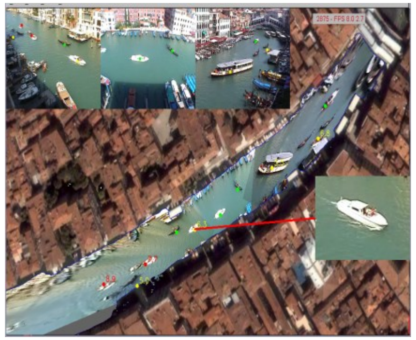
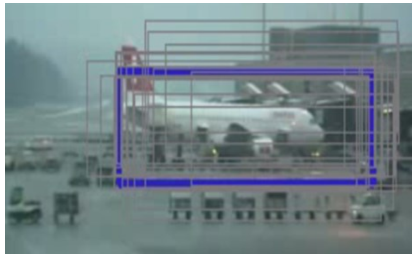
UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA

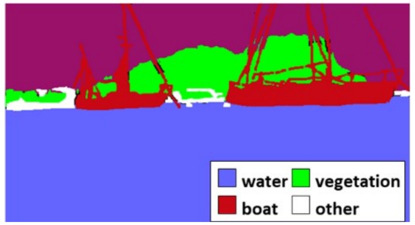
Corso di Visione e Percezione

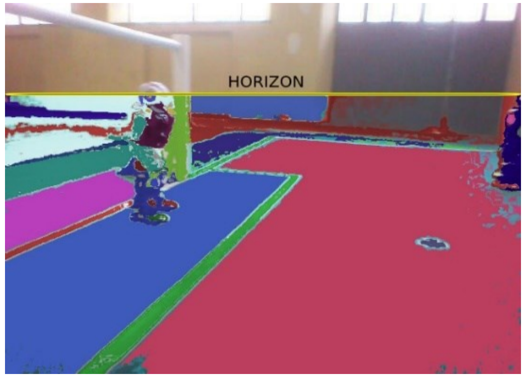


Docente
Domenico D. Bloisi

Feature Descriptors

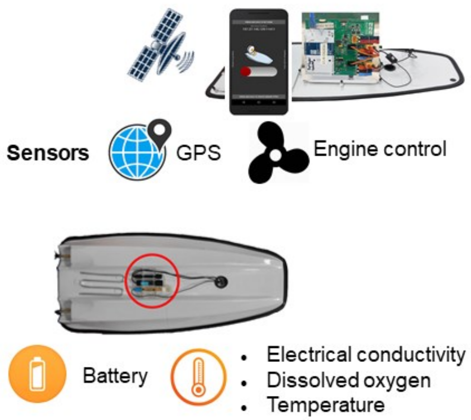




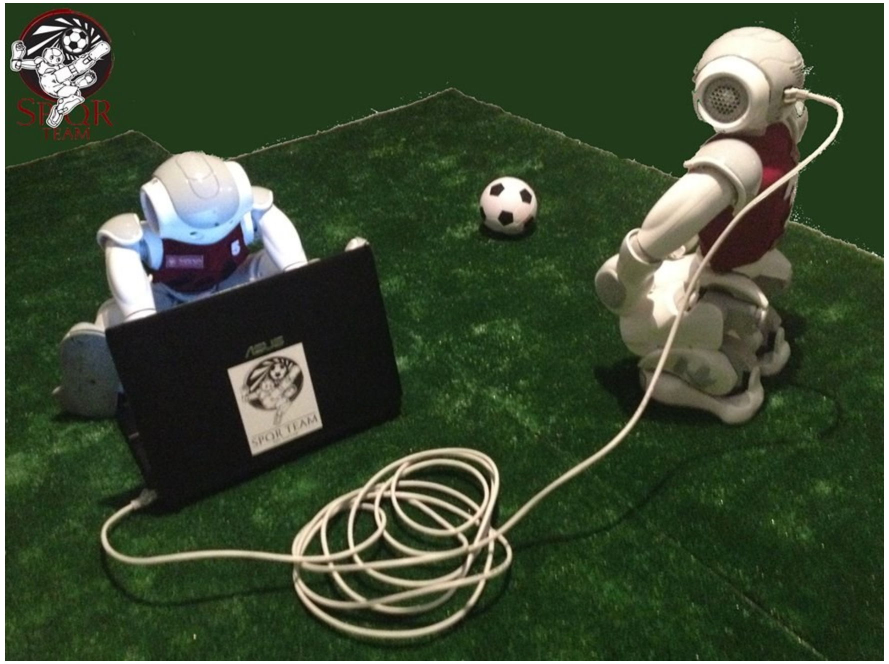


Domenico Daniele Bloisi

- 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: II 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/NjI2MjA4MzgzNDFa?cjc=xgolays

