

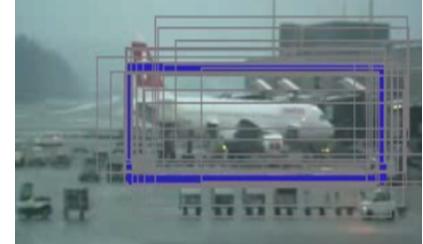
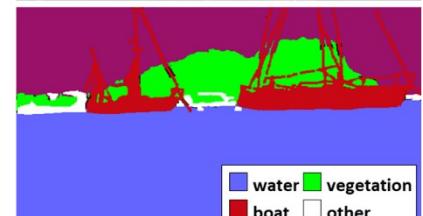
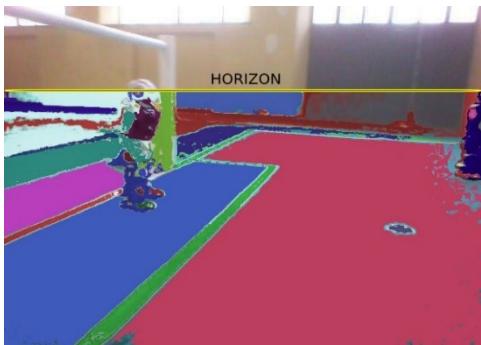
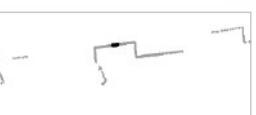


**UNIVERSITÀ DEGLI STUDI  
DELLA BASILICATA**

*Corso di Visione e Percezione*  
A.A. 2019/2020

# Feature Descriptors

Aprile 2020



■ water ■ vegetation  
■ boat ■ other

# Il corso

---

- Home page del corso  
<http://web.unibas.it/bloisi/corsi/visione-e-percezione.html>
- Docente: Domenico Daniele Bloisi
- Periodo: Il semestre marzo 2020 – giugno 2020

Martedì 17:00-19:00 (Aula GUGLIELMINI)

Mercoledì 8:30-10:30 (Aula GUGLIELMINI)

# Riferimenti

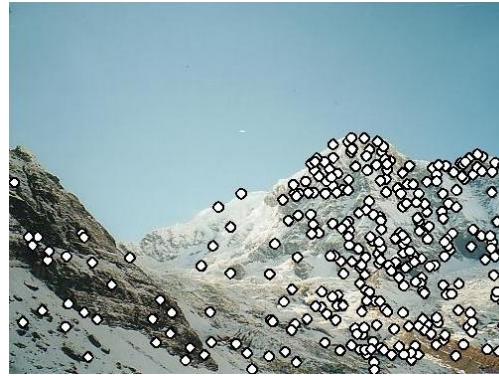
---

- Queste slide sono adattate da  
Noah Snavely - CS5670: Computer Vision  
"Lecture 5: Feature descriptors and matching"
- 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/>

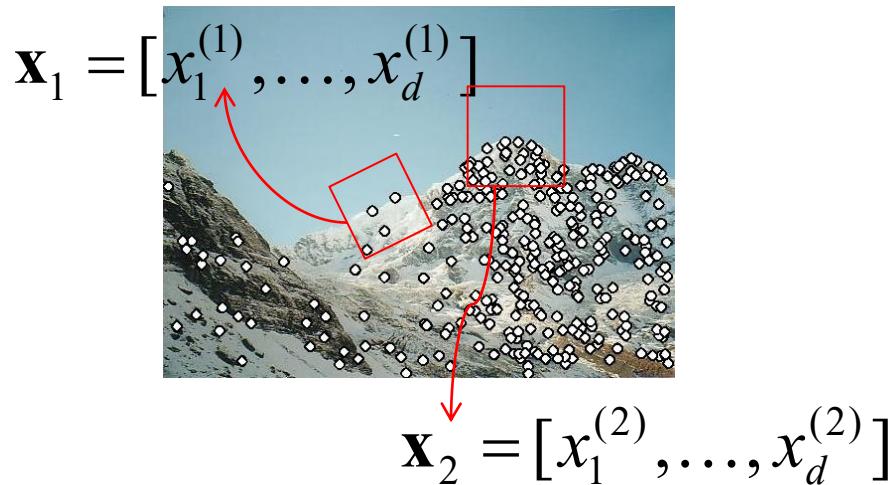
# Local features: main components

---

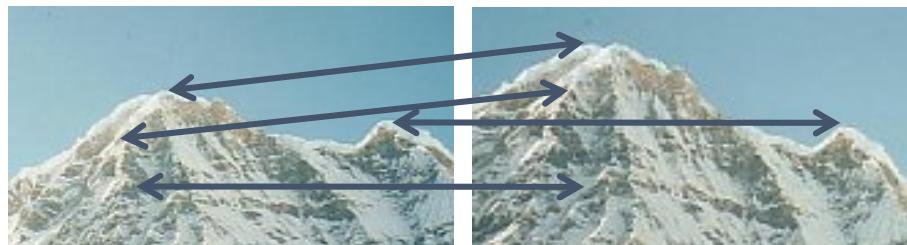
- 1) Detection: Identify the interest points



