuntamento tramite Google Meet

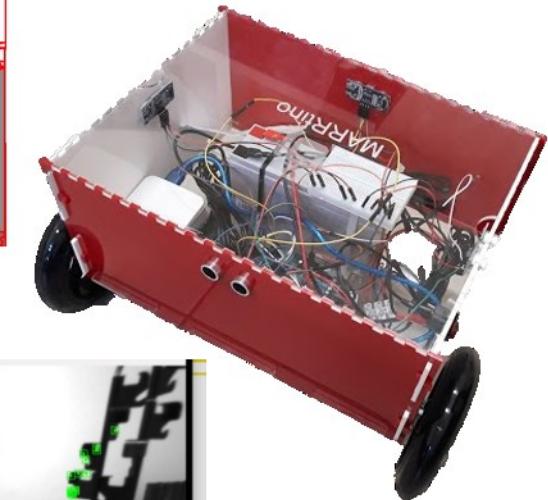
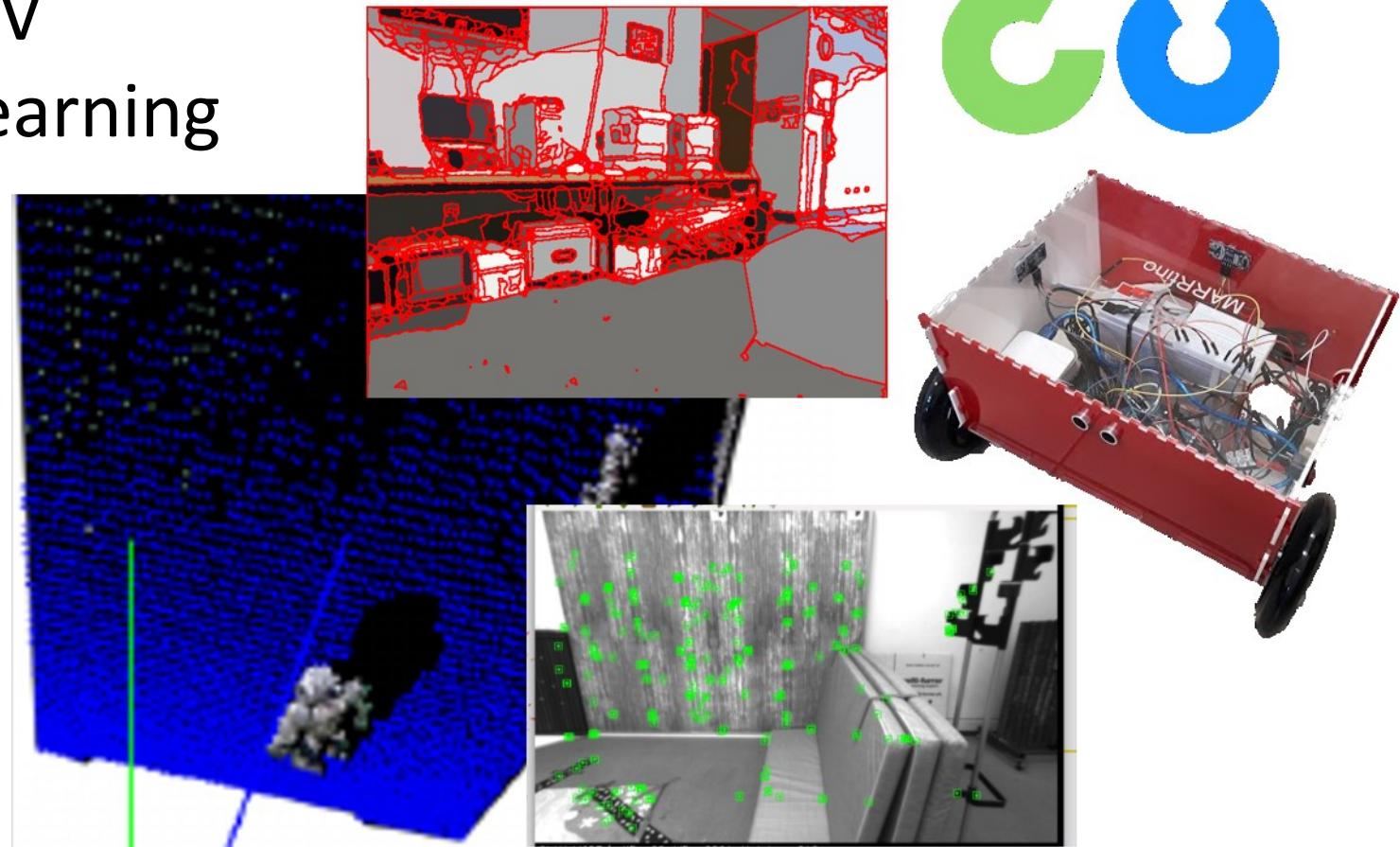
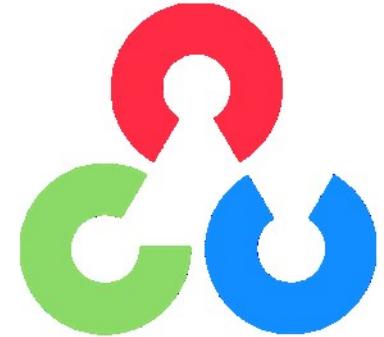
Per prenotare un appuntamento inviare
una email a

domenico.bloisi@unibas.it



Programma – Visione e Percezione

- Introduzione al linguaggio Python
- [Elaborazione delle immagini con Python](#)
- Percezione 2D – OpenCV
- Introduzione al Deep Learning
- ROS
- Il paradigma publisher and subscriber
- Simulatori
- Percezione 3D - PCL

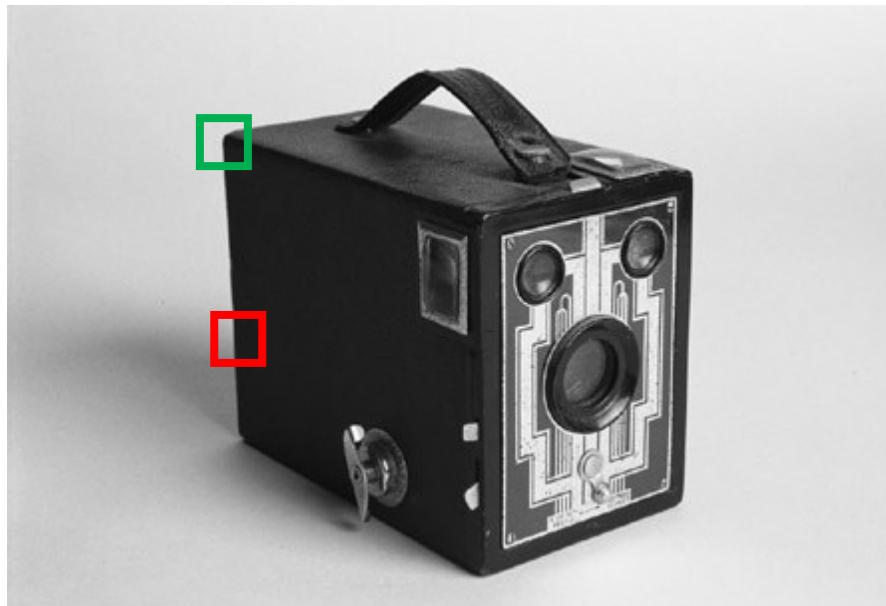
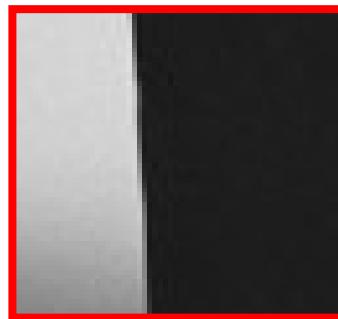


Riferimenti

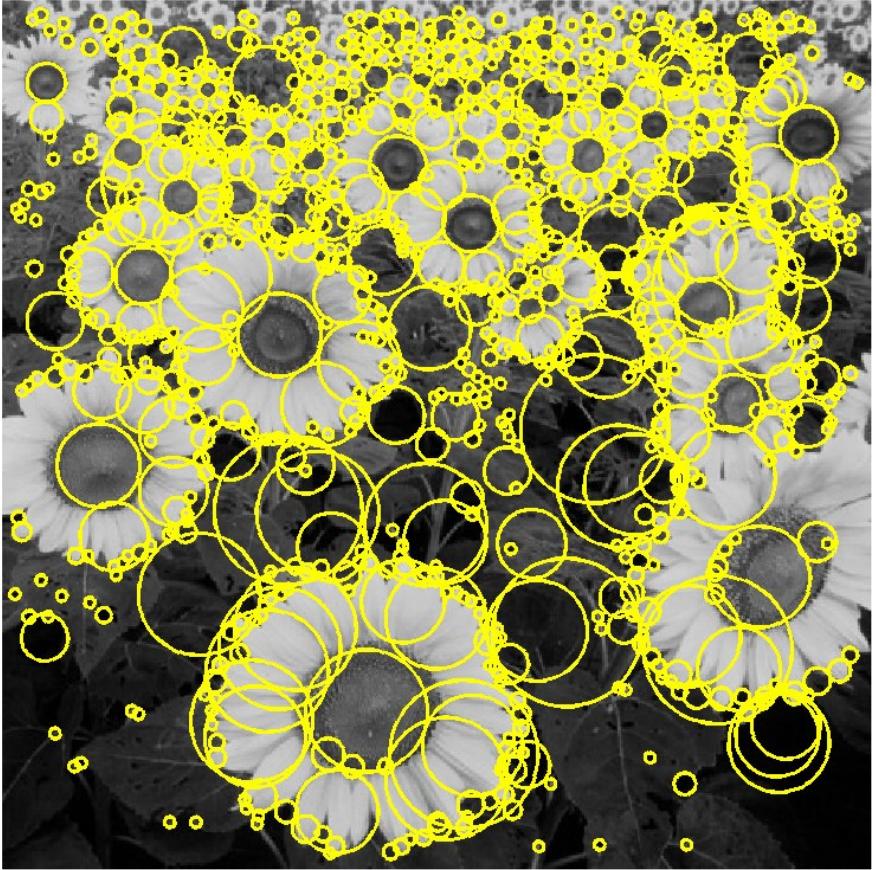
- La maggior parte di queste slide sono adattate da Noah Snavely - CS5670: Computer Vision
["Lecture 4: Intro to local features and Harris corner detection"](#)
- Per le slide adattate da altri autori vengono fornite le relative citazioni
- I contenuti fanno riferimento al capitolo 4 del libro "Computer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

Corners

Large gradients in more than one direction



Blobs



A blob is a region of an image in which some properties like intensity or colour are approximately constant

Distinctive Features

- A point is distinctive when it looks different from its neighbors
- A blob also looks different from neighbors at different scales

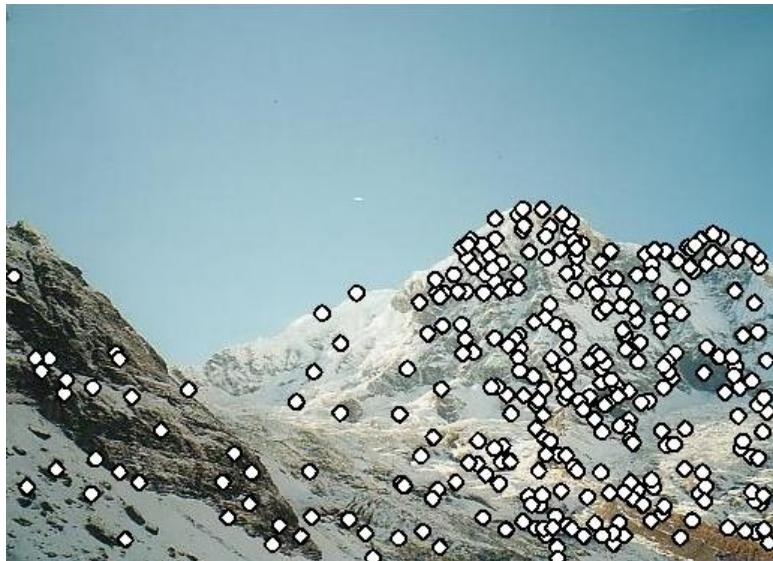
Feature matching

We have two images, how do we combine them?



Feature matching

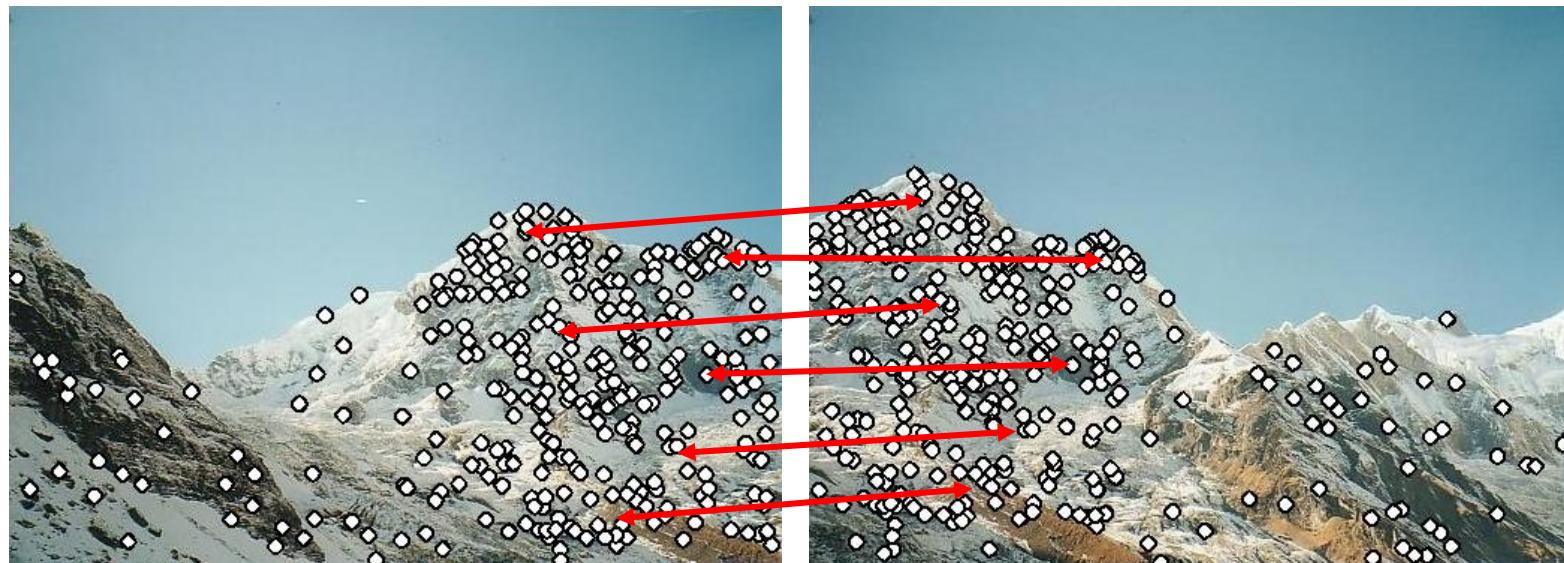
Step 1: extract features



Feature matching

Step 1: extract features

Step 2: match features



Feature matching

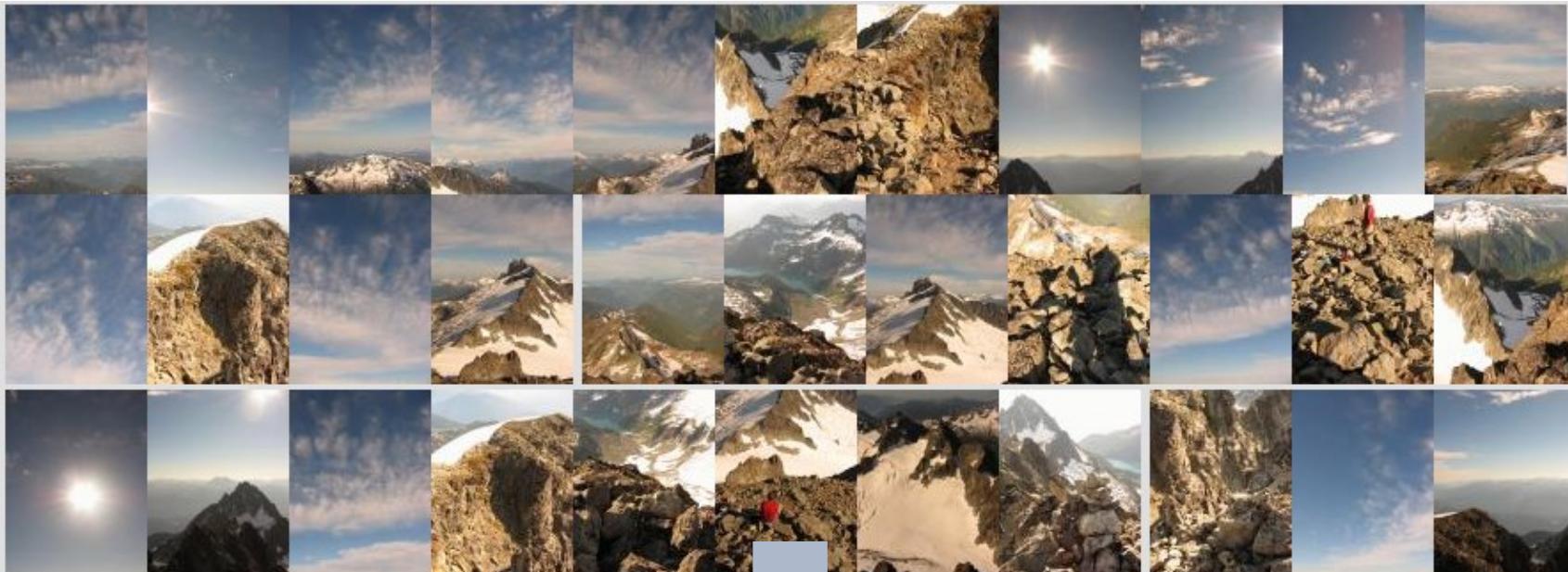
Step 1: extract features

Step 2: match features

Step 3: align images



Automatic panoramas



Credit: Matt Brown

Automatic panoramas



Visual SLAM



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

Image matching



by [Diva Sian](#)



by [swashford](#)