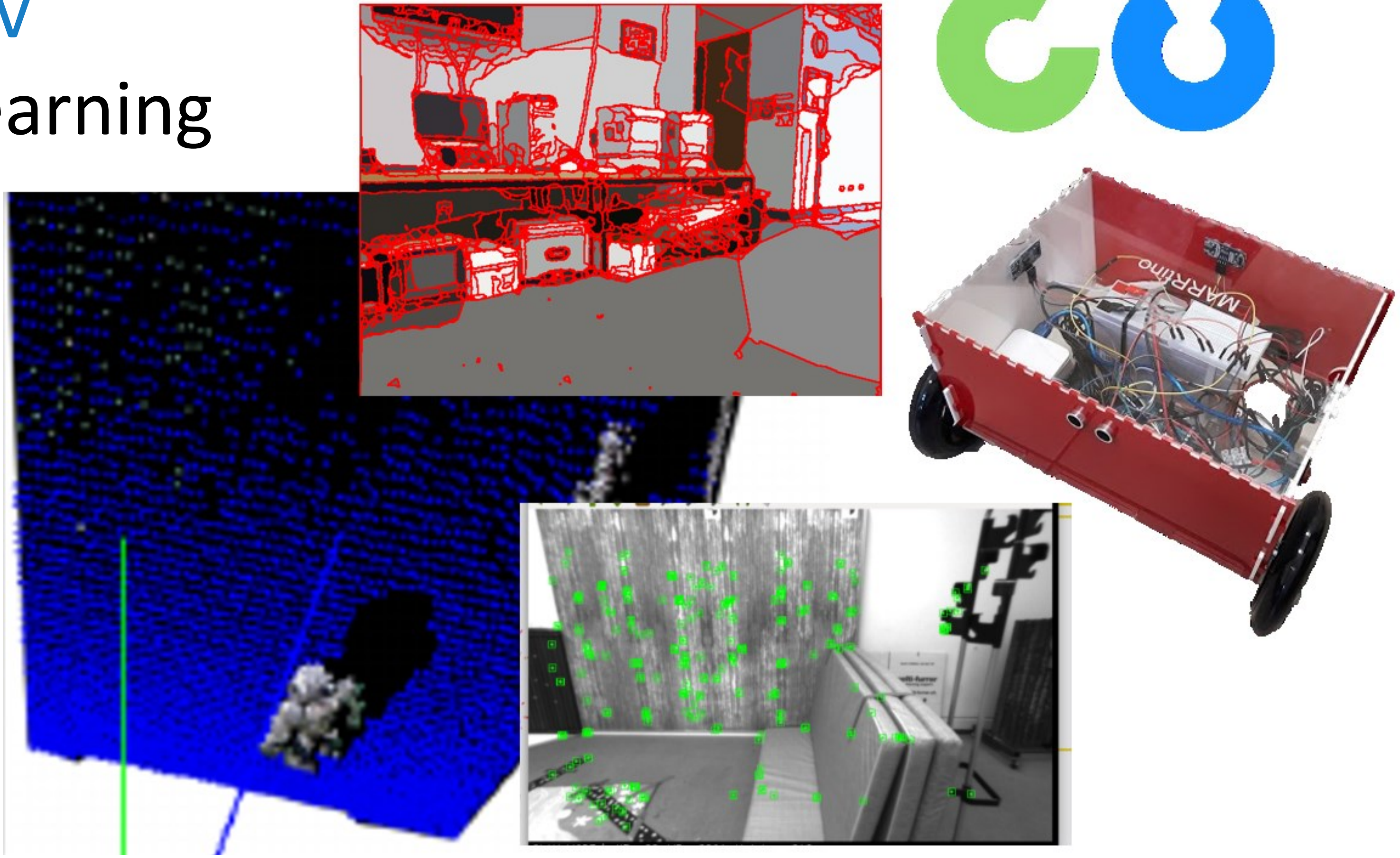
Ricevimento

- 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

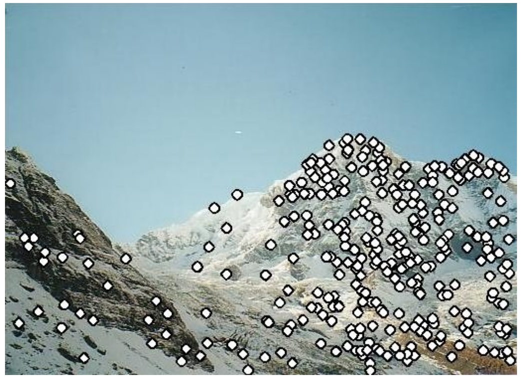
- Queste slide sono adattate da
  Noah Snavely - CS5670: Computer Vision
  "Lecture 5: Feature descriptors and matching"