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



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

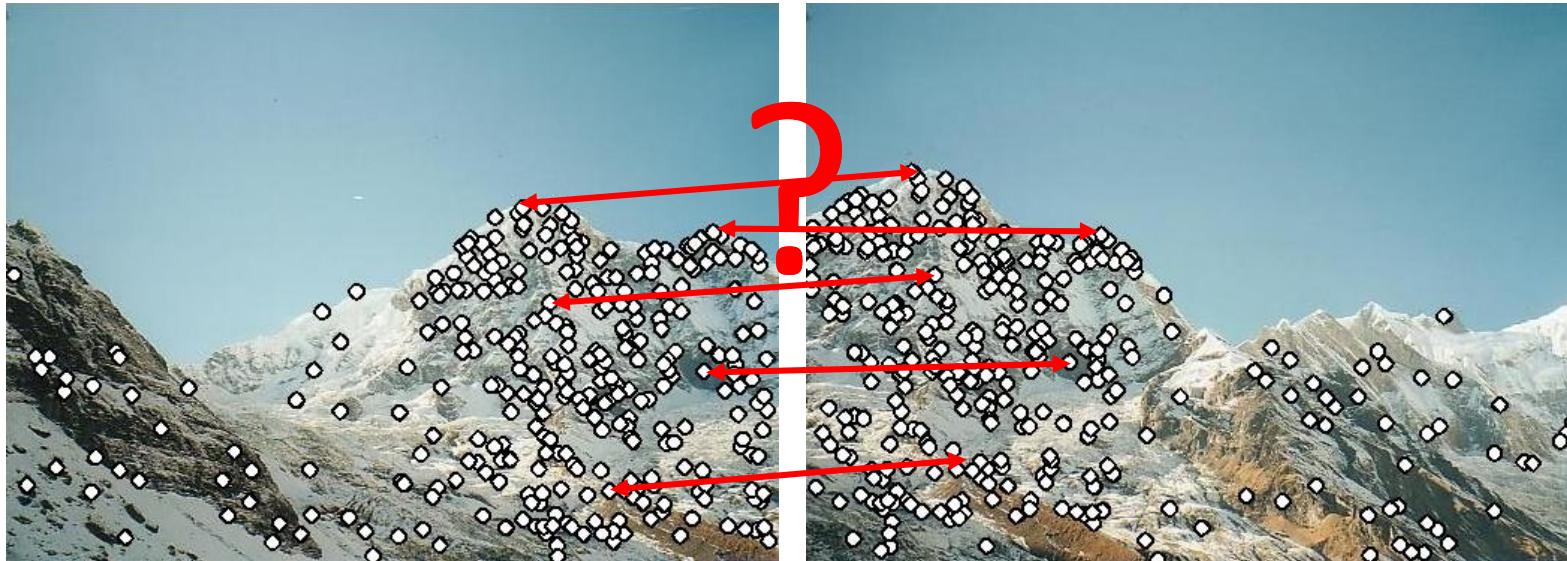


# Feature descriptors

---

We know how to detect good points

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



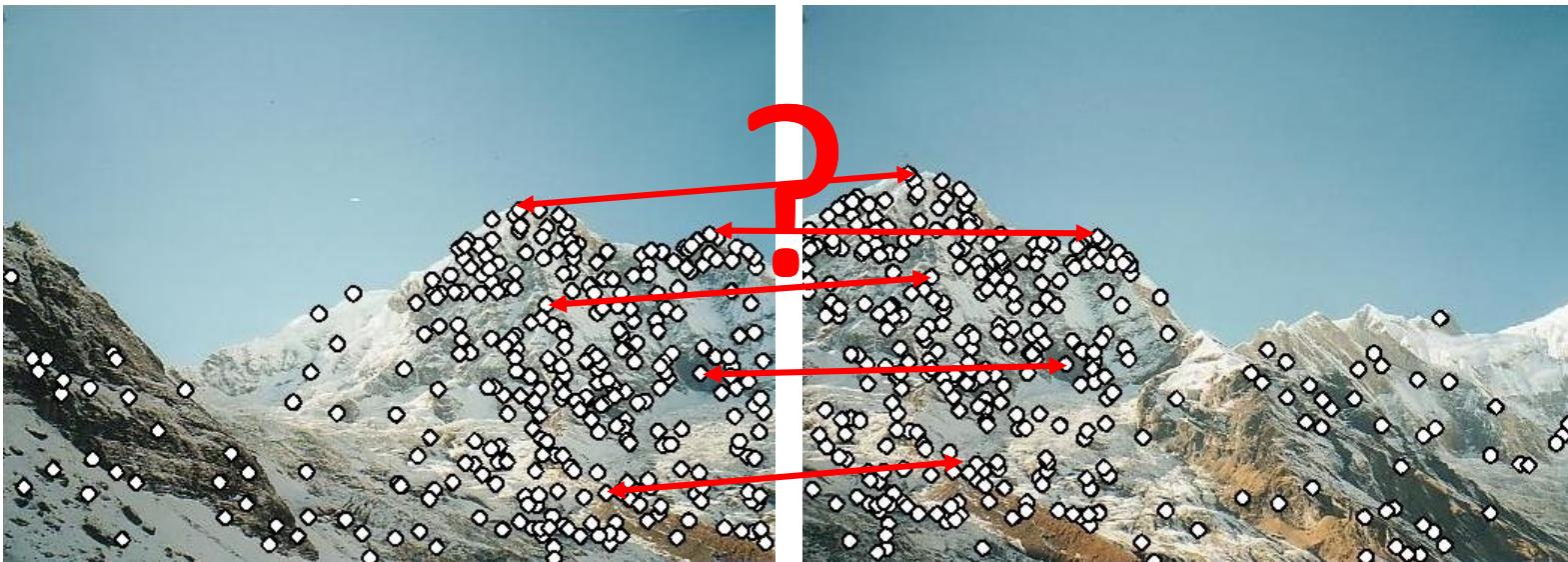
**Answer:** Come up with a *descriptor* for each point,  
find similar descriptors between the two images

# Feature descriptors

---

We know how to detect good points

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



Lots of possibilities

- Simple option: match square windows around the point
- Better option: use invariant and discriminative descriptors (SIFT, SURF, BRIEF, BRISK, ORB)

# Invariance vs. discriminability

---

## Invariance:

Descriptor should **not change** even if image is transformed

## Discriminability:

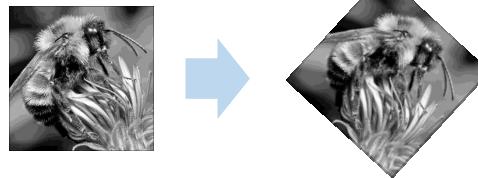
Descriptor should be highly **unique** for each point

# Image transformations

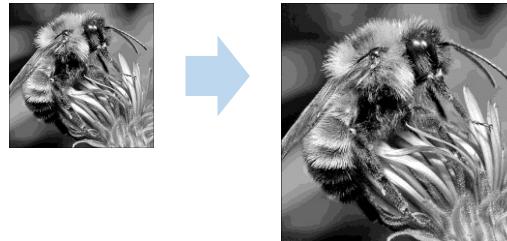
---

Geometric

**Rotation**



**Scale**



Photometric

**Intensity change**



# Invariant descriptors

---

- We looked at invariant / covariant **detectors**
- Most feature **descriptors** are also 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 transforms (some are fully affine invariant)
  - Limited illumination/contrast changes

# Rotation invariance for feature descriptors

---

- Find dominant orientation of the image patch
- Rotate the patch according to this angle