Harder case



by [Diva Sian](#)



by [scgbt](#)

Image Matching Challenge



Image matching on Mars

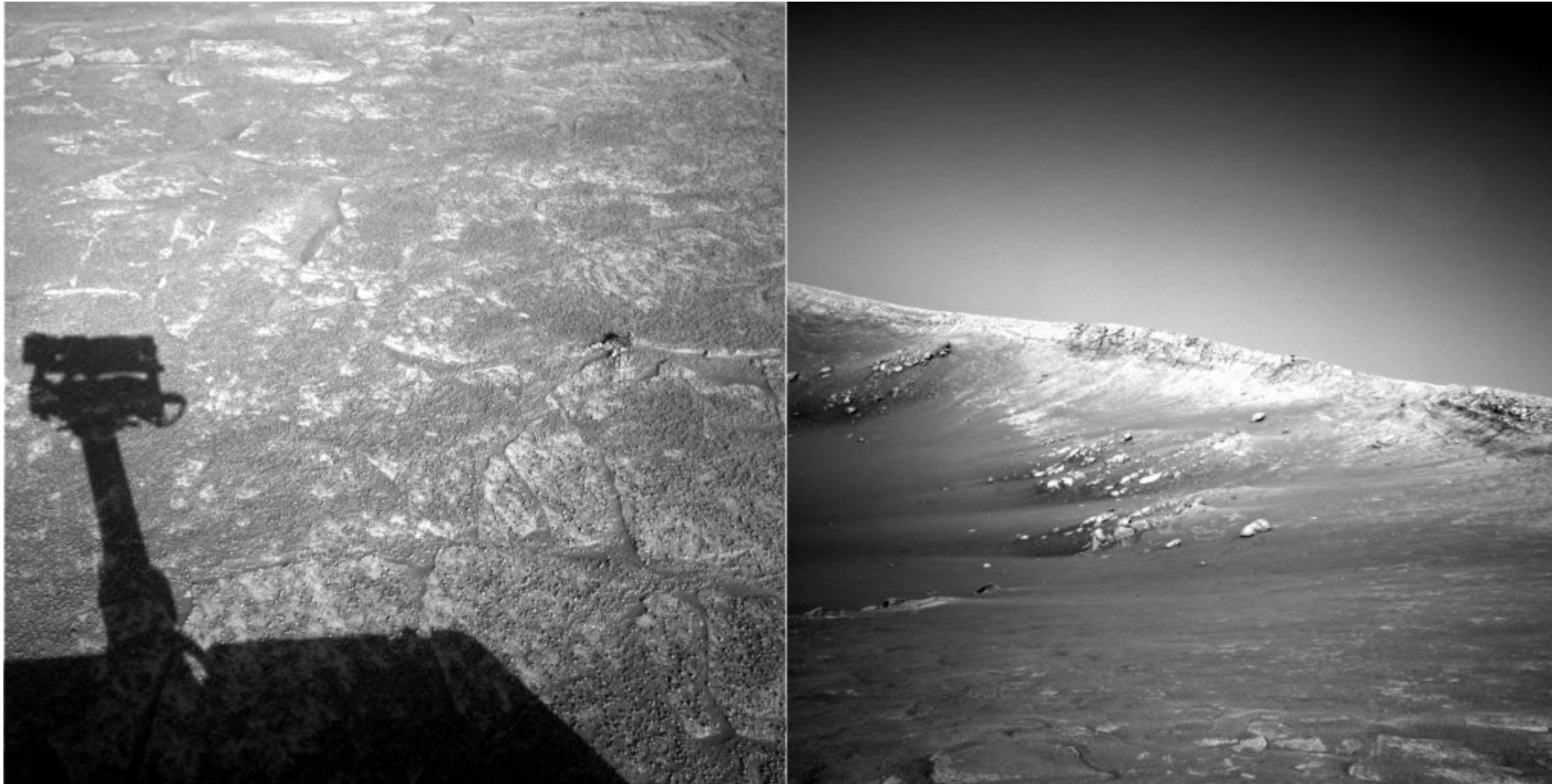
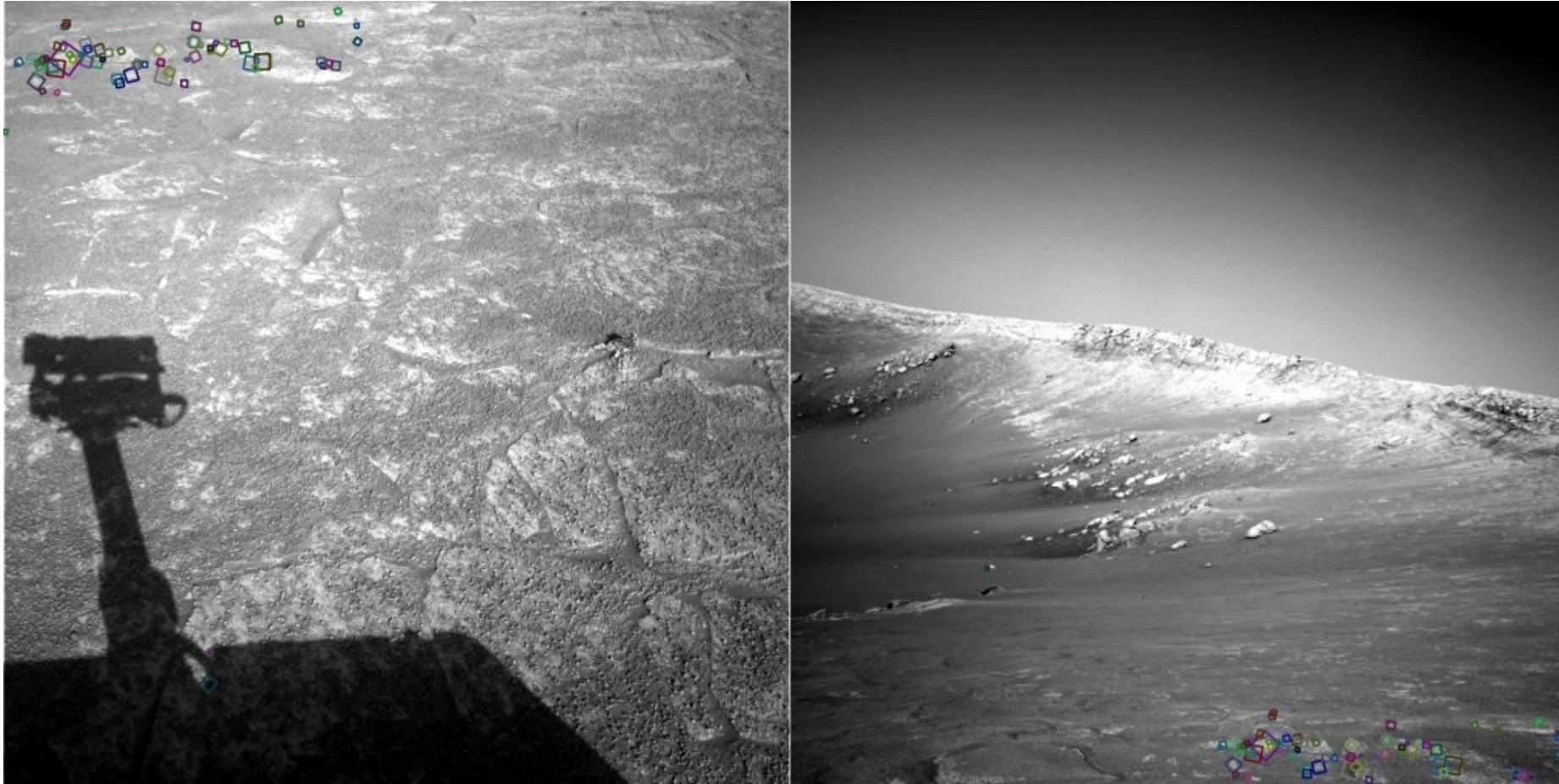


Image matching on Mars



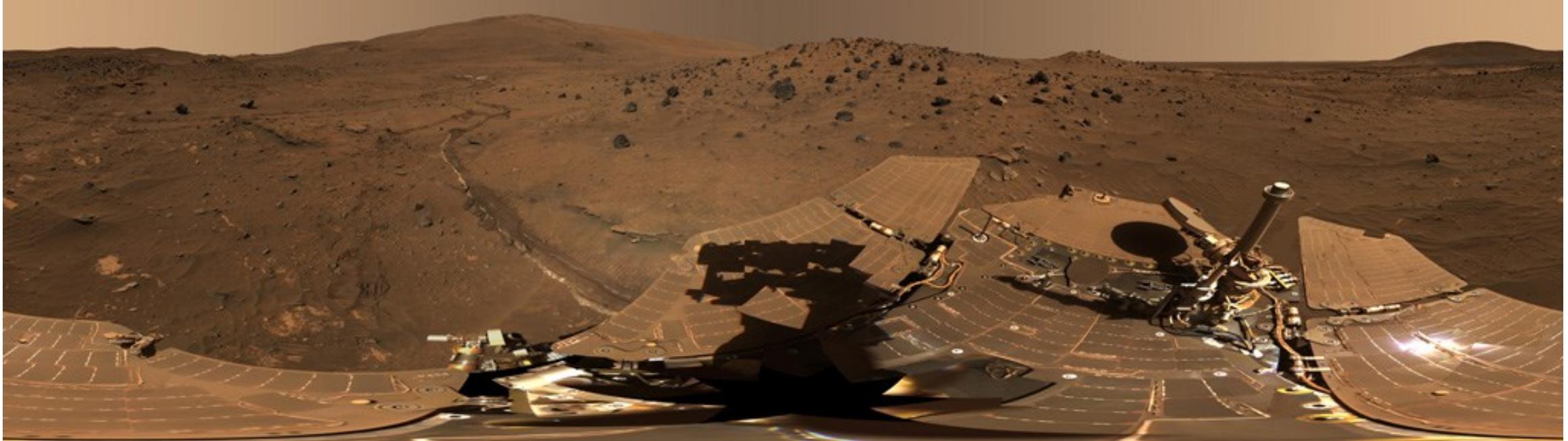
NASA Mars Rover images
with SIFT feature matches

Image matching on Mars



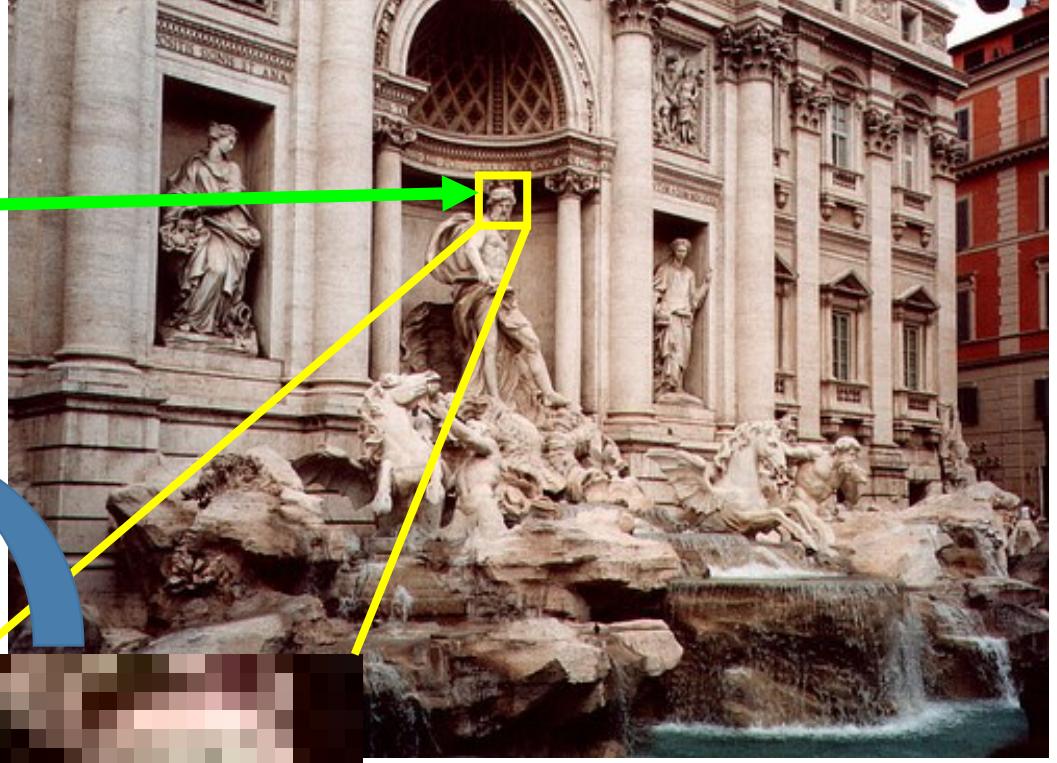
NASA Mars Rover images
with SIFT feature matches

Spirit Mars Rover Panorama



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

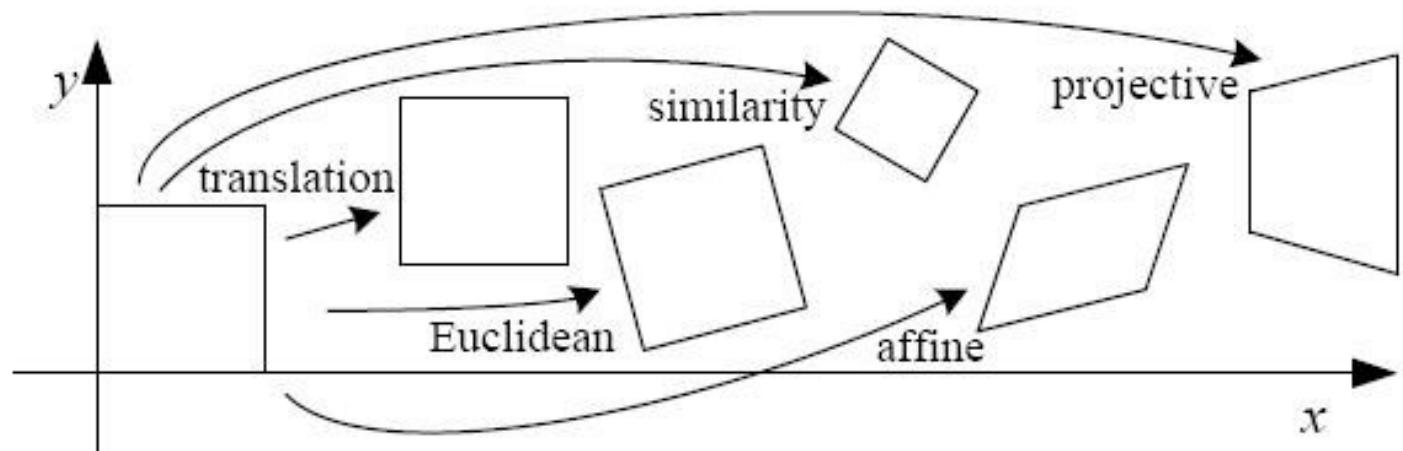
Challenges in feature matching



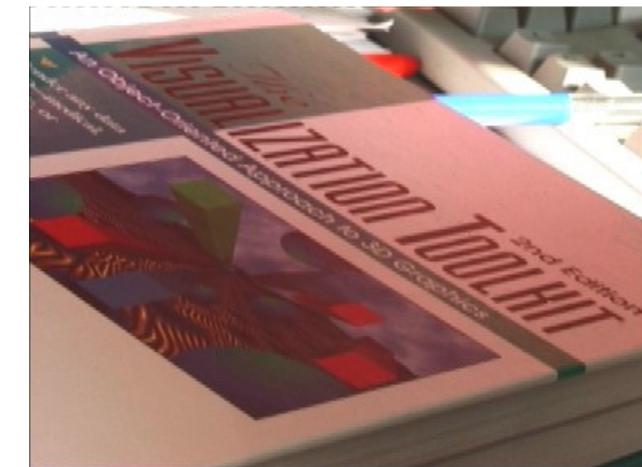
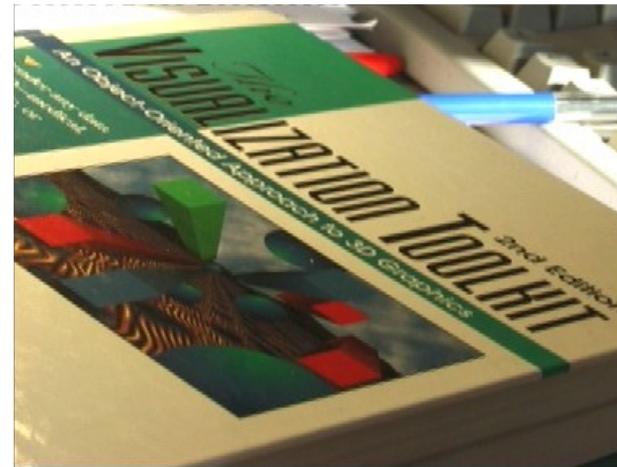
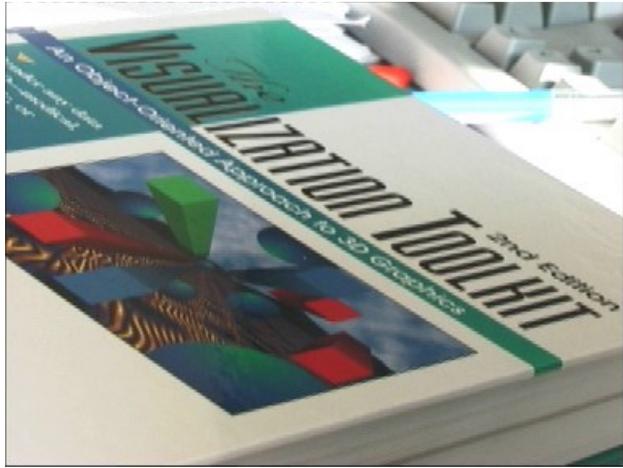
Geometric transformations