- 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:

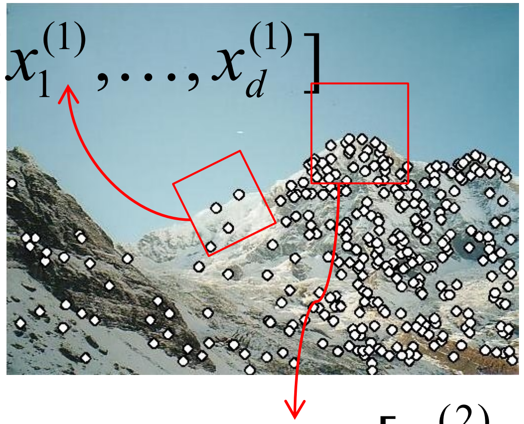
   Identify the interest points



2) Description:

   Extract vector feature descriptor surrounding each interest point

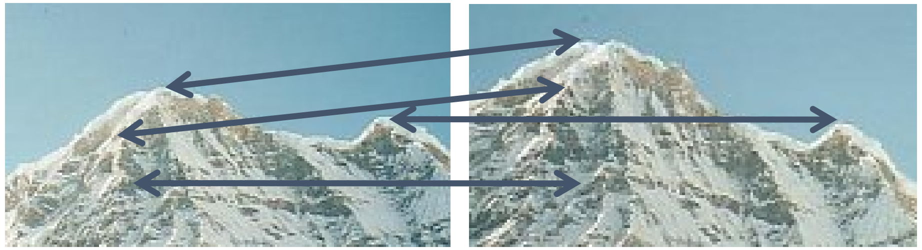
$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

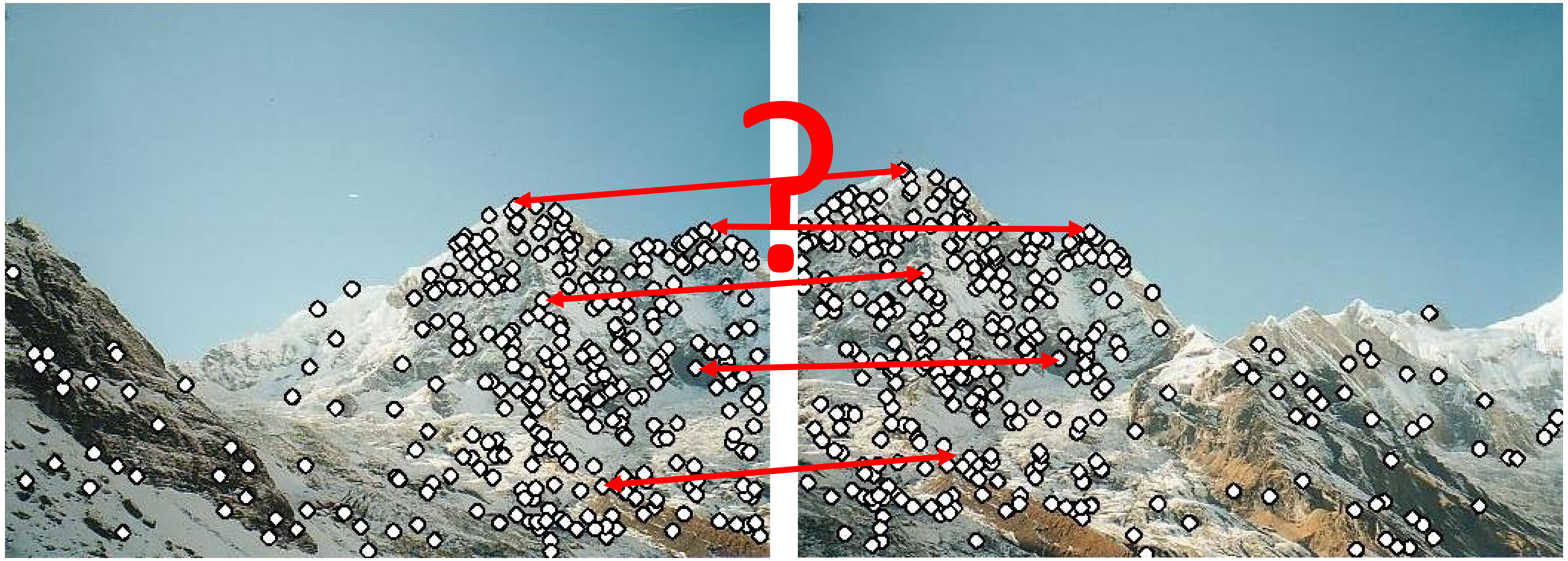
3) Matching:

   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

Come capire se due descrittori sono simili?