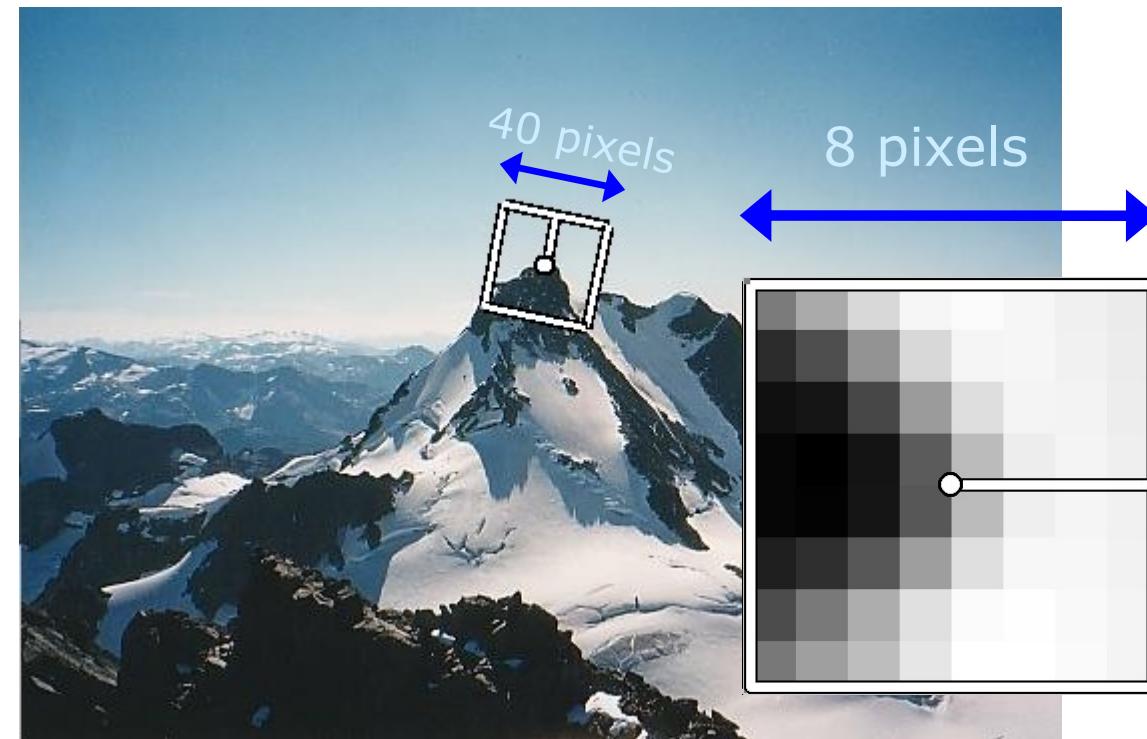
Figure by Matthew Brown

# Multiscale Oriented PatcheS descriptor

---

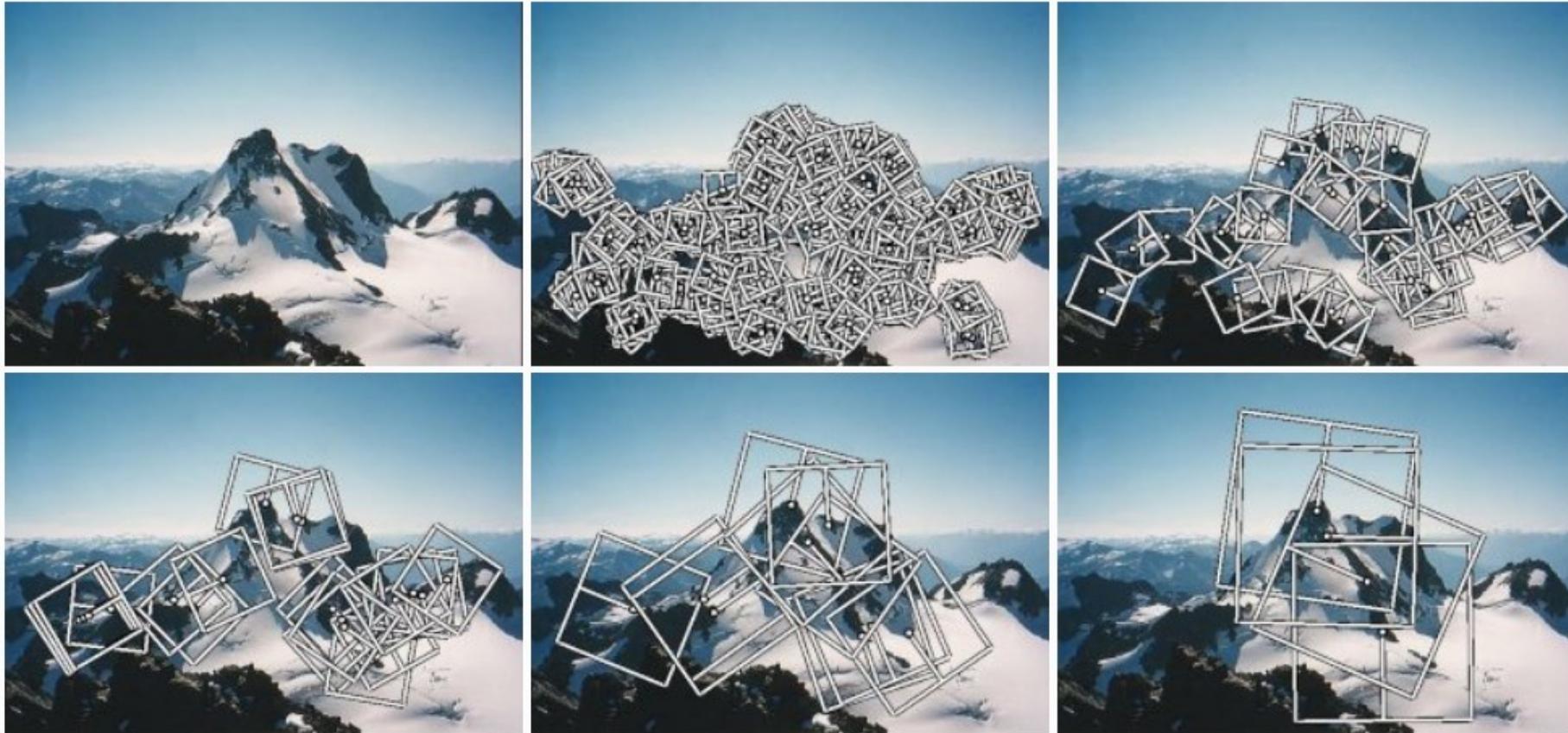
Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- 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 (to obtain bias/gain normalised intensity values)