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

Only holds
for planar patches



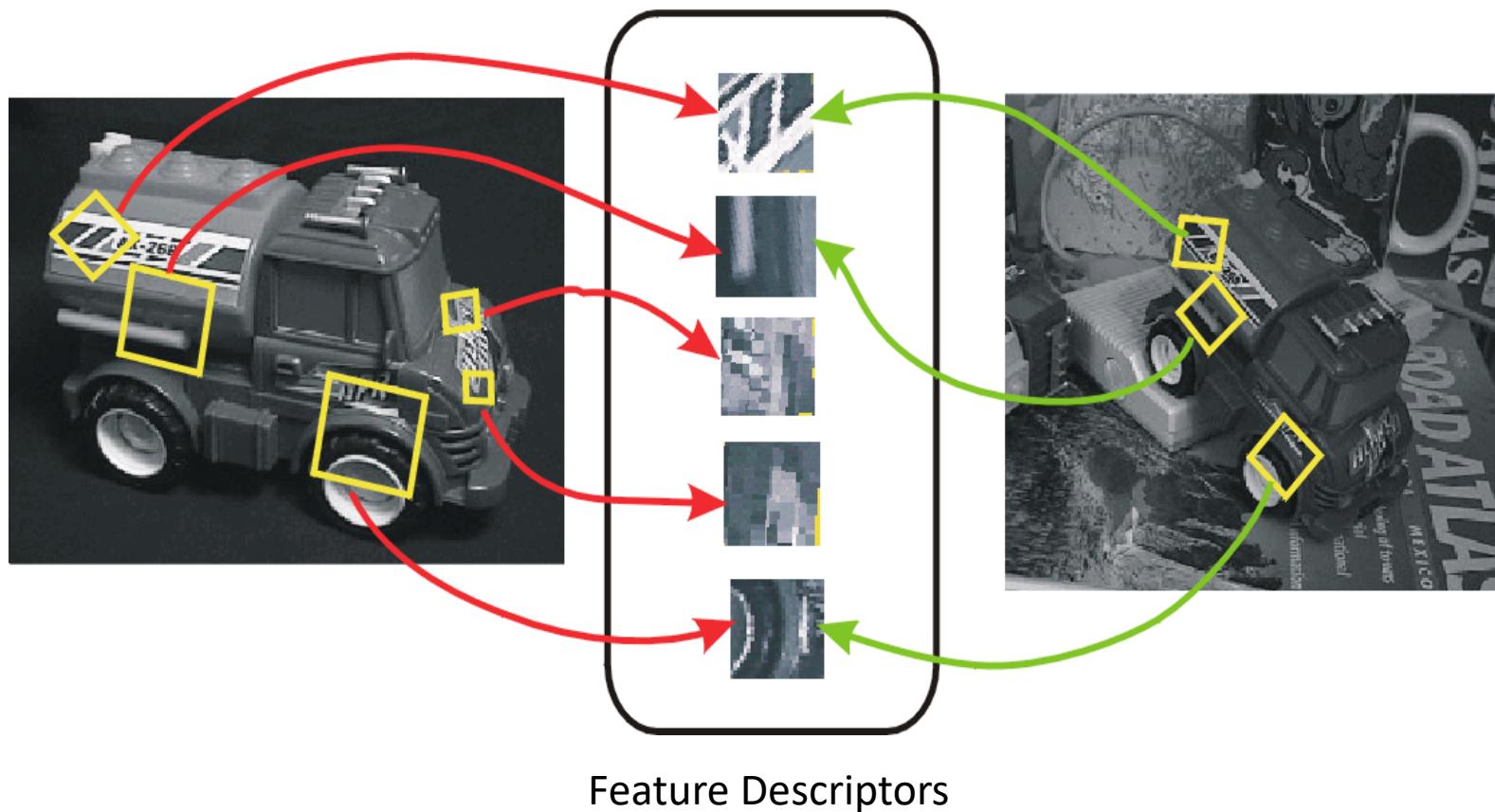
Photometric transformations



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

More motivations...

Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking (e.g. for AR)
- Object recognition
- Image retrieval
- Robot/car navigation
- ... other



<https://www.youtube.com/watch?v=Kr0W7io5rHw>

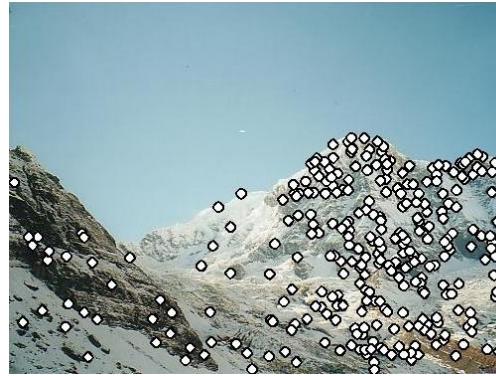
Approach

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

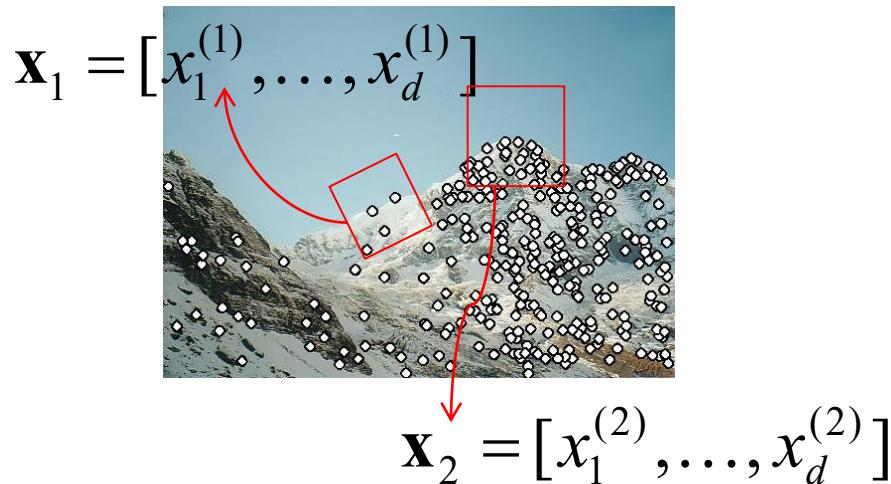
Feature tracking: track it, when motion

Local features: main components

- 1) Detection: Identify the interest points



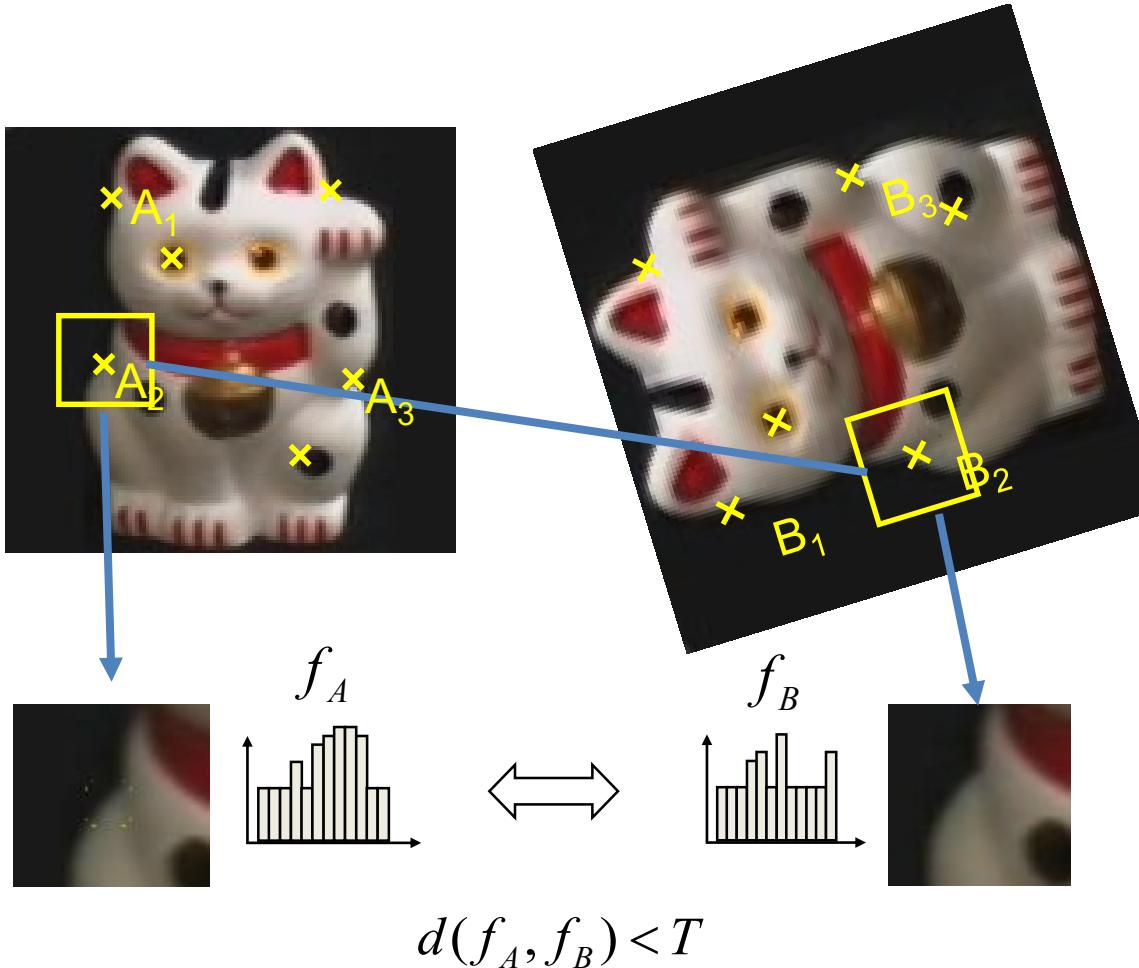
- 2) Description: Extract vector feature descriptor surrounding each interest point



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



Overview of Keypoint Matching



What points to choose?



Low-texture region?



NO

- Textureless patches are nearly impossible to localize
- Gradients have small magnitude

Edge?



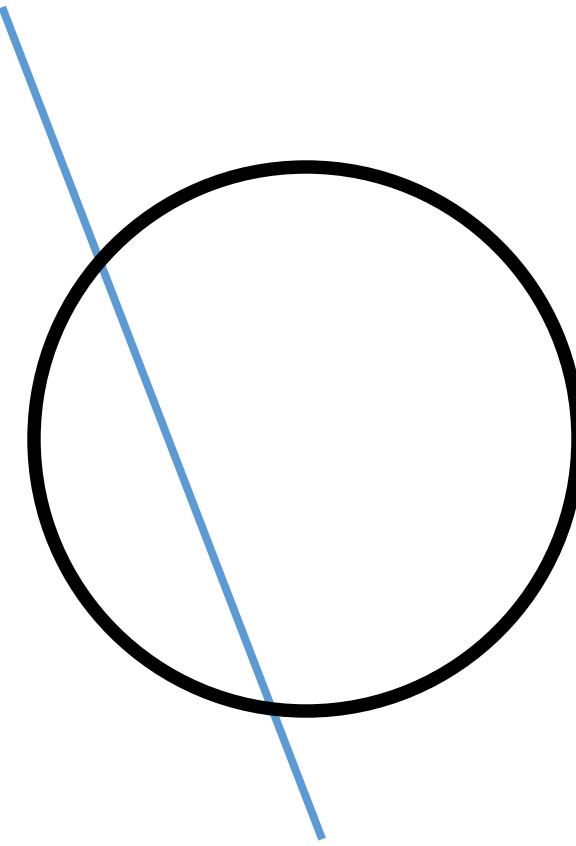
Gradients
very large
(or very
small)

NO

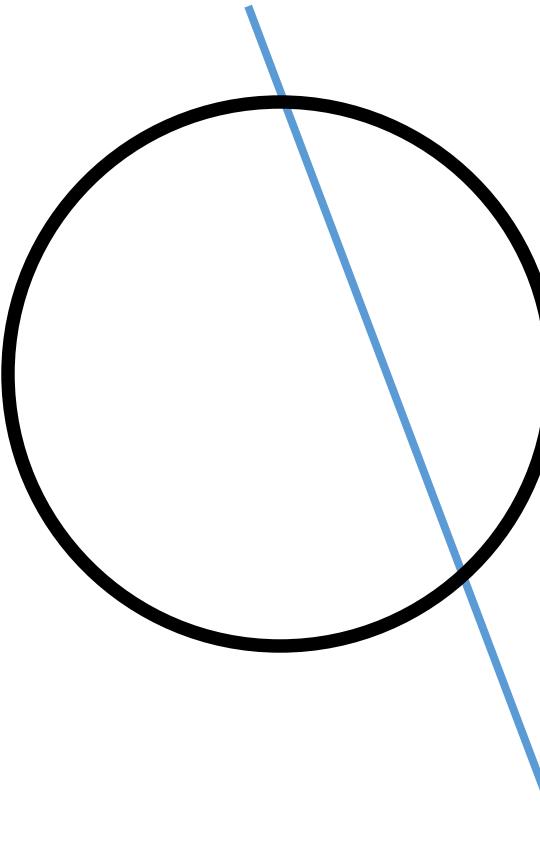
- Patches with large contrast changes (gradients) are easier to localize, BUT
- Straight line segments at a single orientation suffer from the aperture problem, i.e., it is only possible to align the patches along the direction normal to the edge direction