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

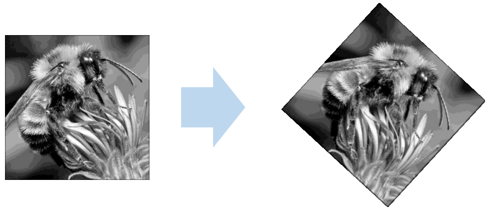
Discriminability:

Descriptor should be highly unique for each point

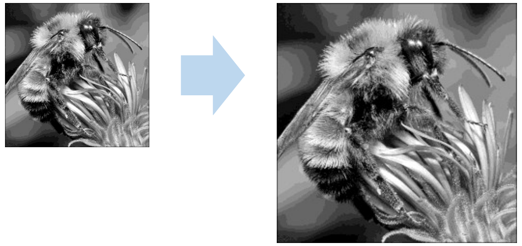
Image transformations

Geometric

Rotation



Scale



Photometric

Intensity change

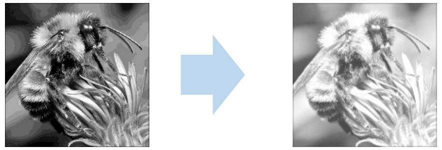
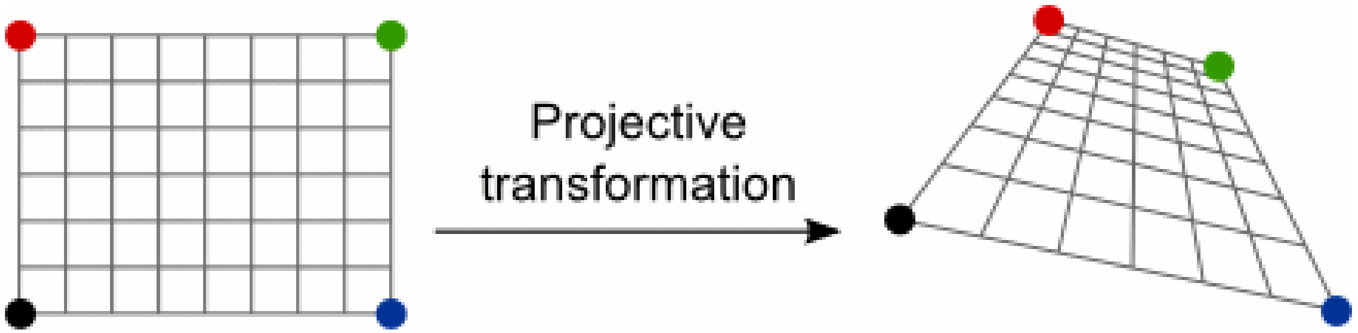


Image transformations

Affine transformations preserve parallelism



Projective transformations do not preserve parallelism, length, and angle

https://www.graphicsmill.com/docs/gm/affine-and-projective-transformations.htm

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

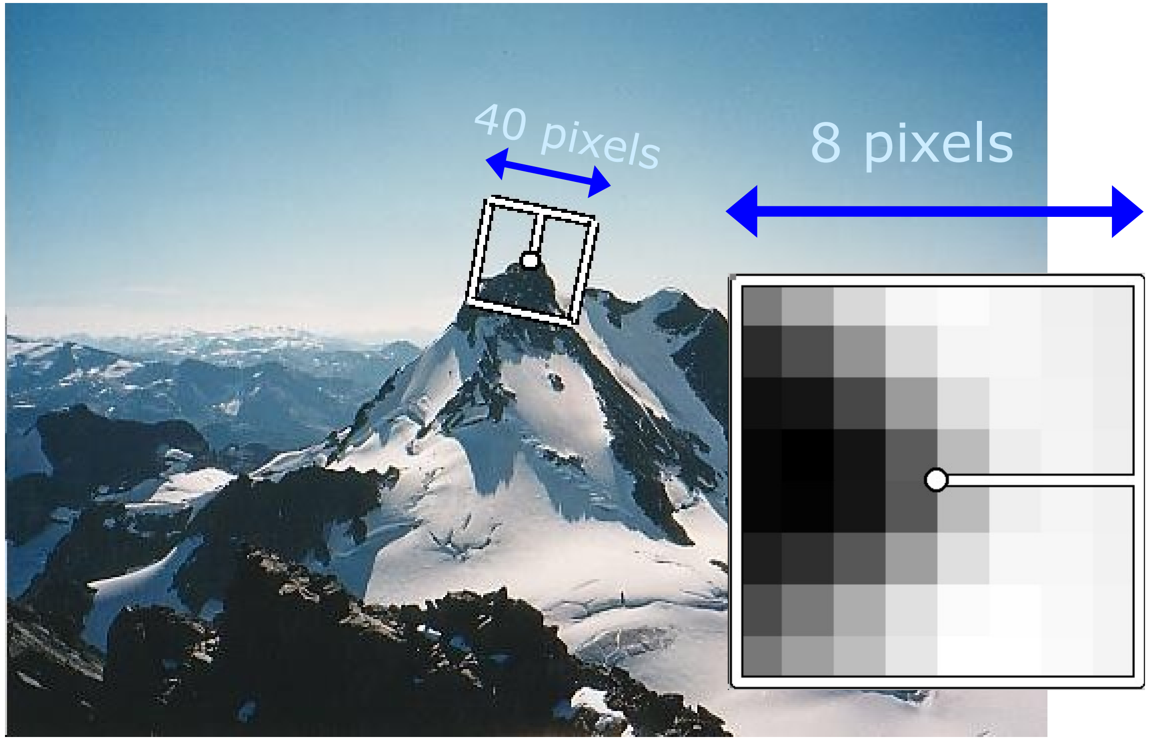
- Find dominant orientation of the image patch
- Rotate the patch according to this angle