# Detection at multiple scales

---



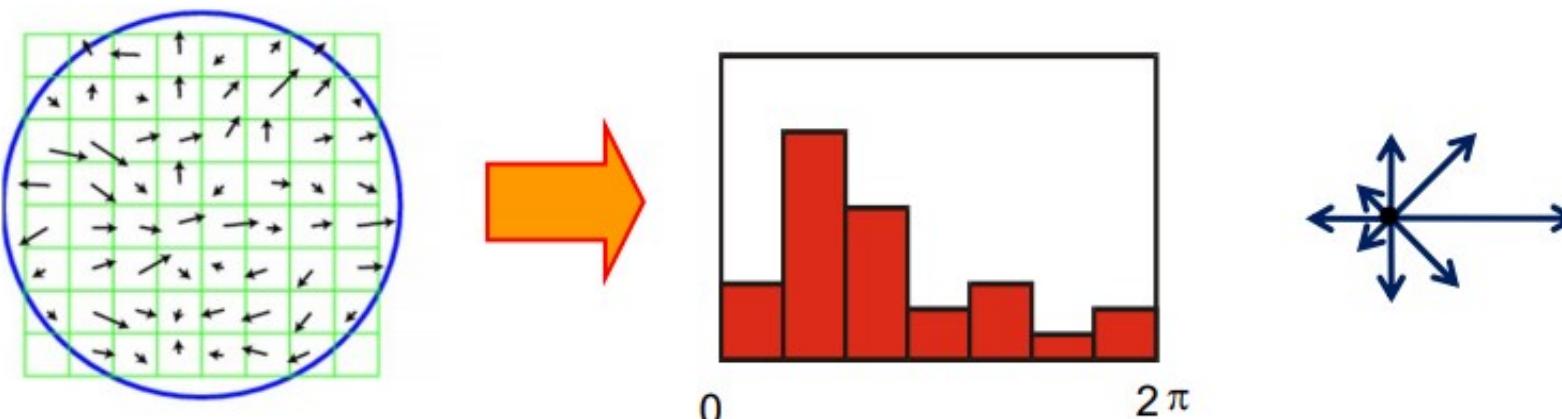
*Multi-scale Oriented Patches (MOPS) extracted at 5 pyramid levels*

# Histograms as descriptors

- Disadvantage of patches as descriptors:
  - Small shifts can affect matching score a lot



- Solution: histograms

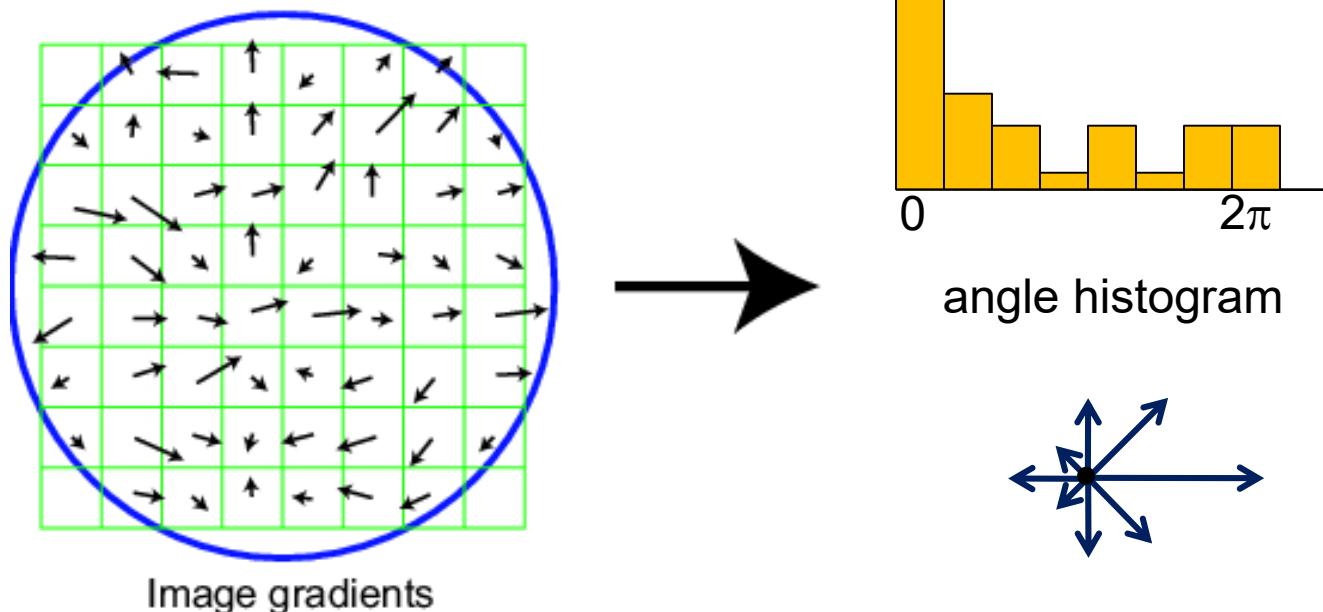


Source: Fei-Fei Li

# Scale Invariant Feature Transform

Basic idea:

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

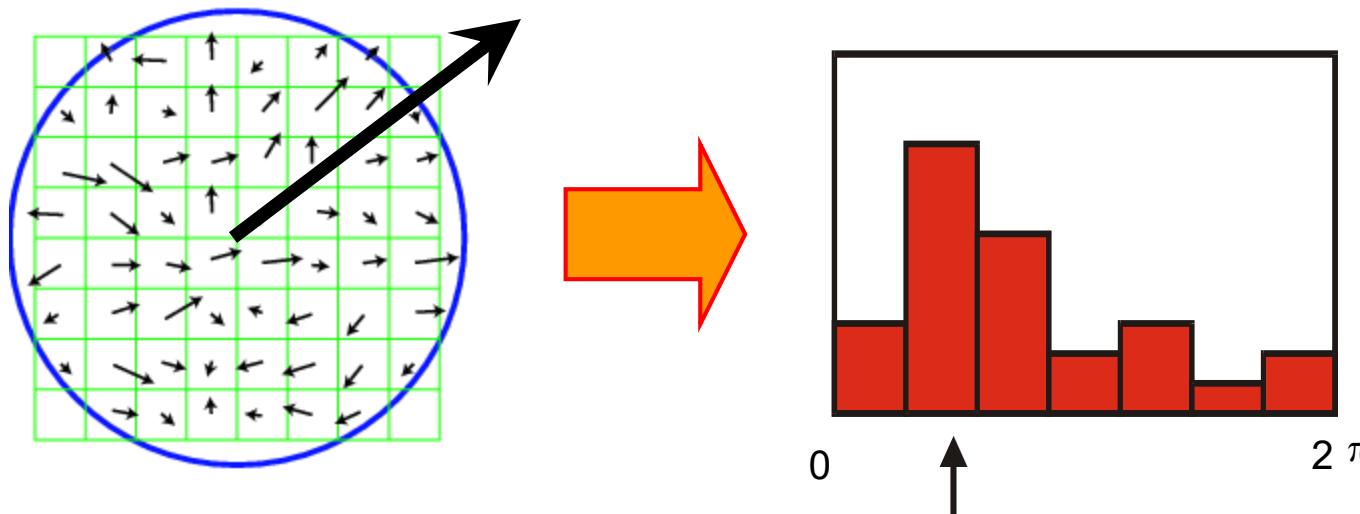


Adapted from slide by David Lowe

# Finding a reference orientation

---

Assign reference orientation at  
peak of smoothed histogram



# SIFT descriptor

---

Full version

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

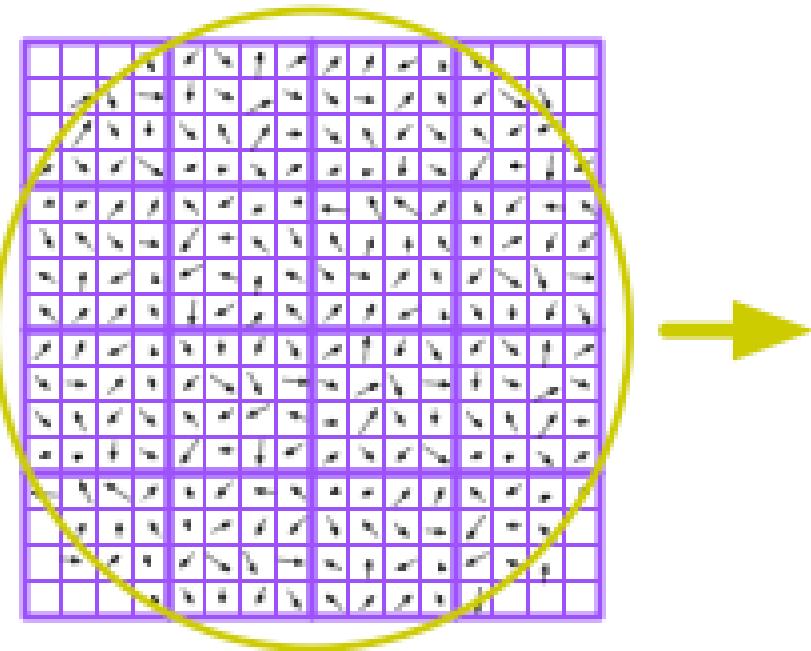
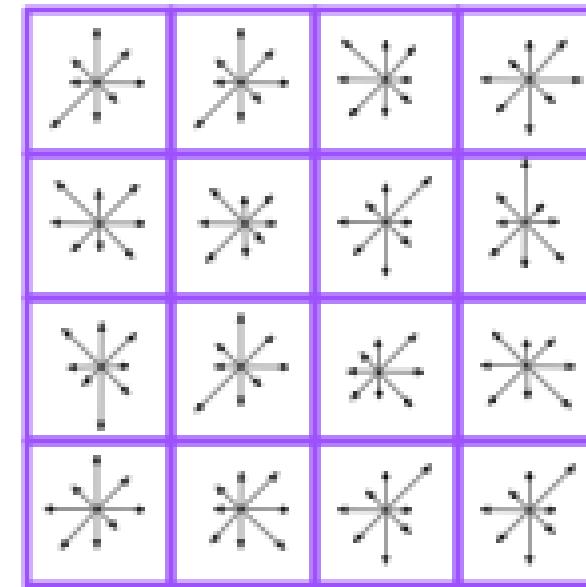


Image gradients



Keypoint descriptor

# What about 3D rotations?

---

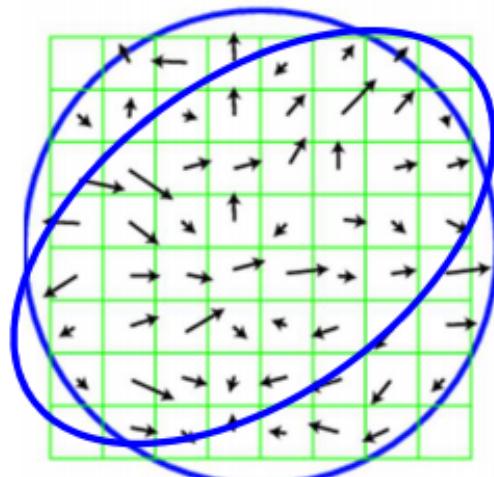
Affine transformation approximates viewpoint changes for roughly planar objects and roughly orthographic cameras



# Affine adaptation

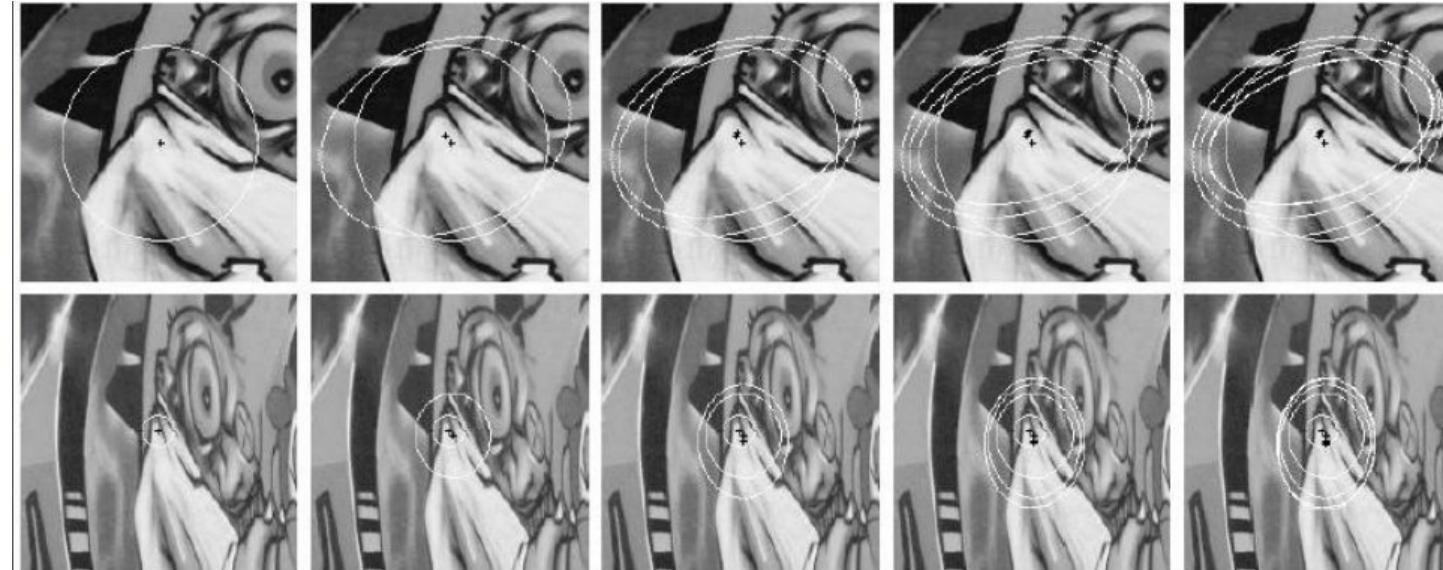
---

- Problem:
  - Determine the characteristic shape of the region.
  - Assumption: shape can be described by “local affine frame”.
- Solution: iterative approach
  - Use a circular window to compute second moment matrix.
  - Compute eigenvectors to adapt the circle to an ellipse.
  - Recompute second moment matrix using new window and iterate...