Source: Derek Hoiem

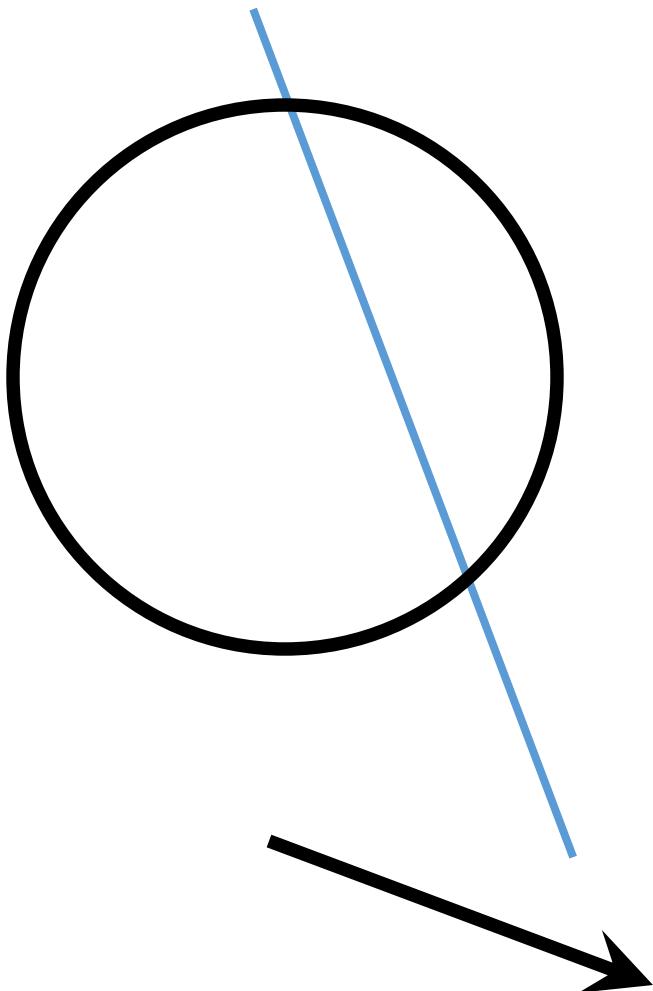
The aperture problem



The aperture problem



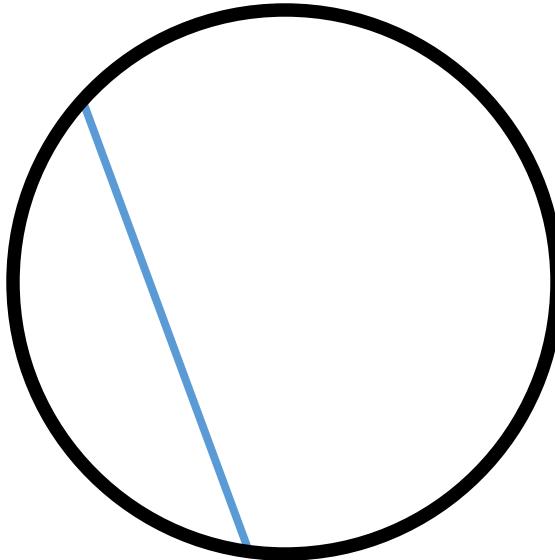
The aperture problem



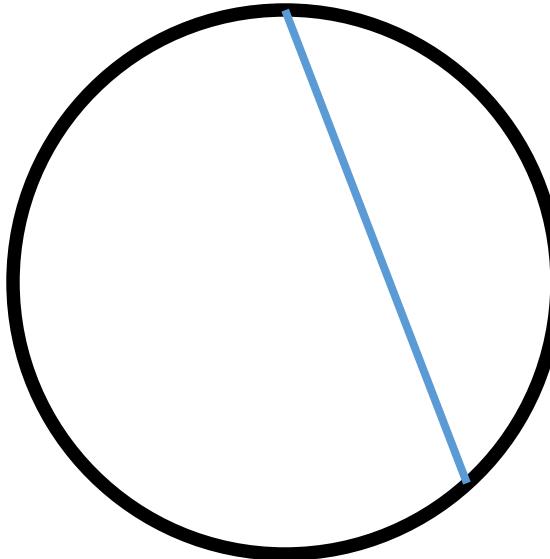
Actual motion

Source: Derek Hoiem

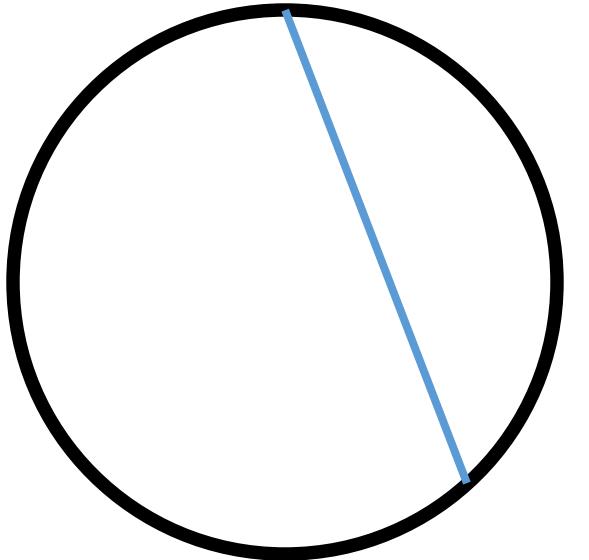
The aperture problem



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

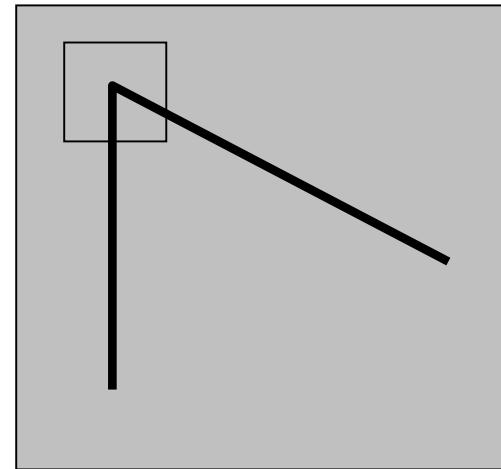
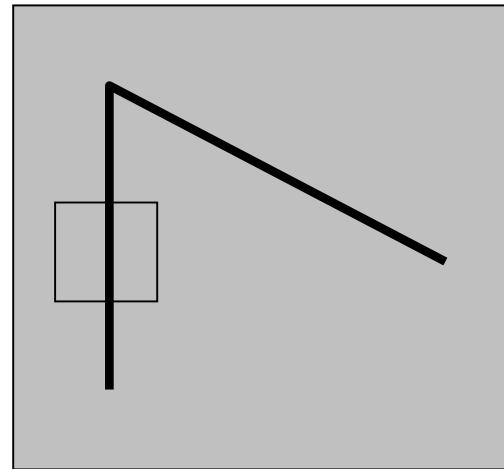
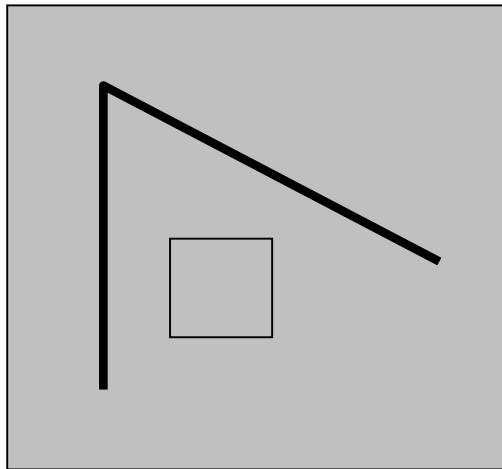
- Lead to unambiguous matches in other images

How to define “unusual”?

Want uniqueness

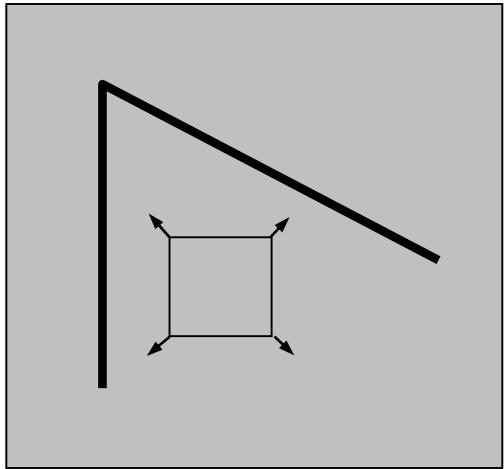
Suppose we only consider a small window of pixels

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

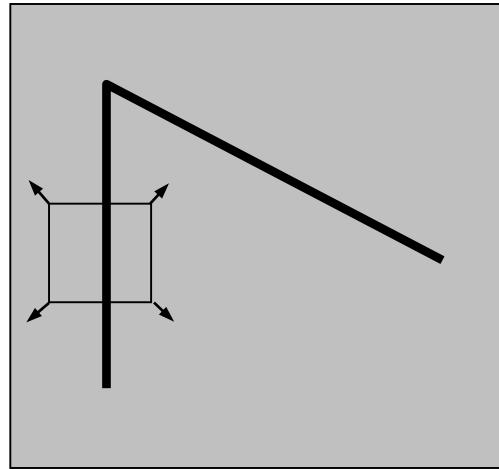


Want uniqueness

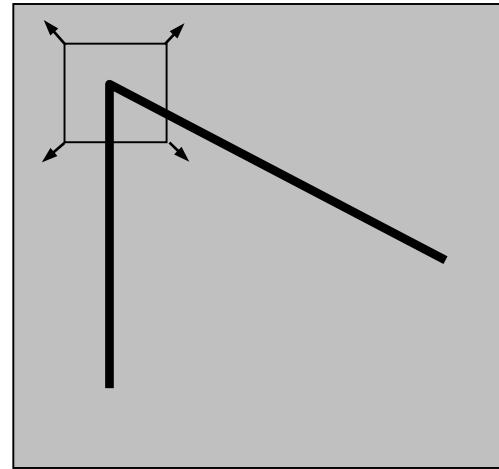
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



"flat":
no change in all
directions



"edge":
no change along the
edge direction

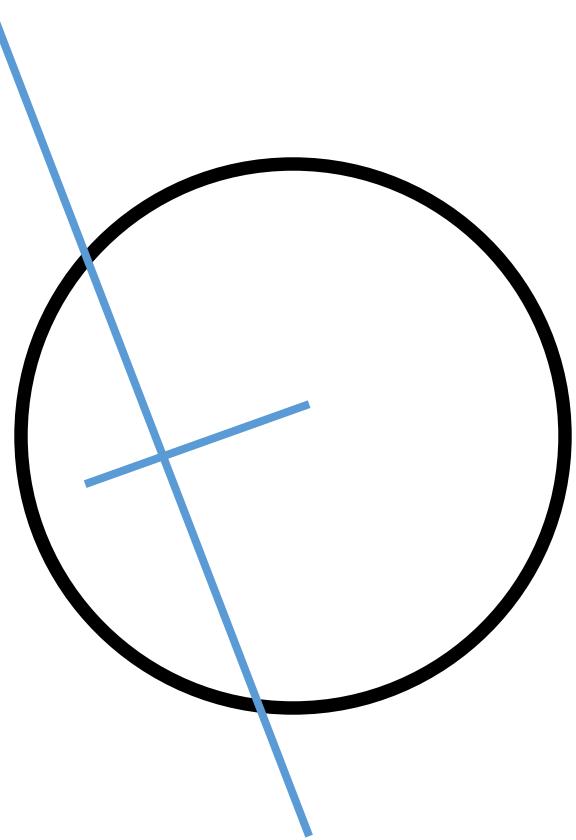


"corner":
significant change in
all directions

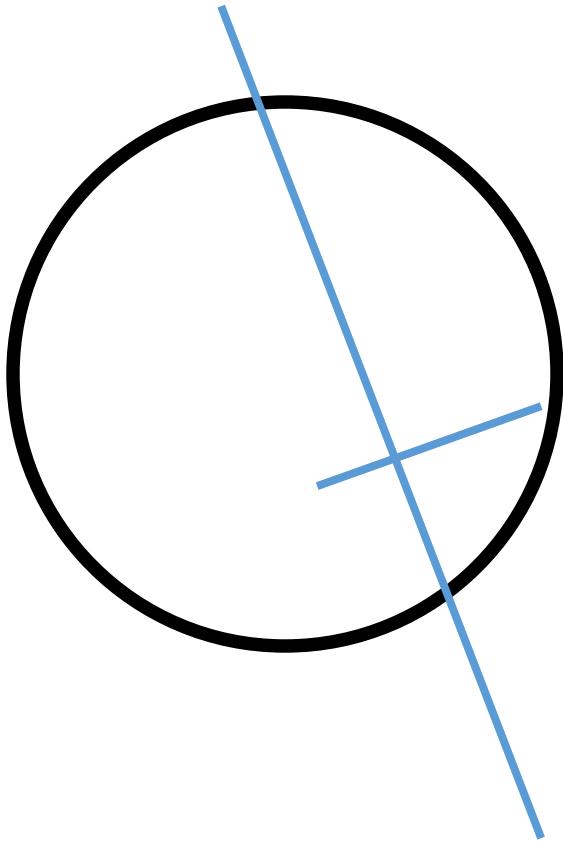
Corners

- 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**

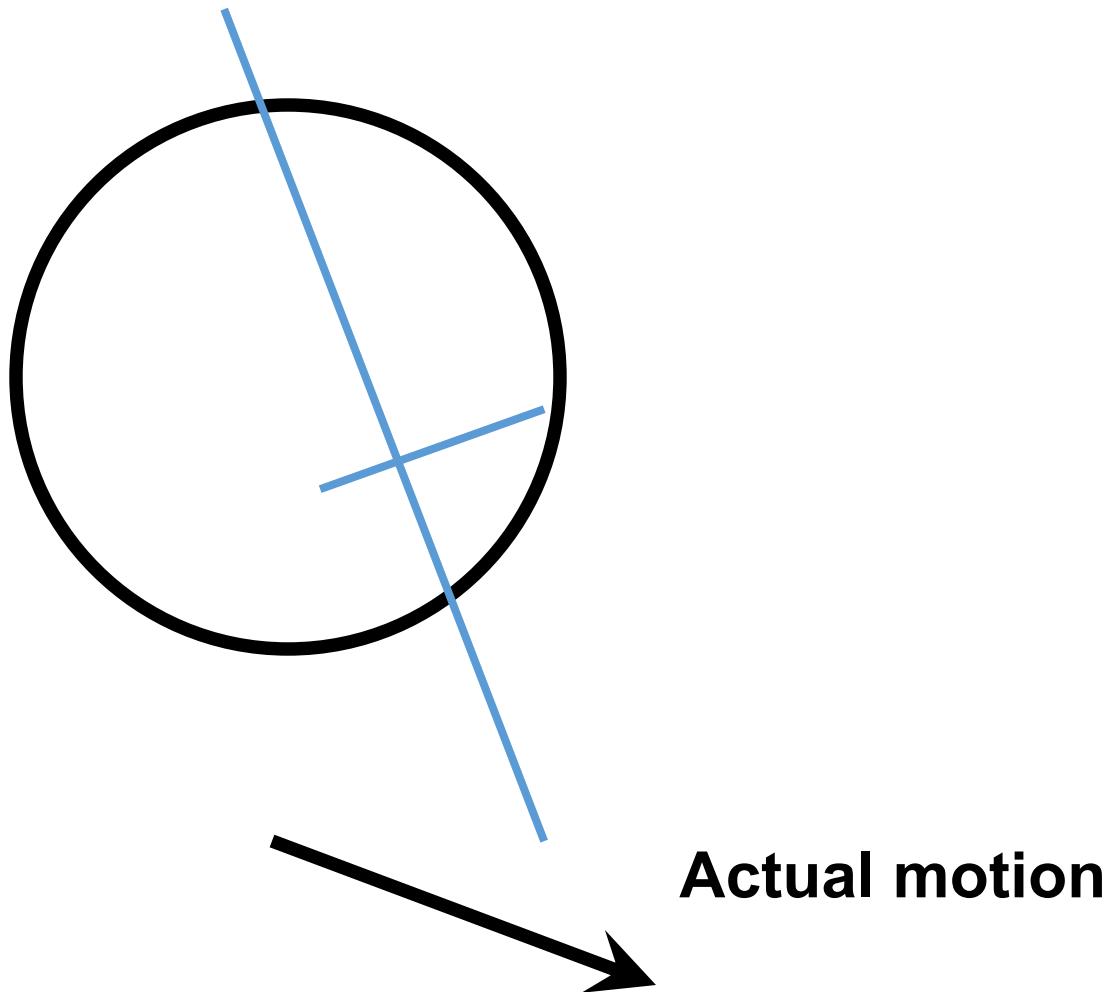
The aperture problem resolved



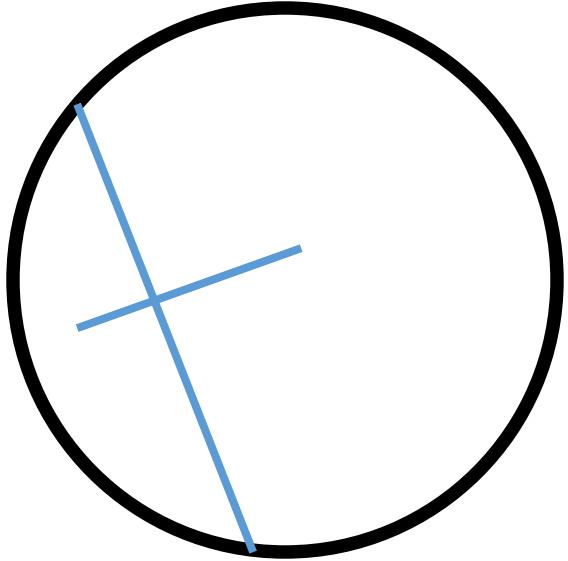
The aperture problem resolved



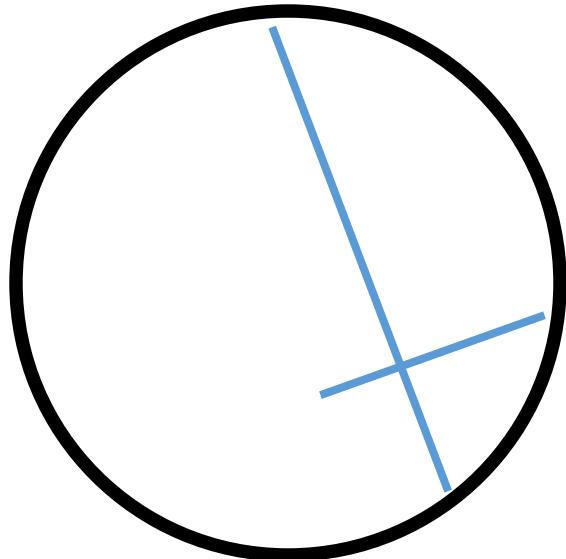
The aperture problem resolved



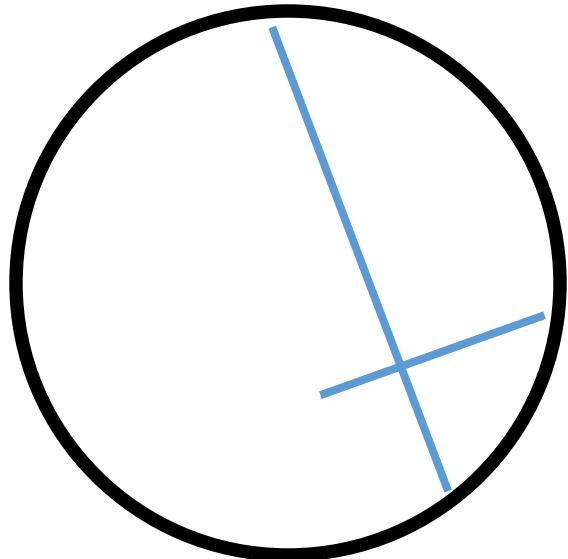
The aperture problem resolved



The aperture problem resolved



The aperture problem resolved

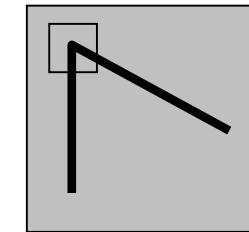
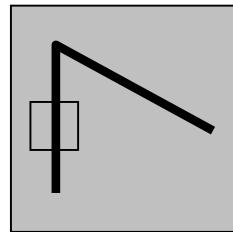
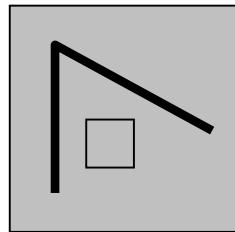