Figure by Matthew Brown

Multiscale Oriented PatcheS descriptor

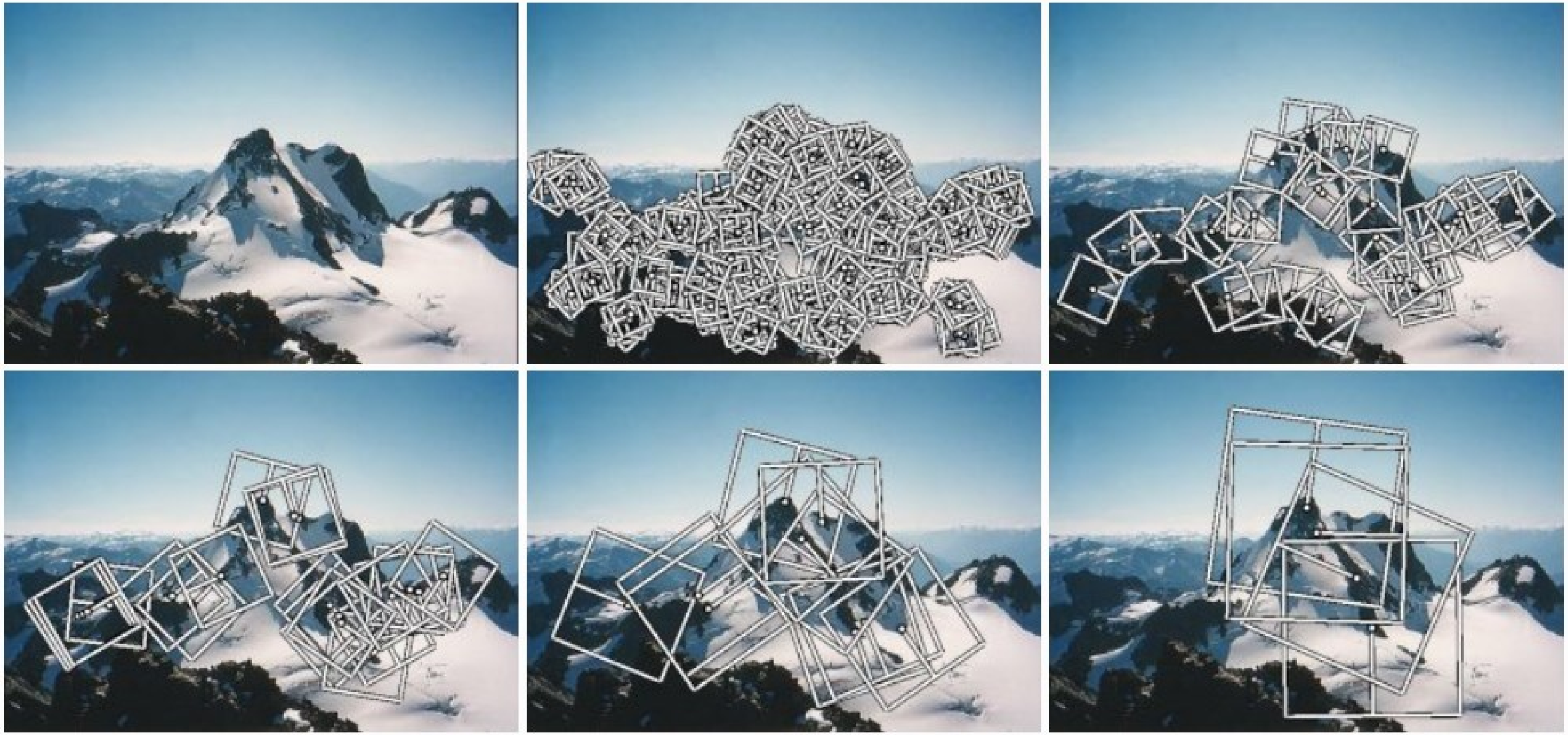
Take 40x40 square window around detected feature