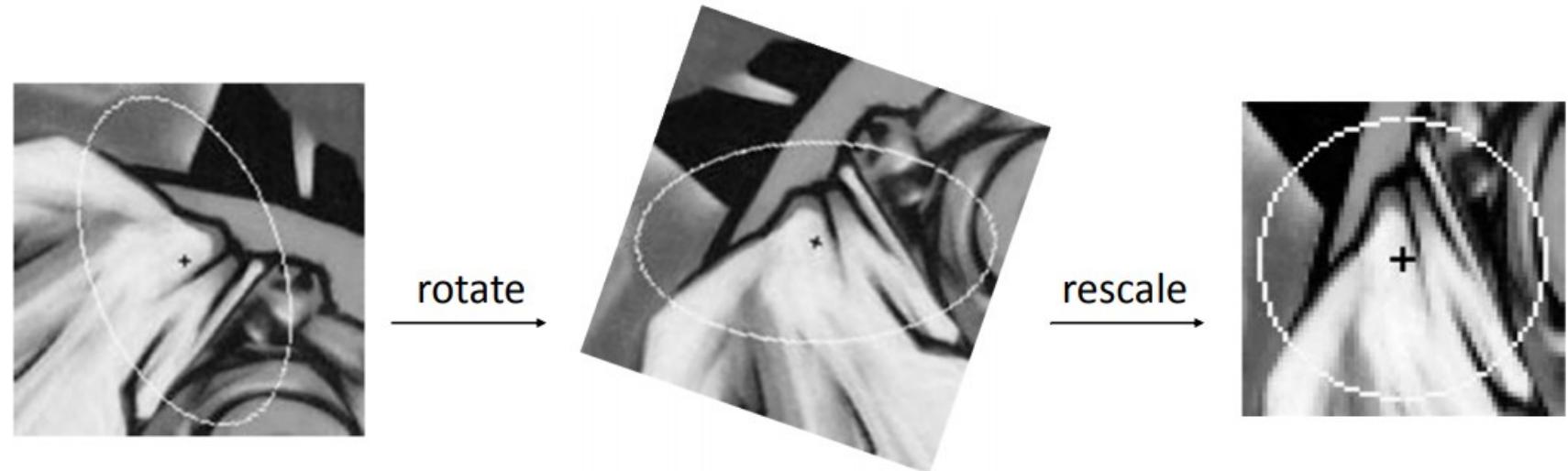


# Affine normalization ('deskewing')

Iterative  
Affine  
Adaptation



- Rotate the ellipse's main axis to horizontal
- Scale the x axis, such that it forms a circle



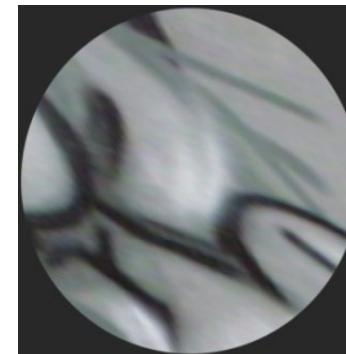
# Summary: Affine-Inv. Feature Extraction

---

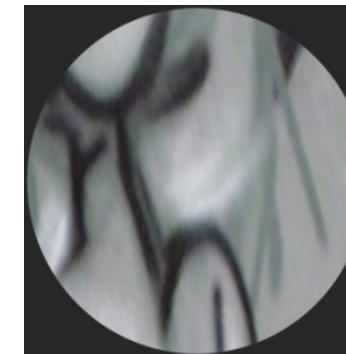
Extract affine regions



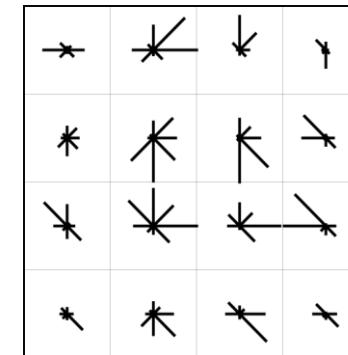
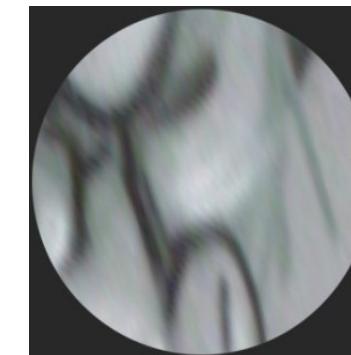
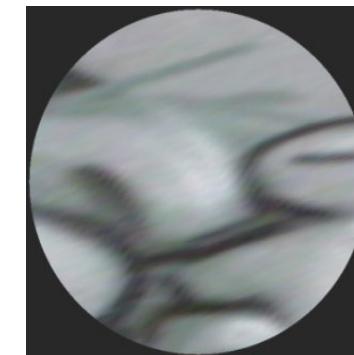
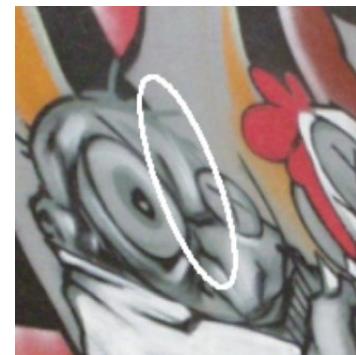
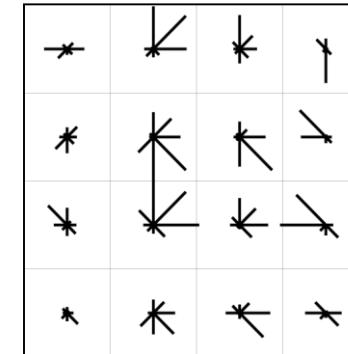
Normalize regions