Perceived motion

Harris corner detection

1. Generate a **cornerness score** for each image

window



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

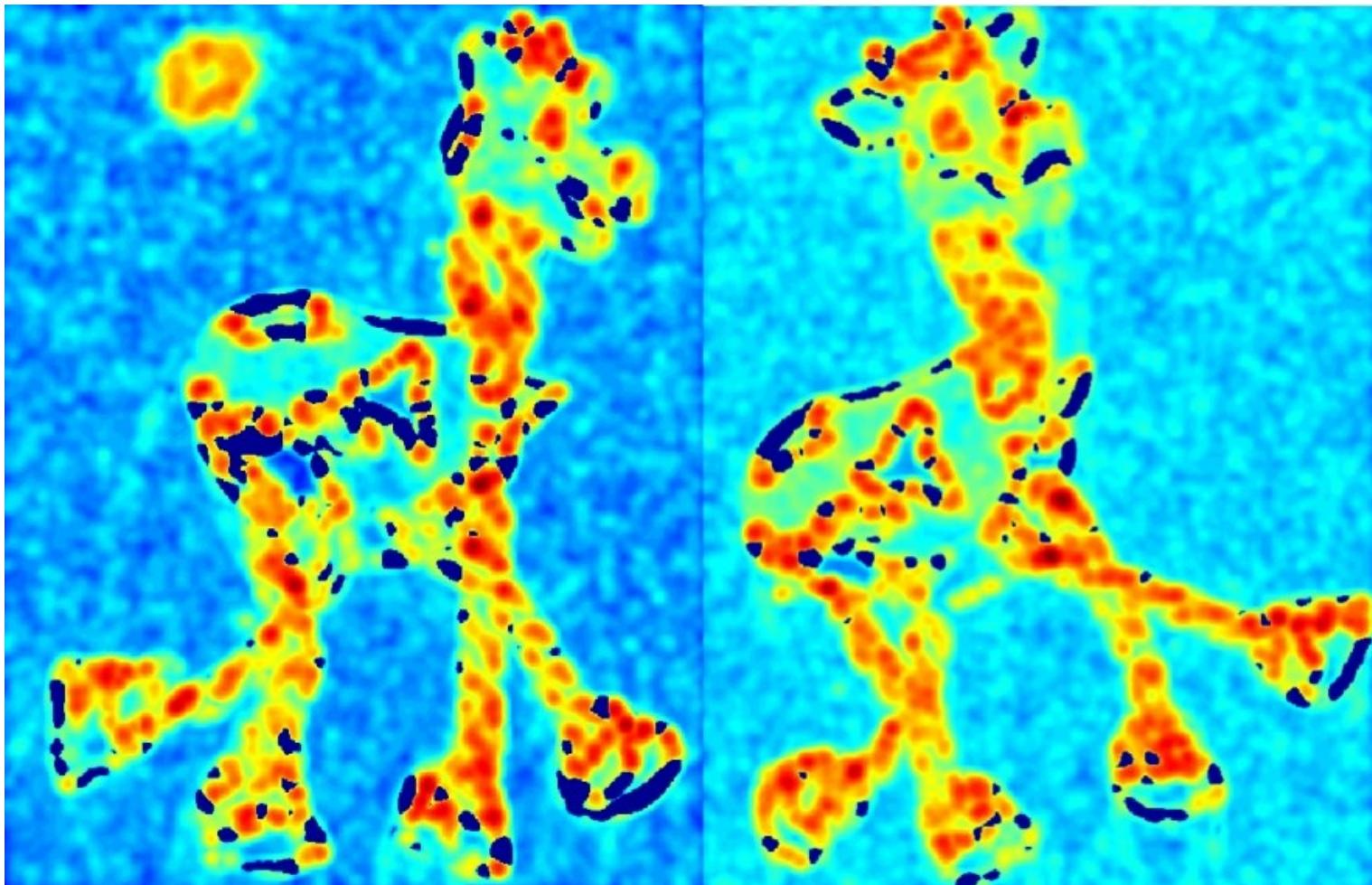
Harris detector steps



Source: James Hays

Harris detector steps

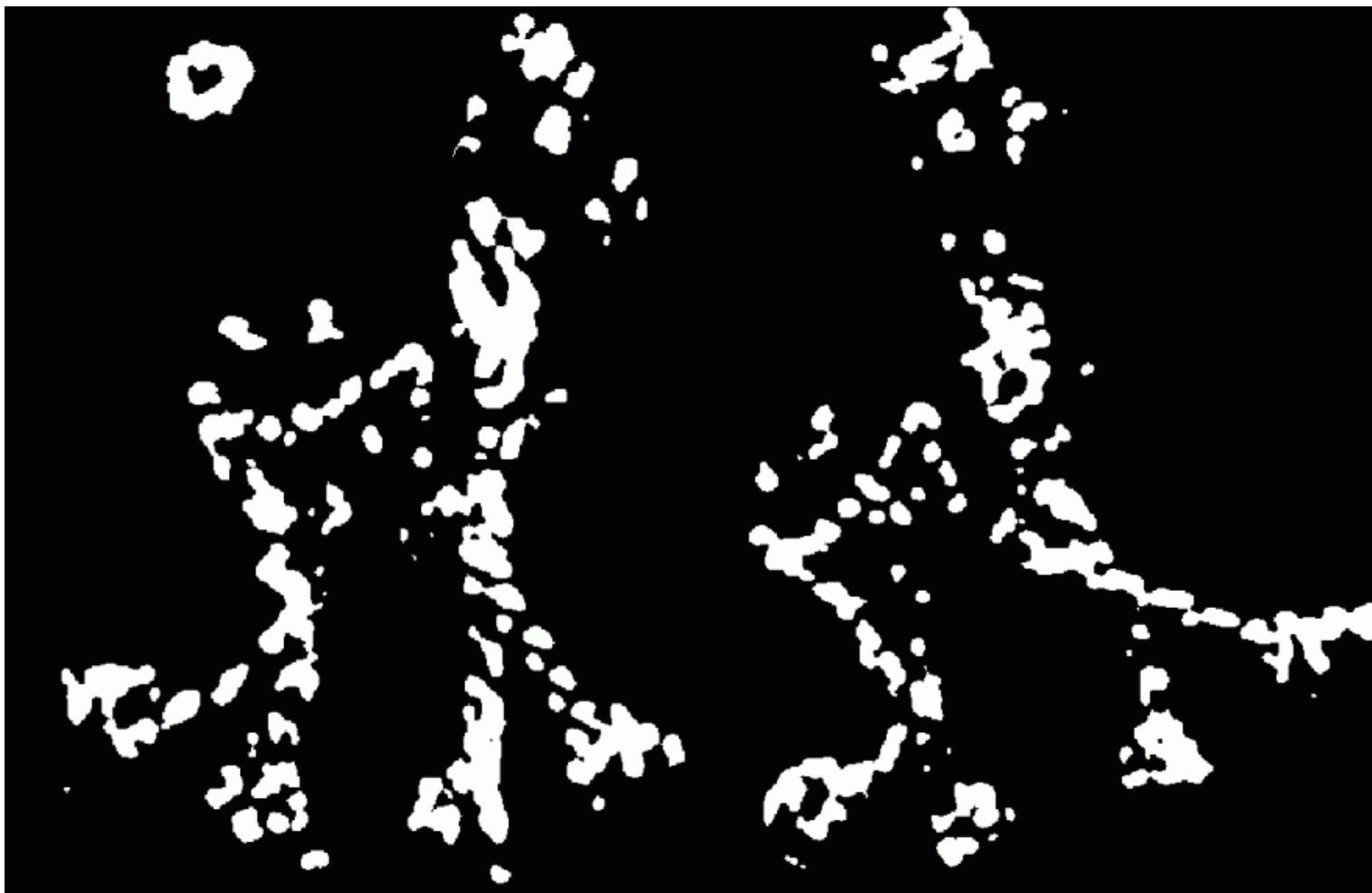
Compute corner response R



Source: James Hays

Harris detector steps

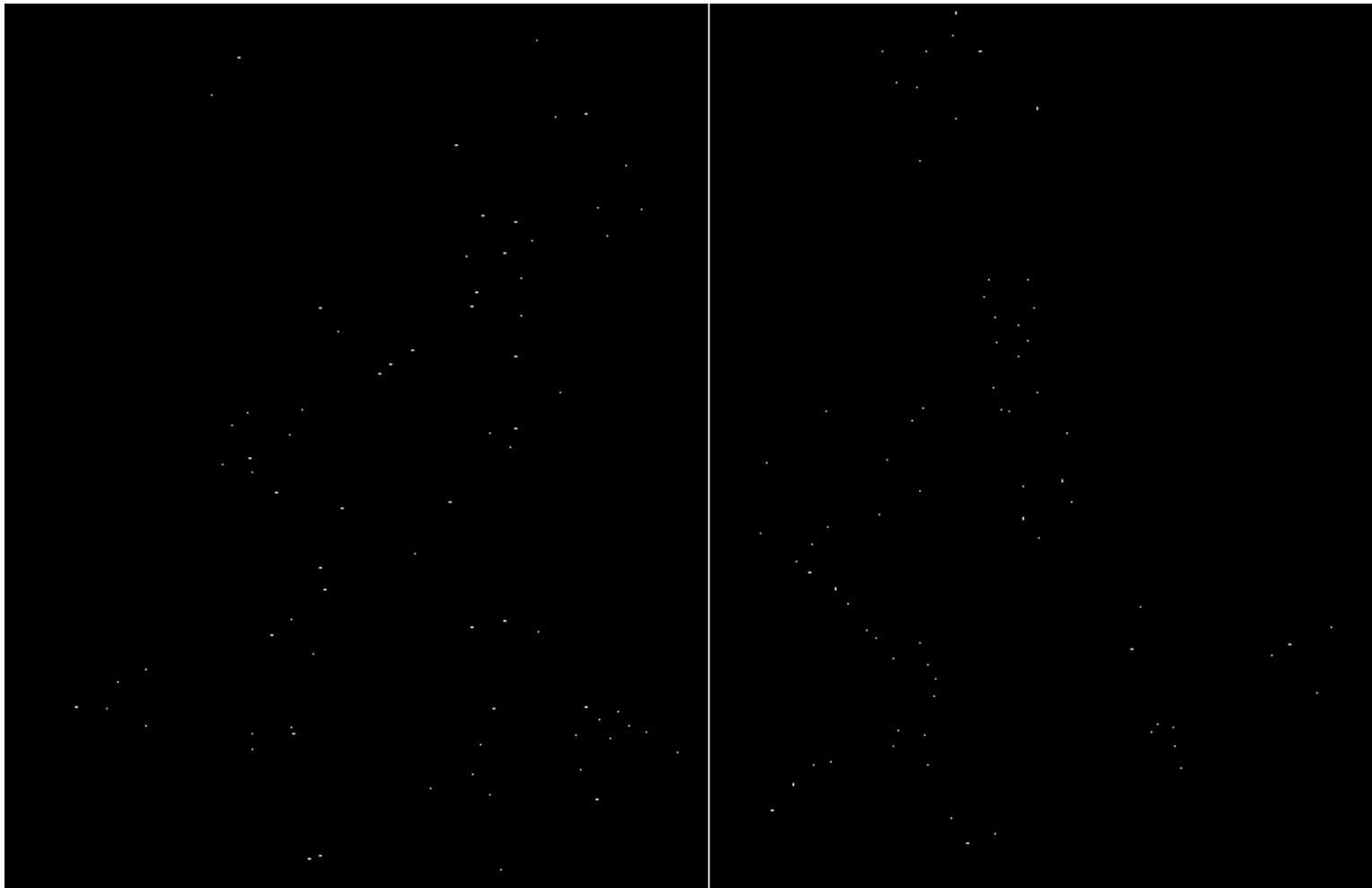
Find points with large corner response: $R > \text{threshold}$



Source: James Hays

Harris detector steps

Take only the points of local maxima of R



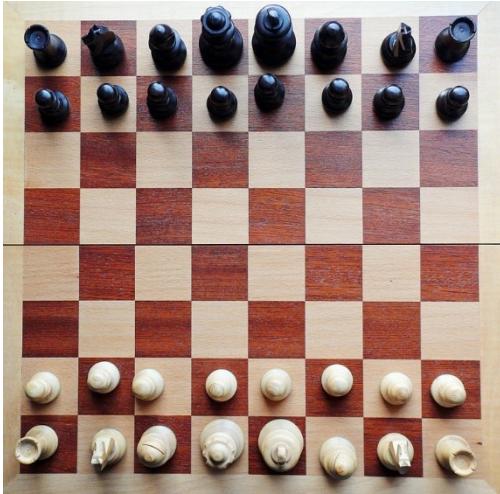
Source: James Hays

Harris detector steps



Source: James Hays

Harris corner detection: esempi



Harris corner detection: esempi

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

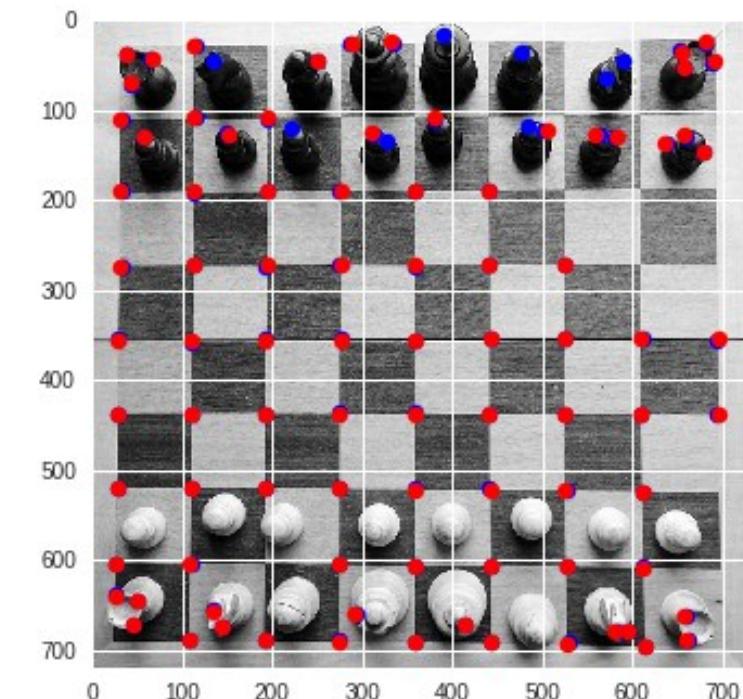
from skimage.io import imread
from skimage.color import rgb2gray
from skimage.feature import corner_harris, corner_subpix, corner_peaks