- 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)



Adapted from slide by Matthew Brown

Detection at multiple scales

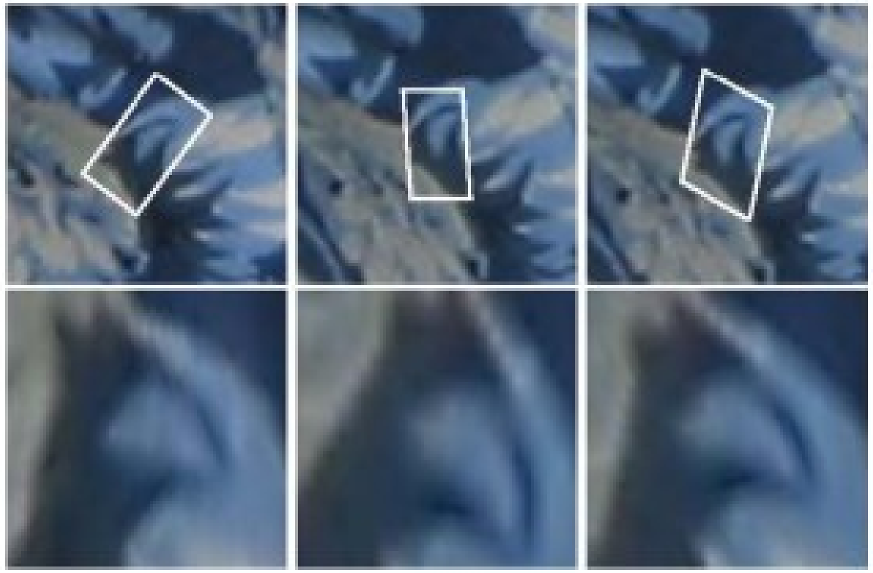


Multi-scale Oriented Patches (MOPS) extracted at 5 pyramid levels

http://matthewalunbrown.com/mops/mops.html

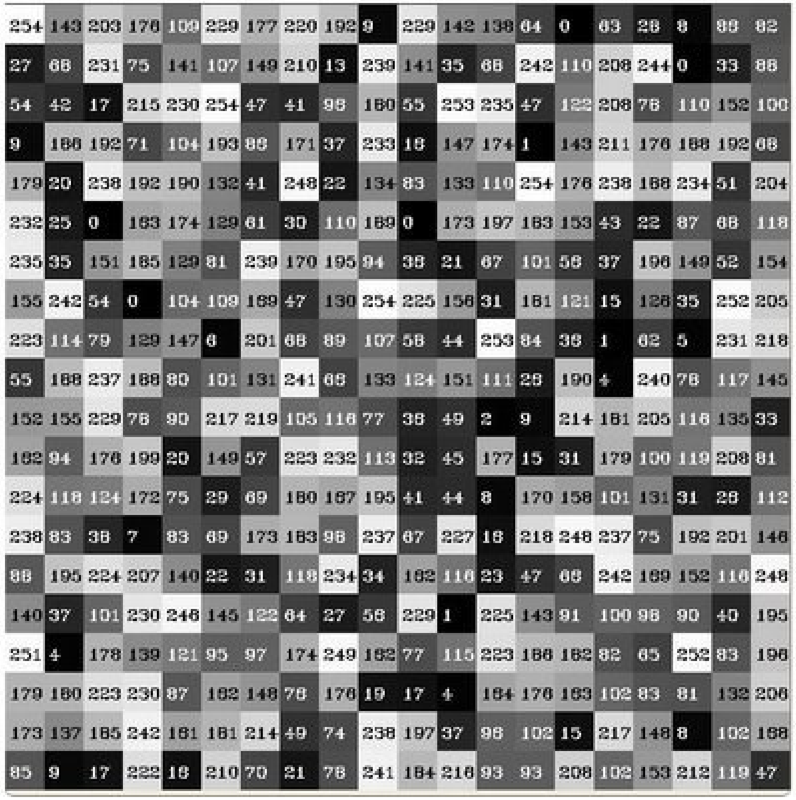
Svantaggi delle patches come descrittori