Eliminate rotational ambiguity

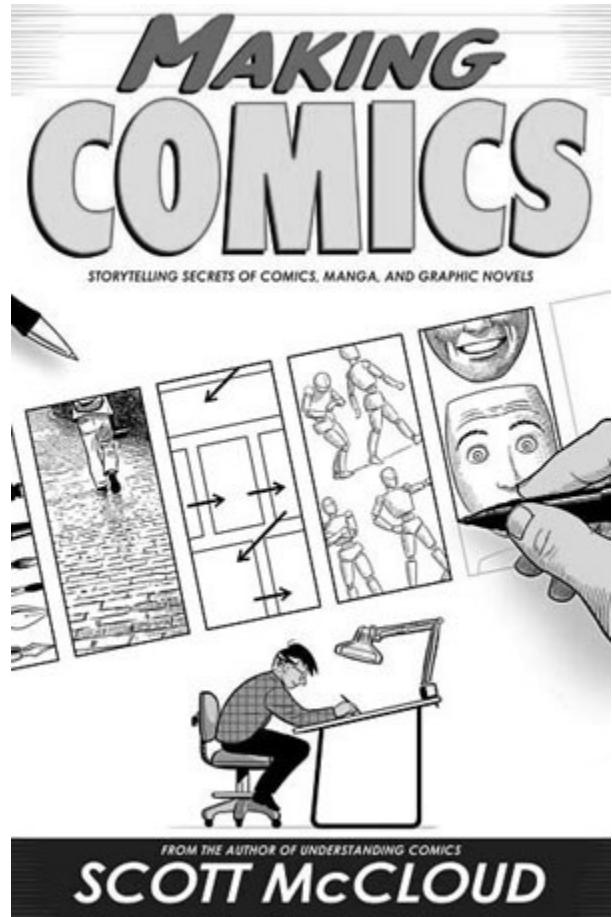


Compare descriptors

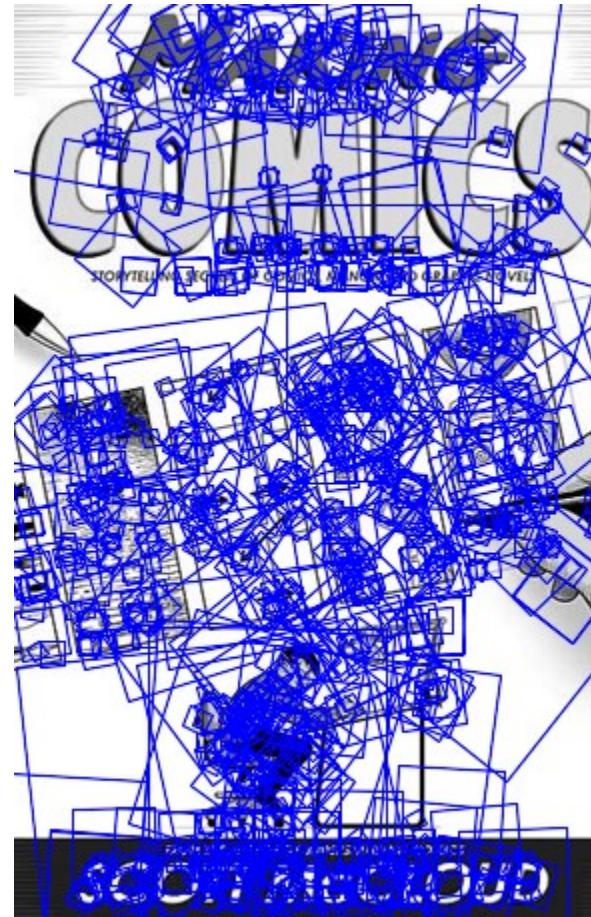


# SIFT example

---



sift



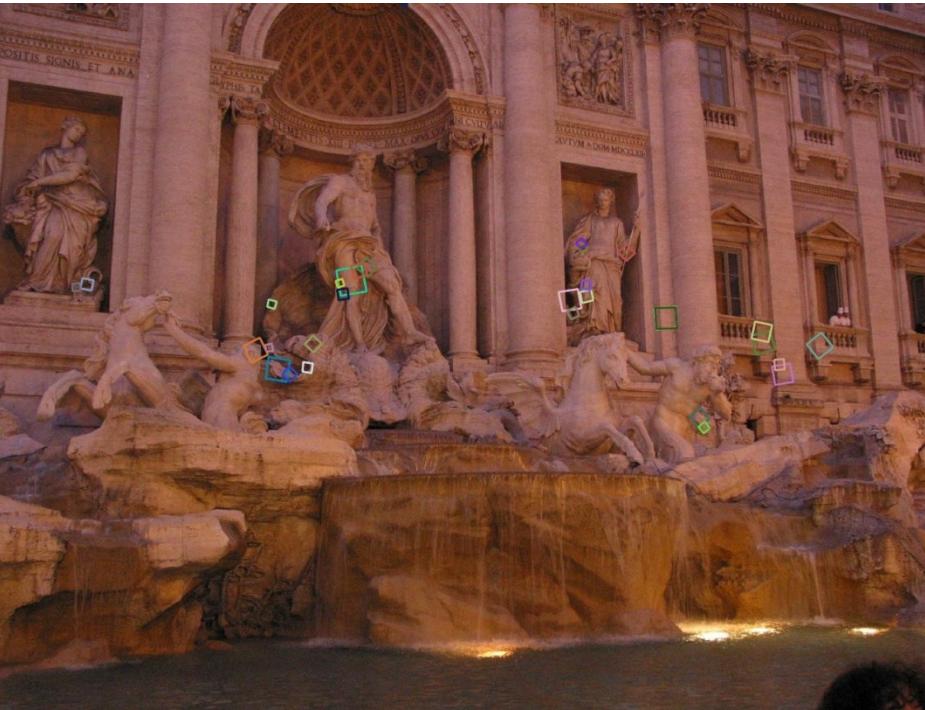
868 SIFT features

# 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



# Storia dei feature descriptor

---

Traditional (slower, accurate):

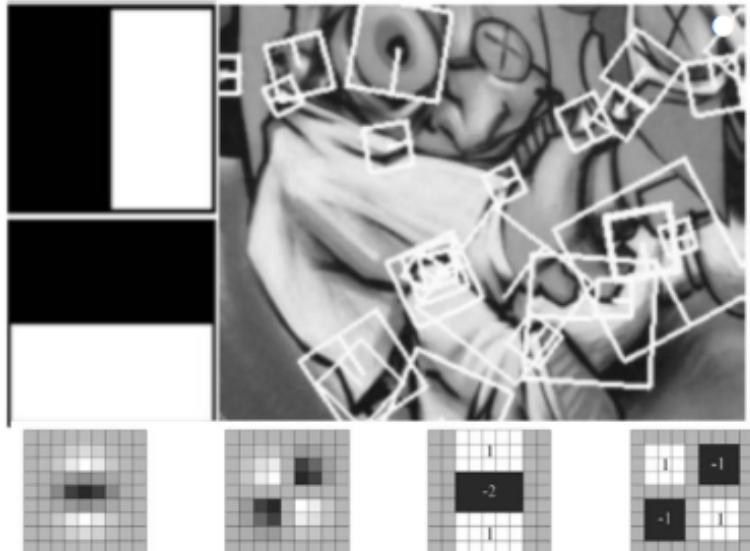
- 1999 Scale Invariant Feature Transform (Lowe)
- 2006 Speeded Up Robust Features (Bay, Tuytelaars, Van Gool)