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

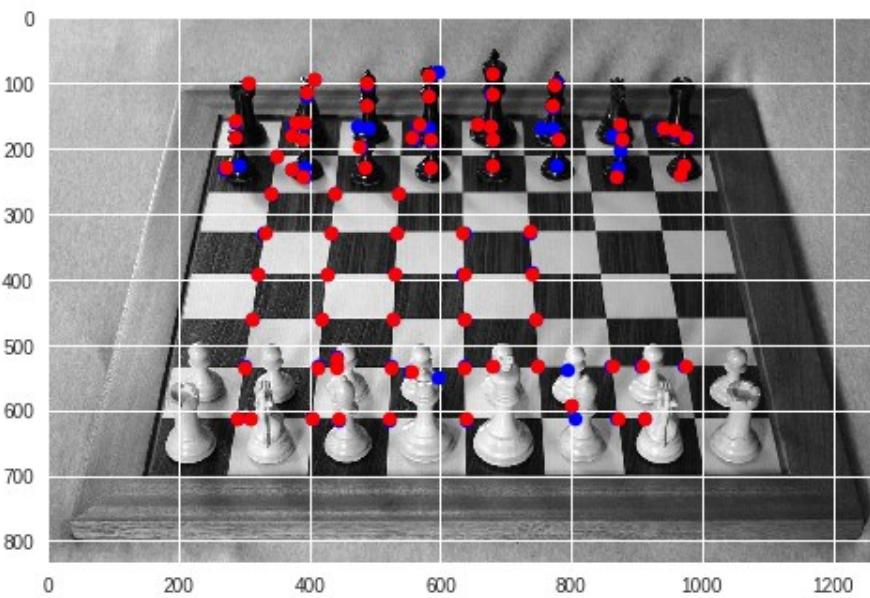
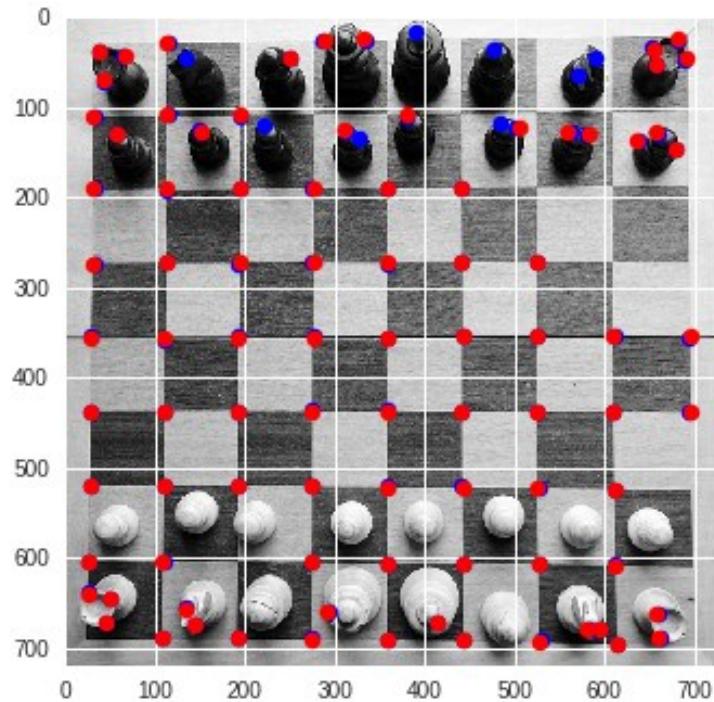
image = imread(urllib.request.urlopen(url))
image = rgb2gray(image)