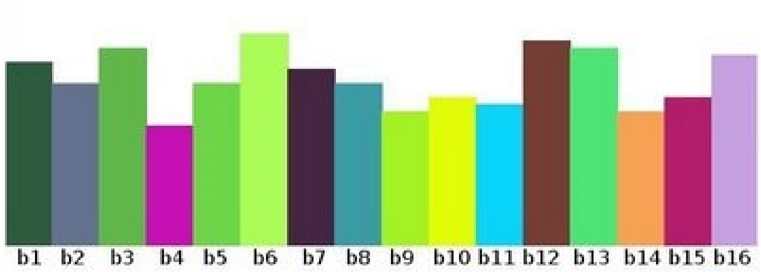
- Disadvantage of patches as descriptors:
  - Small shifts can affect matching score a lot



Source: Fei-Fei Li

Histograms as descriptors





https://docs.opencv.org/3.4/d8/dbc/tutorial_histogram_calculation.html

Scale Invariant Feature Transform

Basic idea:

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

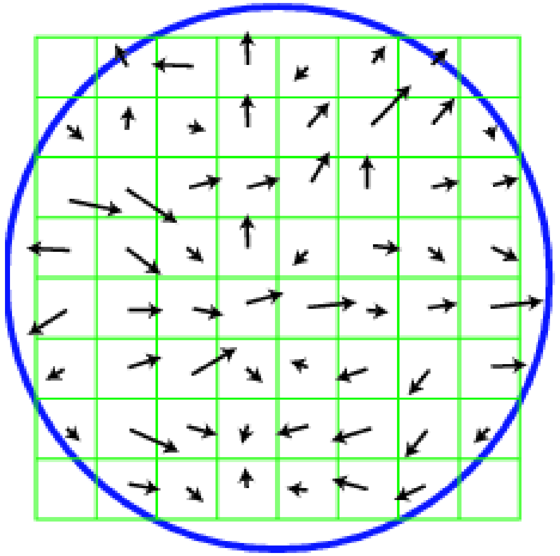
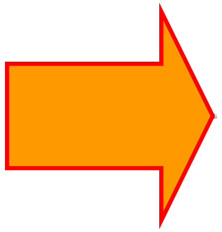
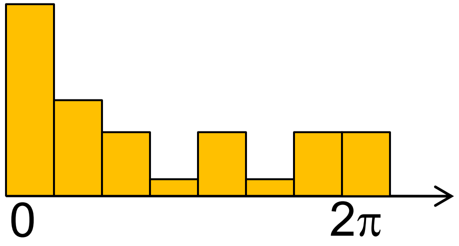
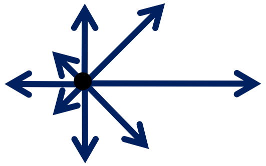