Binary (faster, real time, smartphone, performance):

- 2010 Binary Robust Independent Elementary Features (Michael Calonder et al.)
- 2011 Oriented FAST and Rotated BRIEF (Ethan Rublee et al.)
- 2011 Binary Robust Invariant Scalable Keypoints (Leutenegger, Chli, Siegwart)
- 2012 Fast Retina Keypoint (Alahi, Ortiz, Vandergheynst)

# SURF Speeded Up Robust Features

---



## Fast approximation of SIFT idea

Efficient computation by 2D box filters & integral images  
⇒ 6 times faster than SIFT  
Equivalent quality for object identification

<http://www.vision.ee.ethz.ch/~surf>

## GPU implementation available

Feature extraction @ 100Hz  
(detector + descriptor, 640×480 img)  
<http://homes.esat.kuleuven.be/~ncorneli/gpusurf/>

<http://www.vision.ee.ethz.ch/~surf>

[Bay, ECCV'06], [Cornelis, CVGPU'08]

Source: Fei-Fei Li

# (some) Features descriptors in OpenCV

---

- SIFT (Scale Invariant Feature Transform)
  - SURF (Speeded Up Robust Features)
  - BRISK (Binary Robust Invariant Scalable Keypoints)
  - BRIEF (Binary Robust Independent Elementary Features)
  - ORB (Oriented FAST and Rotated BRIEF)
- 
- NON FREE

# BRISK – Example



```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import urllib.request

url = "https://dbloisi.github.io/corsi/images/castelmezzano-panorama.jpg"

url_response = urllib.request.urlopen(url)
numpy_img = np.array(bytearray(url_response.read()), dtype=np.uint8)
img = cv.imdecode(numpy_img, -1)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

brisk = cv.BRISK_create(thresh=60, octaves=3, patternScale=1.0)

kp = brisk.detect(gray, None)

kp, des = brisk.compute(gray, kp)

img = cv.drawKeypoints(gray, kp, None,
                       flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

plt.axis('off')
plt.imshow(img)

cv.imwrite('brisk.png', img)
```

**brisk.png**



# BRIEF – Example

---



```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import urllib.request

url = "https://dbloisi.github.io/corsi/images/castelmezzano-panorama.jpg"

url_response = urllib.request.urlopen(url)
numpy_img = np.array(bytearray(url_response.read()), dtype=np.uint8)
img = cv.imdecode(numpy_img, -1)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

star = cv.xfeatures2d.StarDetector_create()

brief = cv.xfeatures2d.BriefDescriptorExtractor_create()

kp = star.detect(gray, None)

kp, des = brief.compute(gray, kp)

img = cv.drawKeypoints(gray, kp, None,
                       flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

plt.axis('off')
plt.imshow(img)

cv.imwrite('brief.png', img)
```

**brief.png**



# ORB – Example

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import urllib.request

url = "https://dbloisi.github.io/corsi/images/castelmezzano-panorama.jpg"

url_response = urllib.request.urlopen(url)
numpy_img = np.array(bytearray(url_response.read()), dtype=np.uint8)
img = cv.imdecode(numpy_img, -1)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

orb = cv.ORB_create()

kp = orb.detect(gray, None)

kp, des = orb.compute(gray, kp)

img = cv.drawKeypoints(gray, kp, None,
                       flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

plt.axis('off')
plt.imshow(img)

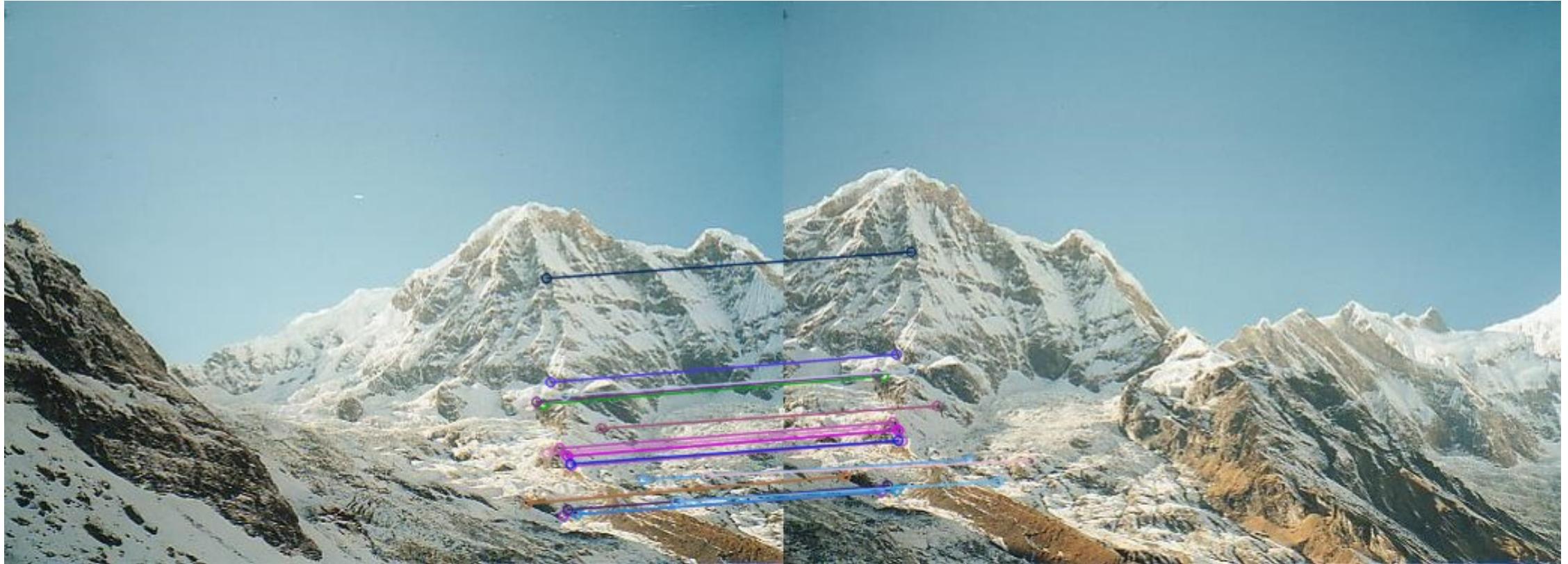
cv.imwrite('orb.png', img)
```

Orb.png



# Feature matching

---





**UNIVERSITÀ DEGLI STUDI  
DELLA BASILICATA**

*Corso di Visione e Percezione*  
A.A. 2019/2020

# Feature Descriptors

Aprile 2020