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

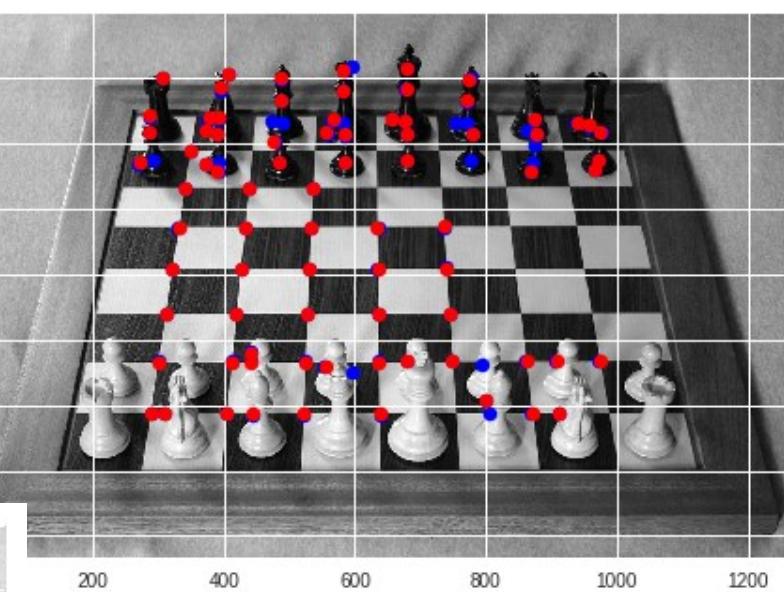
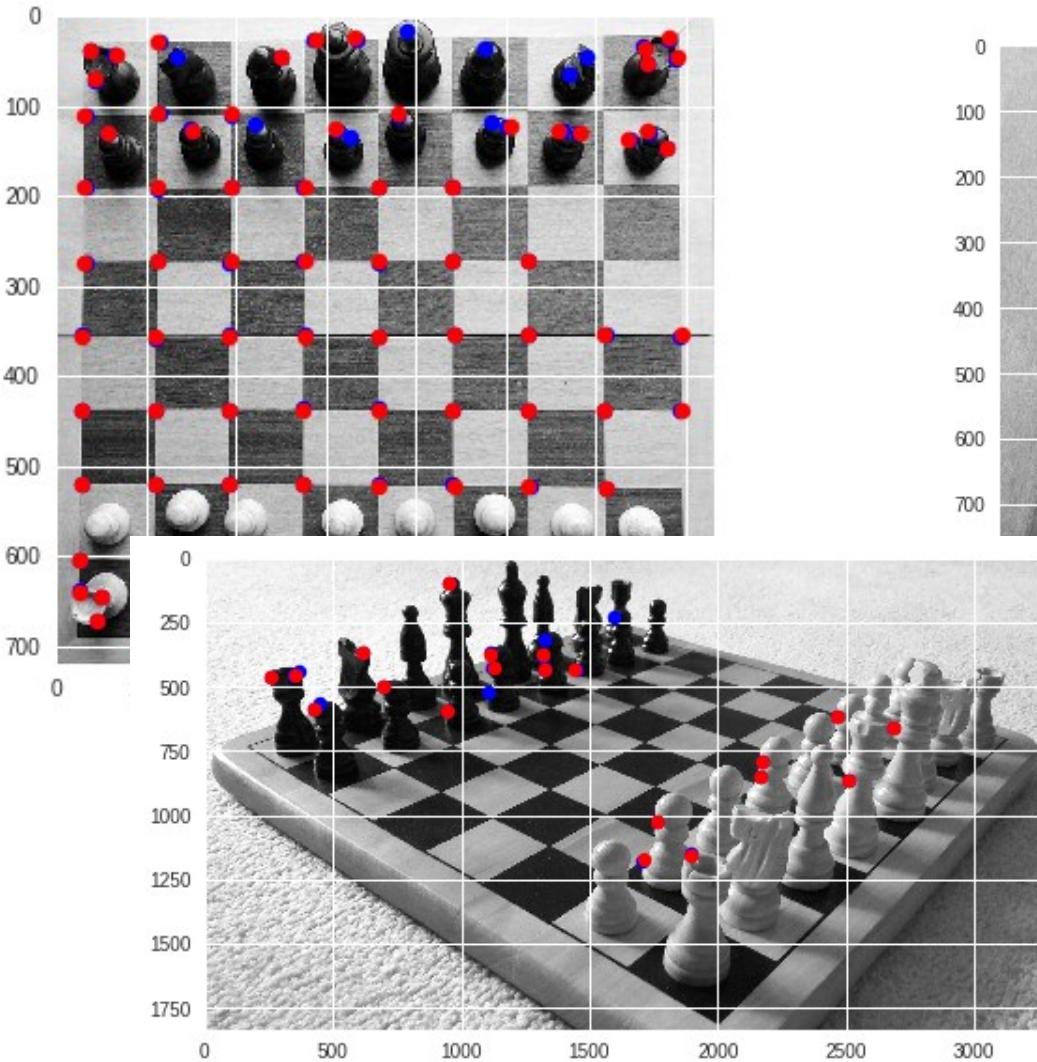
fig, ax = plt.subplots()
ax.imshow(image, interpolation='nearest', cmap=plt.cm.gray)
ax.plot(coords[:, 1], coords[:, 0], '.b', markersize=15)
ax.plot(coords_subpix[:, 1], coords_subpix[:, 0], '.r', markersize=15)
plt.show()
```



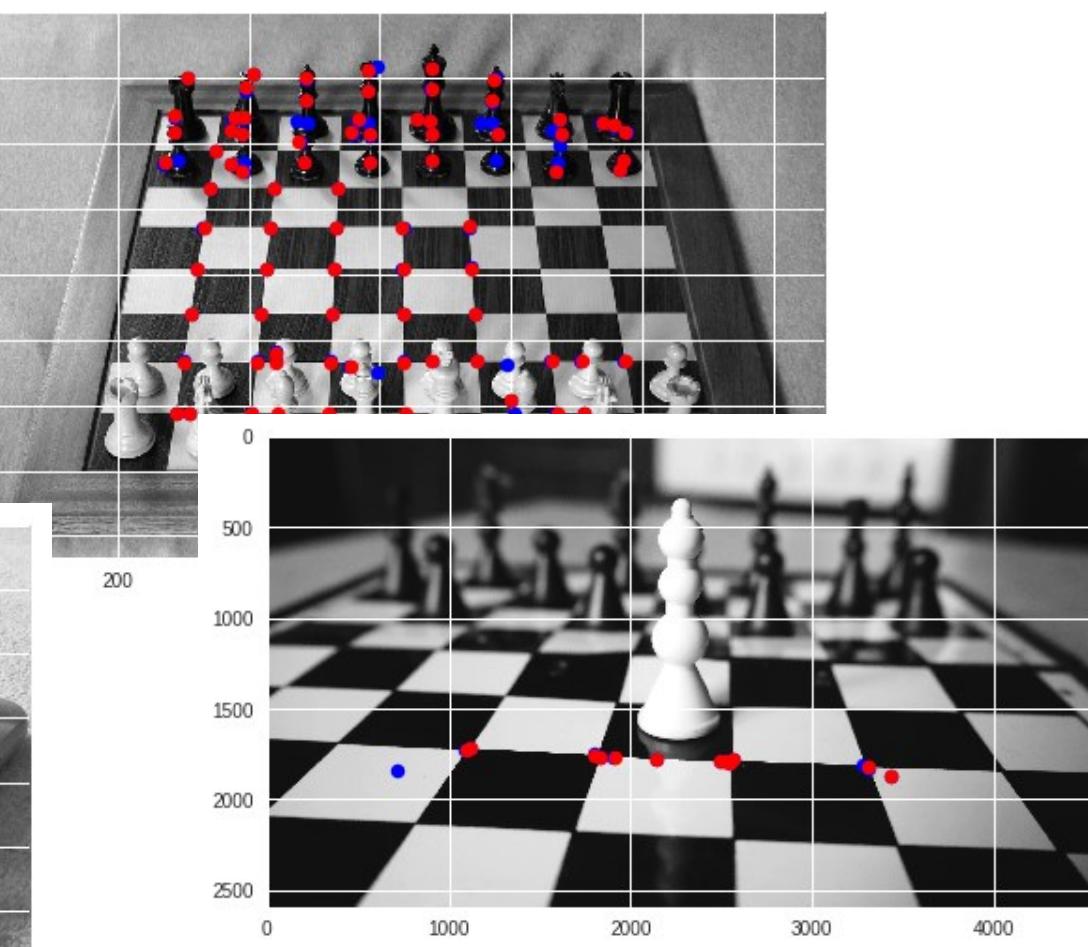
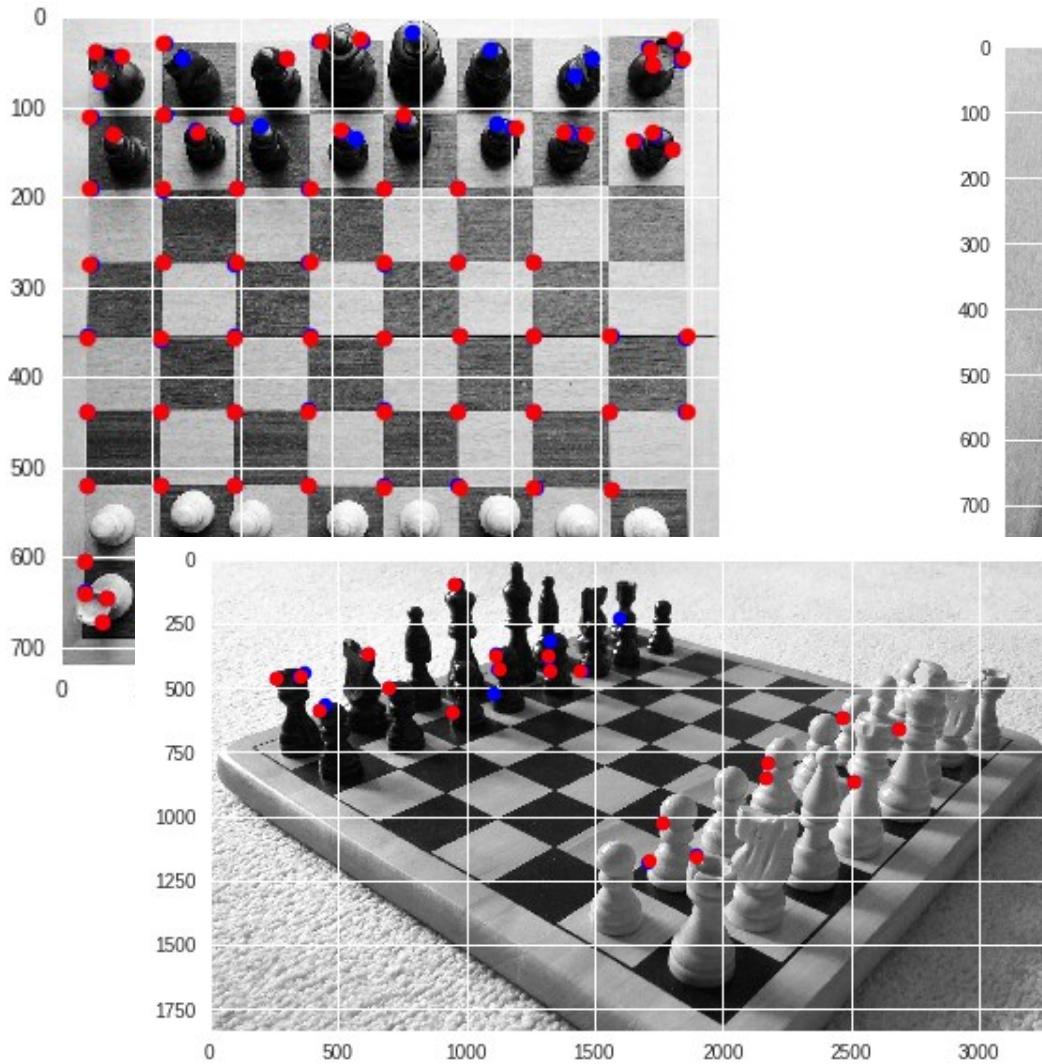
Harris corner detection: esempi



Harris corner detection: esempi



Harris corner detection: esempi



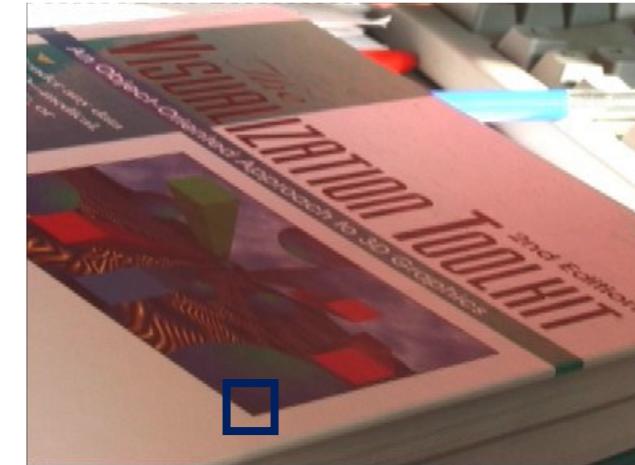
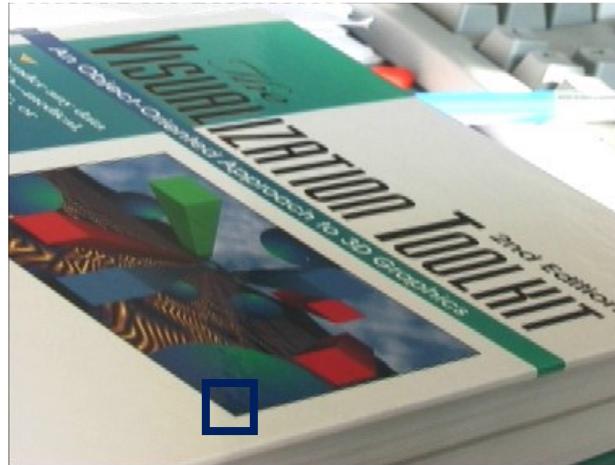
Invariance and covariance of corner locations

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

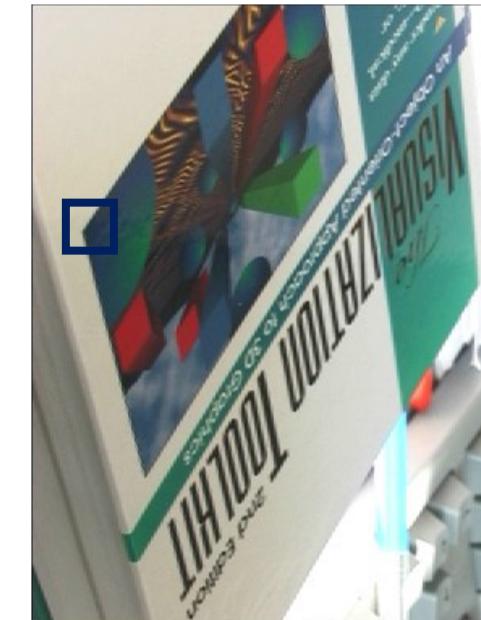
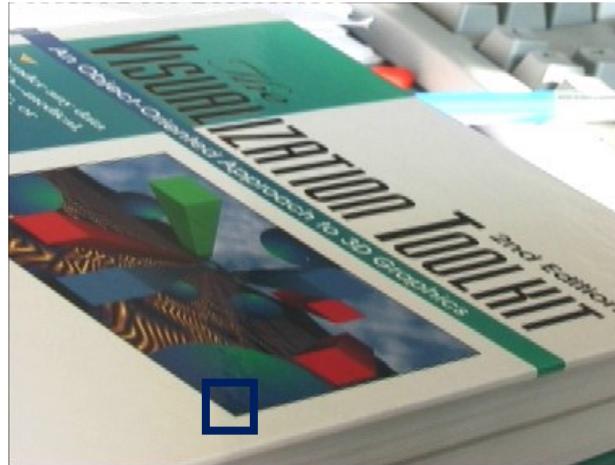
- **Invariance:** image is transformed and corner locations do not change
- **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

Invariance and covariance of corner locations

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

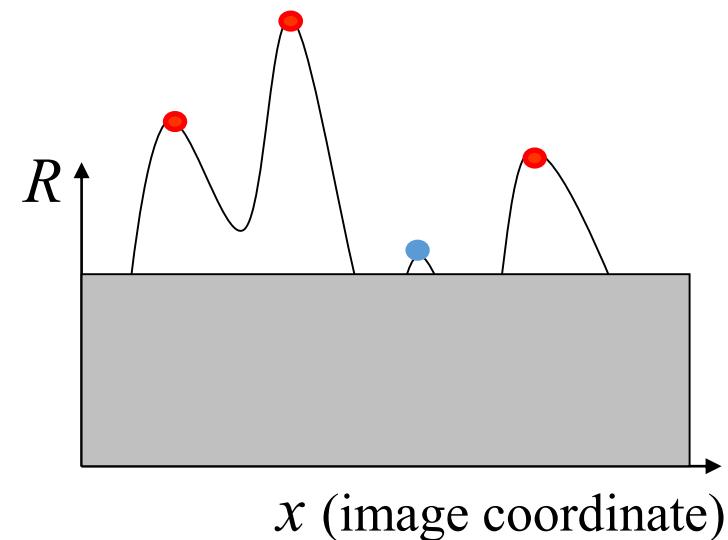
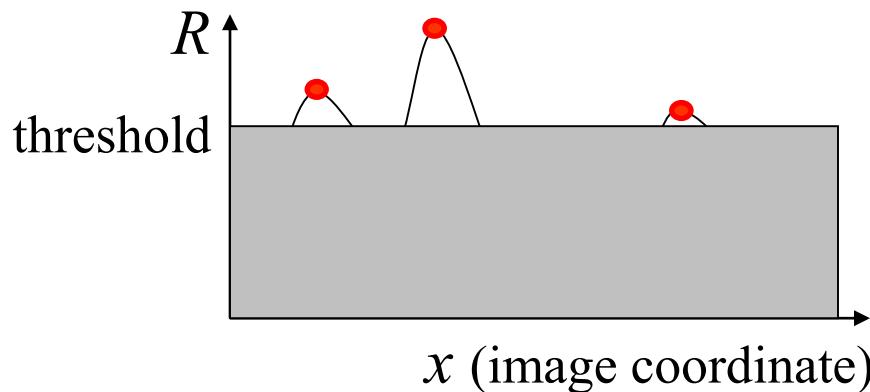


Affine intensity change



$$I \rightarrow a I + b$$

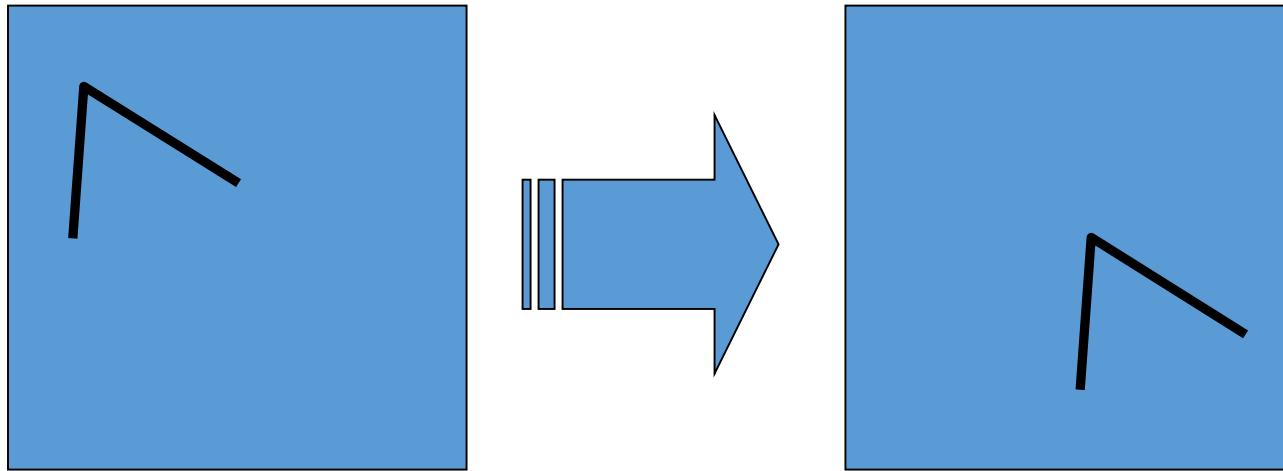
- Only derivatives are used \rightarrow invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a I$



Partially invariant to affine intensity change

Source: James Hays

Translation

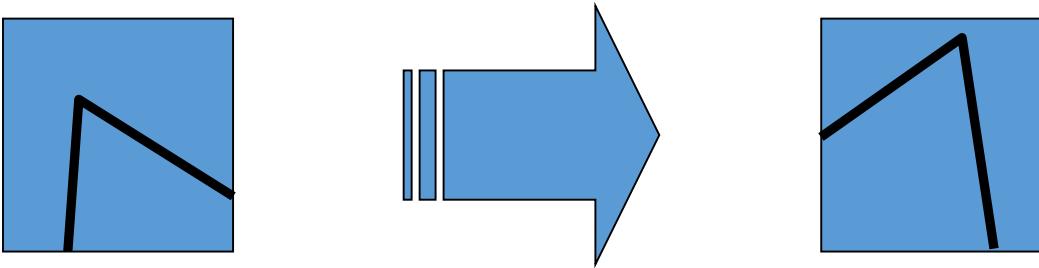


Derivatives and window function are shift-invariant

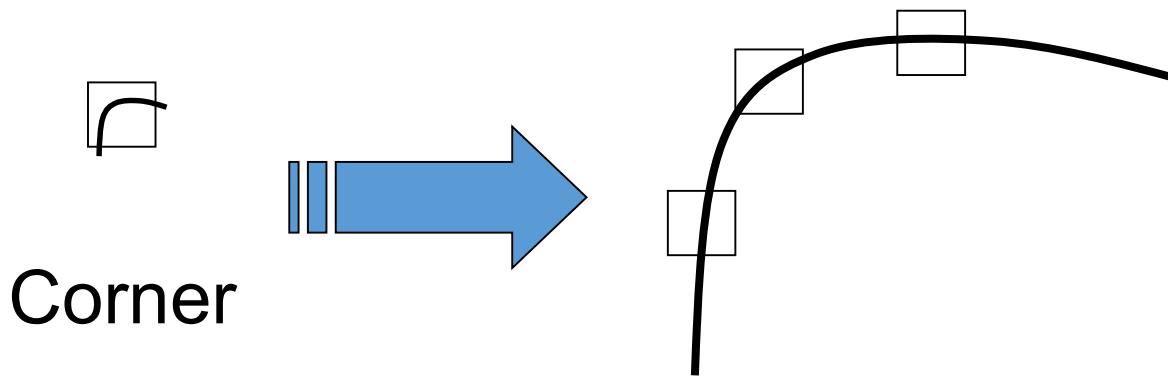
Corner location is covariant w.r.t. translation

Source: James Hays

Rotation and Scaling



Corner location is covariant w.r.t. rotation



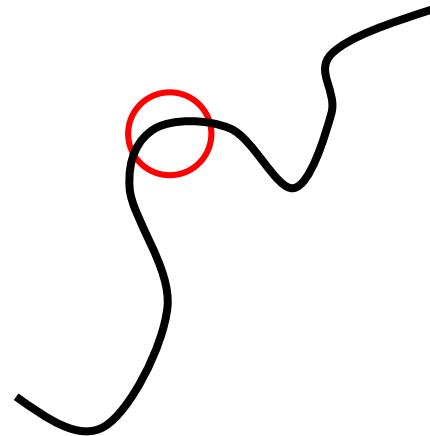
All points will be
classified as
edges

Corner location is not covariant to scaling!

Source: James Hays

Scale invariant detection

Suppose you're looking for corners

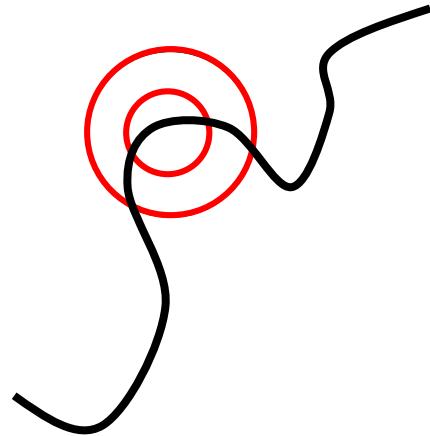


Key idea: find scale that gives local maximum of f

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

Scale invariant detection

Suppose you're looking for corners

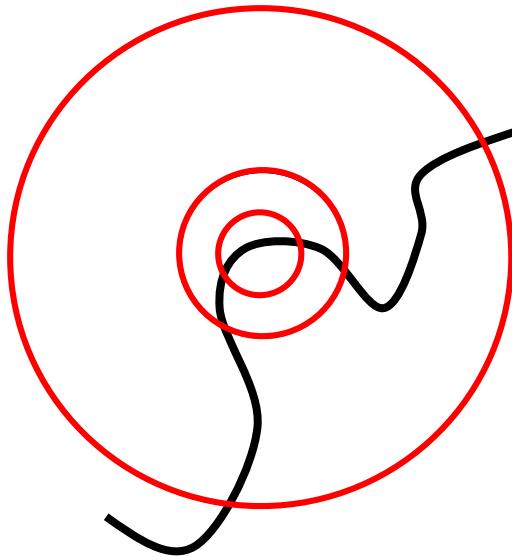


Key idea: find scale that gives local maximum of f

- 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

Scale invariance

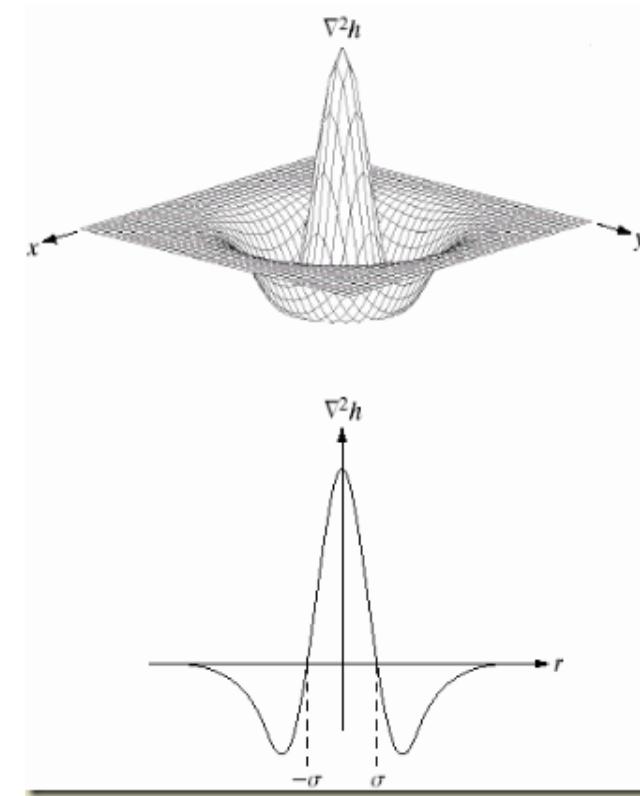
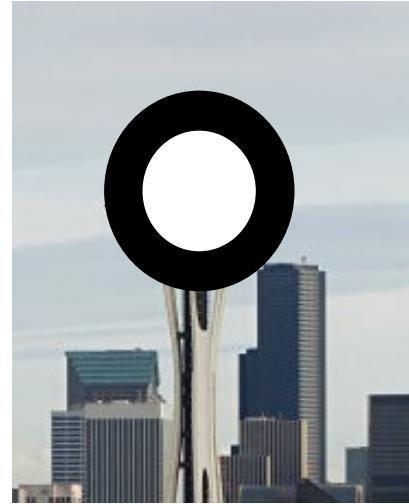
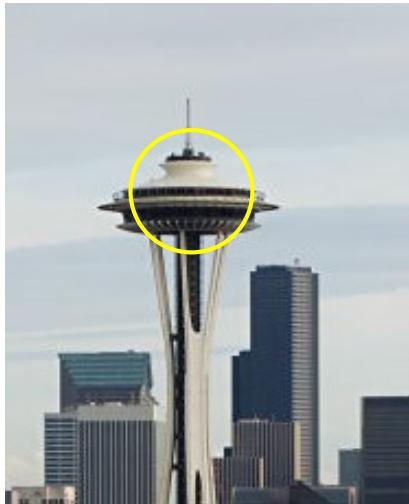


$$f(I_{i_1 \dots i_m}(x, \sigma)) = f(I_{i_1 \dots i_m}(x', \sigma'))$$

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

Differences between inside and outside

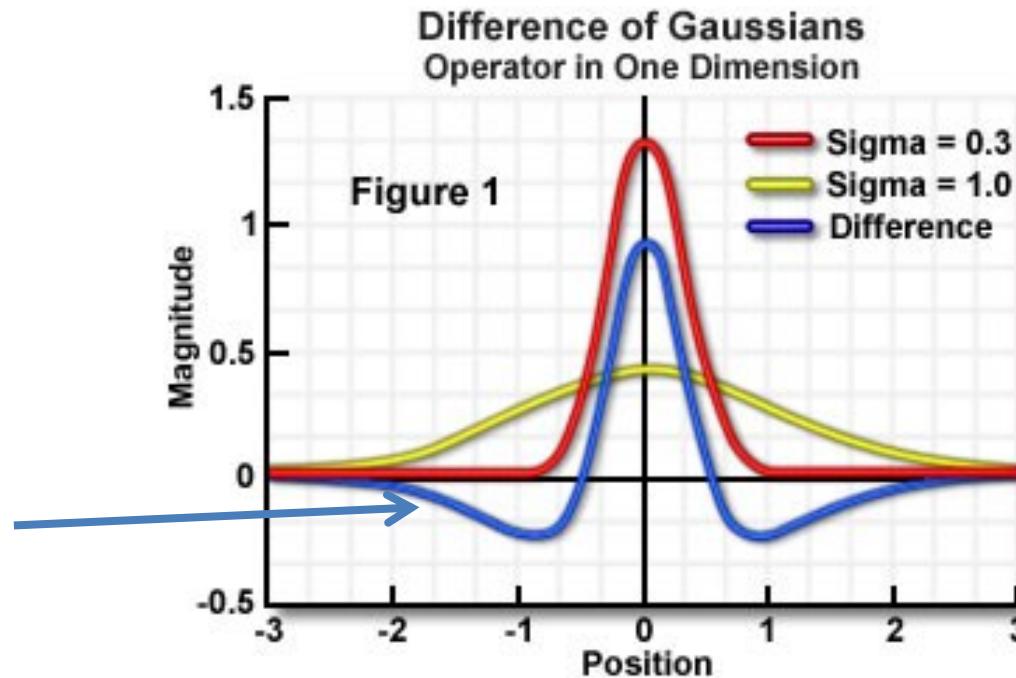
We can use a Laplacian function



Source: Linda Shapiro

Difference-of-Gaussian (DoG)

In practice, the Laplacian is approximated using a Difference of Gaussian (DoG)



Gaussian is invariant to scale change, i.e., $f * \mathcal{G}_\sigma * \mathcal{G}_{\sigma'} = f * \mathcal{G}_{\sigma''}$ and has several other nice properties [Lindeberg, 1994]

Difference-of-Gaussian (DoG)

G1



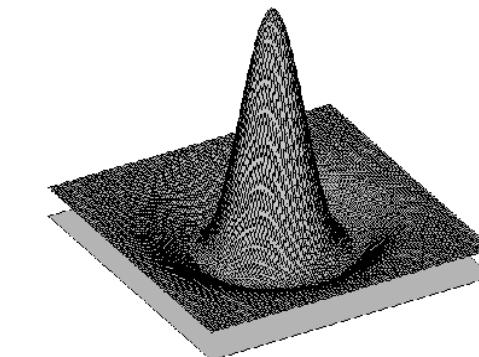
-

G2

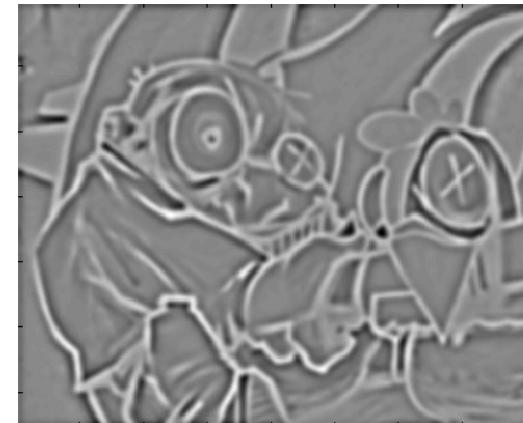


=

DoG

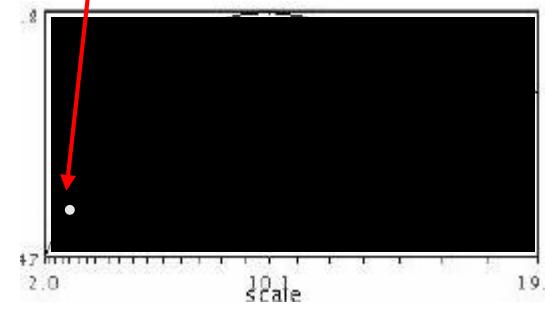
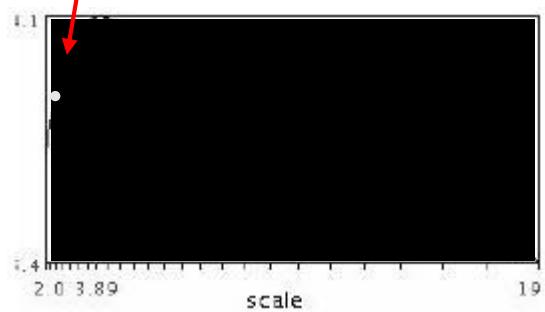


=



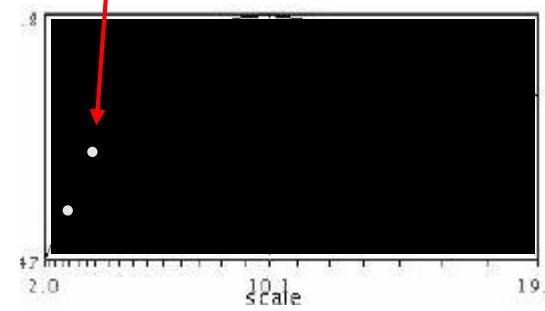
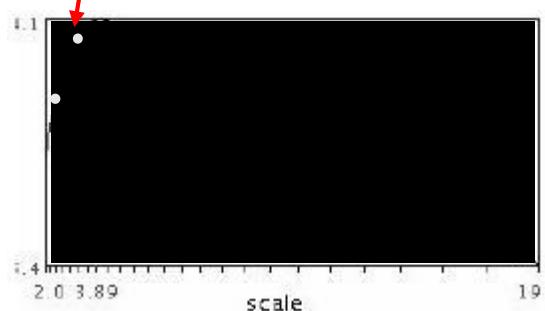
Automatic scale selection

- Function responses for increasing scale (scale signature)



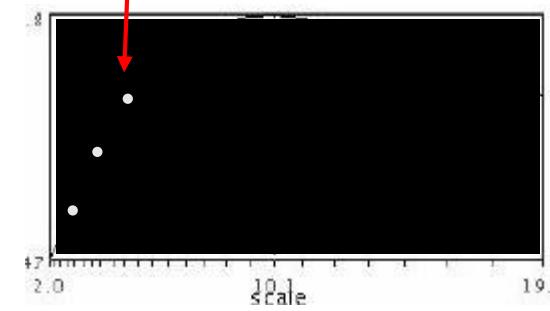
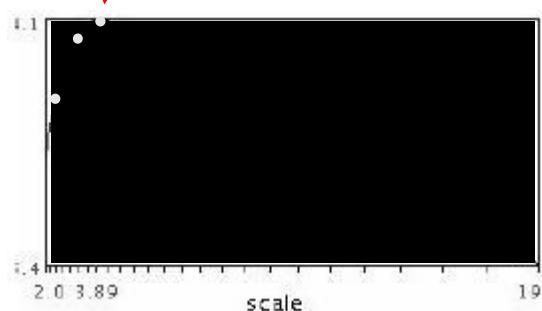
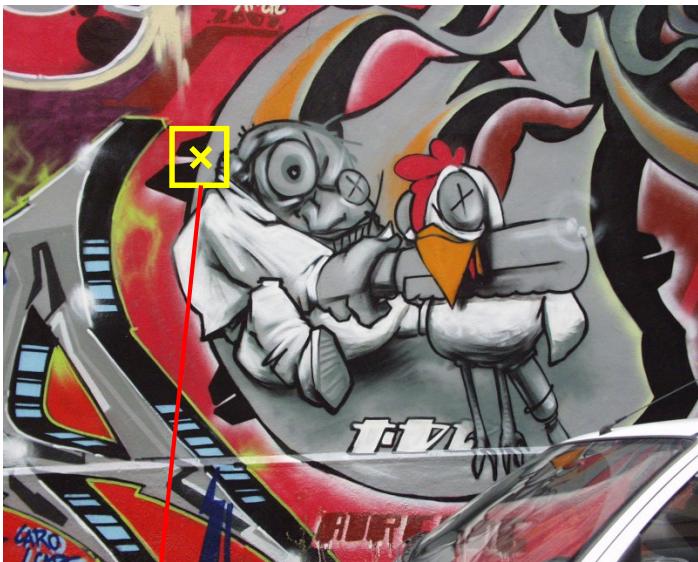
Automatic scale selection

- Function responses for increasing scale (scale signature)



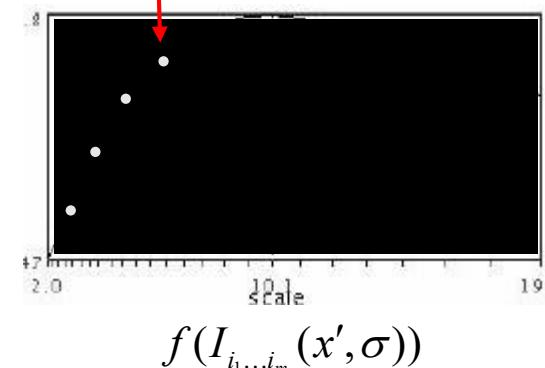
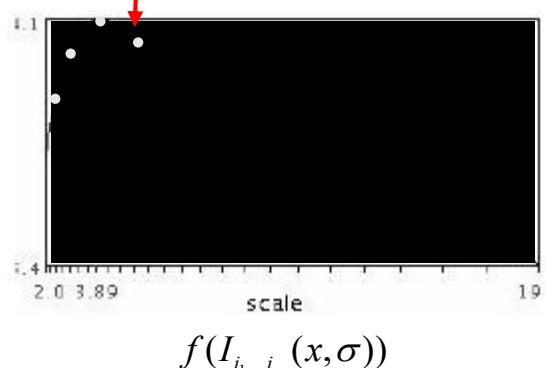
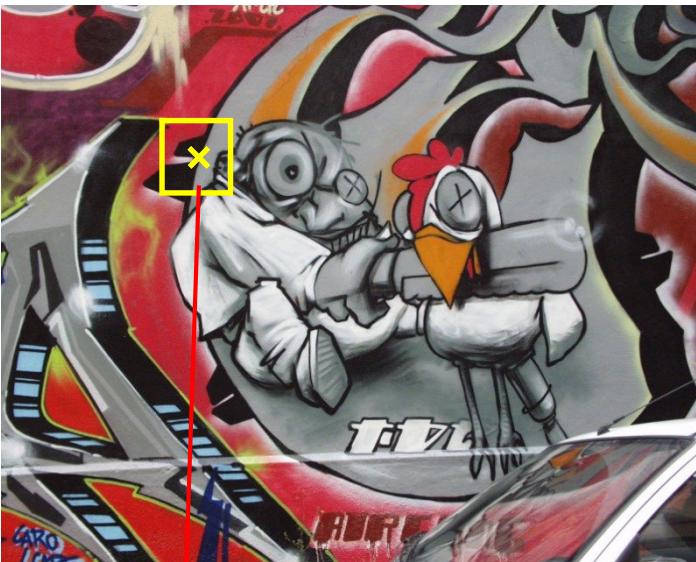
Automatic scale selection

- Function responses for increasing scale (scale signature)



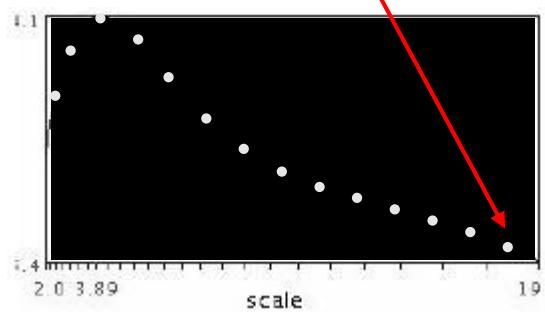
Automatic scale selection

- Function responses for increasing scale (scale signature)

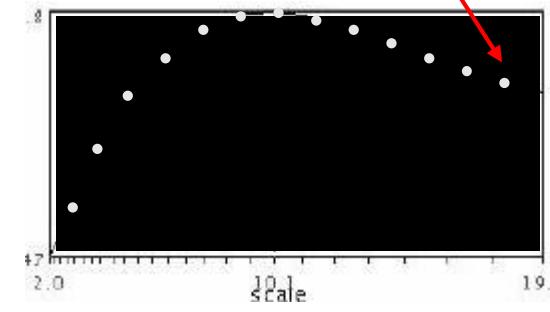


Automatic scale selection

- Function responses for increasing scale (scale signature)



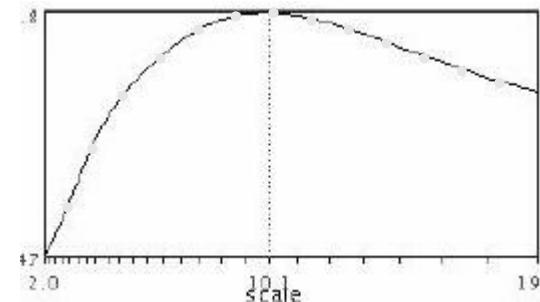
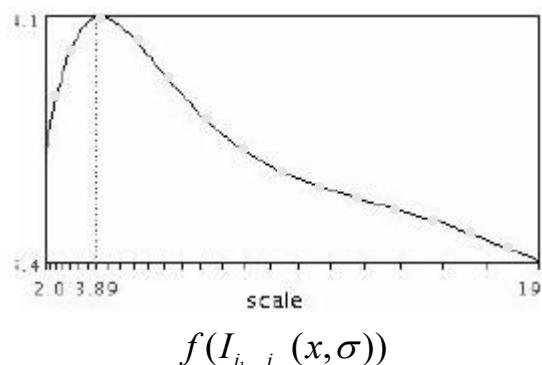
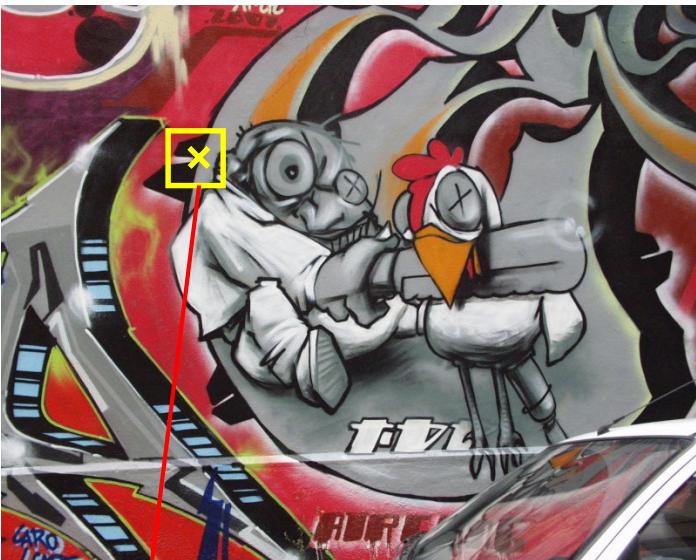
$f(I_{i_1 \dots i_m}(x, \sigma))$



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

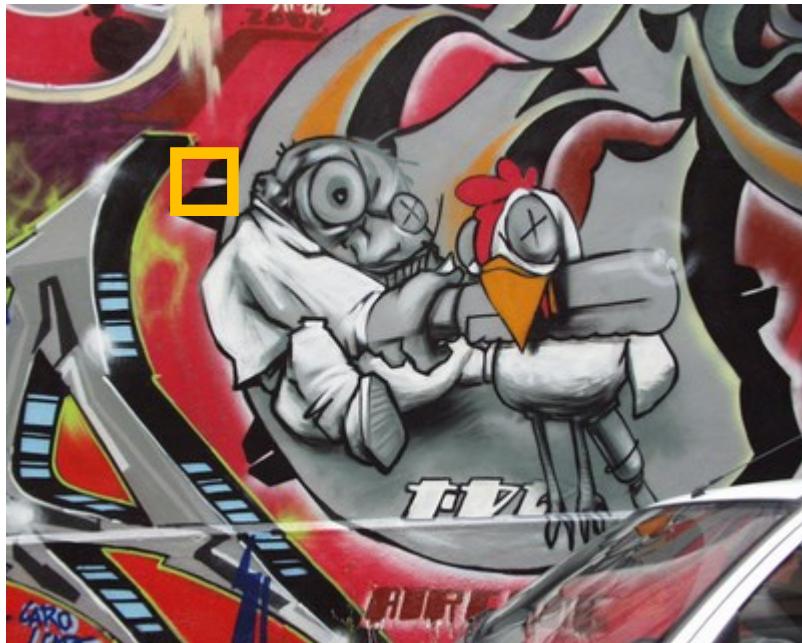
Automatic scale selection

- Function responses for increasing scale (scale signature)



Implementation

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



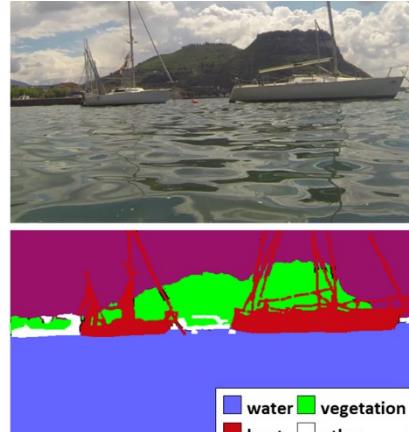
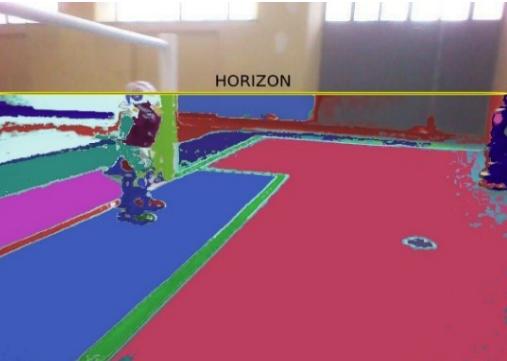
(sometimes need to create in-between
levels, e.g. a $\frac{3}{4}$ -size image)



UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA

Corso di Visione e Percezione

Features



Docente

Domenico D. Bloisi