Image gradients





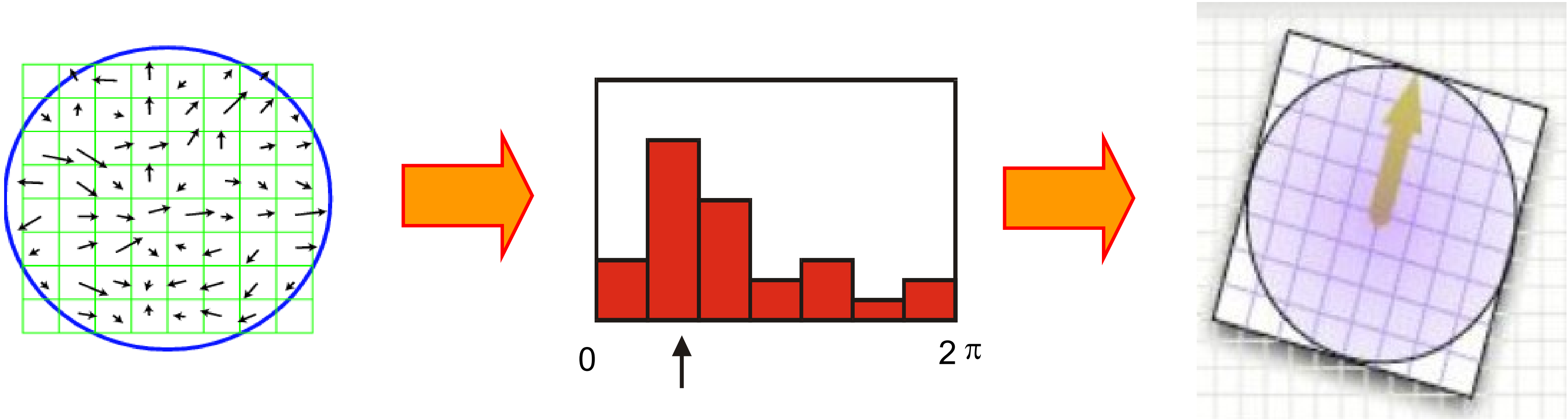
angle histogram



Adapted from slide by David Lowe

Finding a reference orientation

Assign reference orientation at peak of smoothed histogram



Source: Svetlana Lazebnik

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

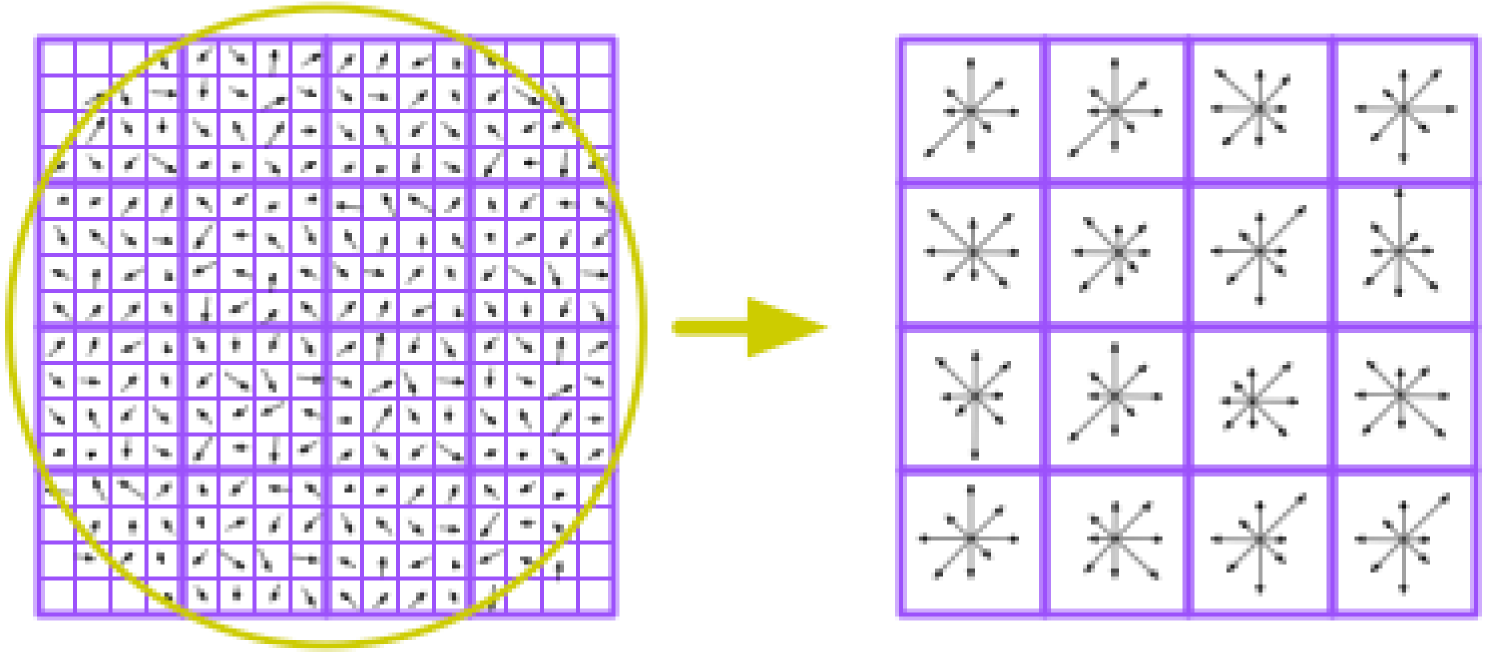


Image gradients

Keypoint descriptor

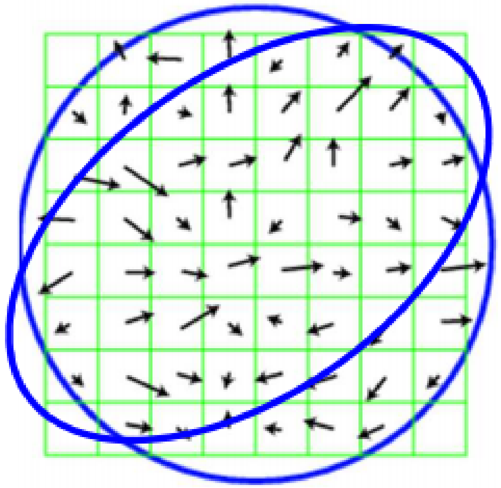
What about 3D rotations?

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

Source: Svetlana Lazebnik

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…



Source: Fei-Fei Li

Affine normalization ('deskewing')

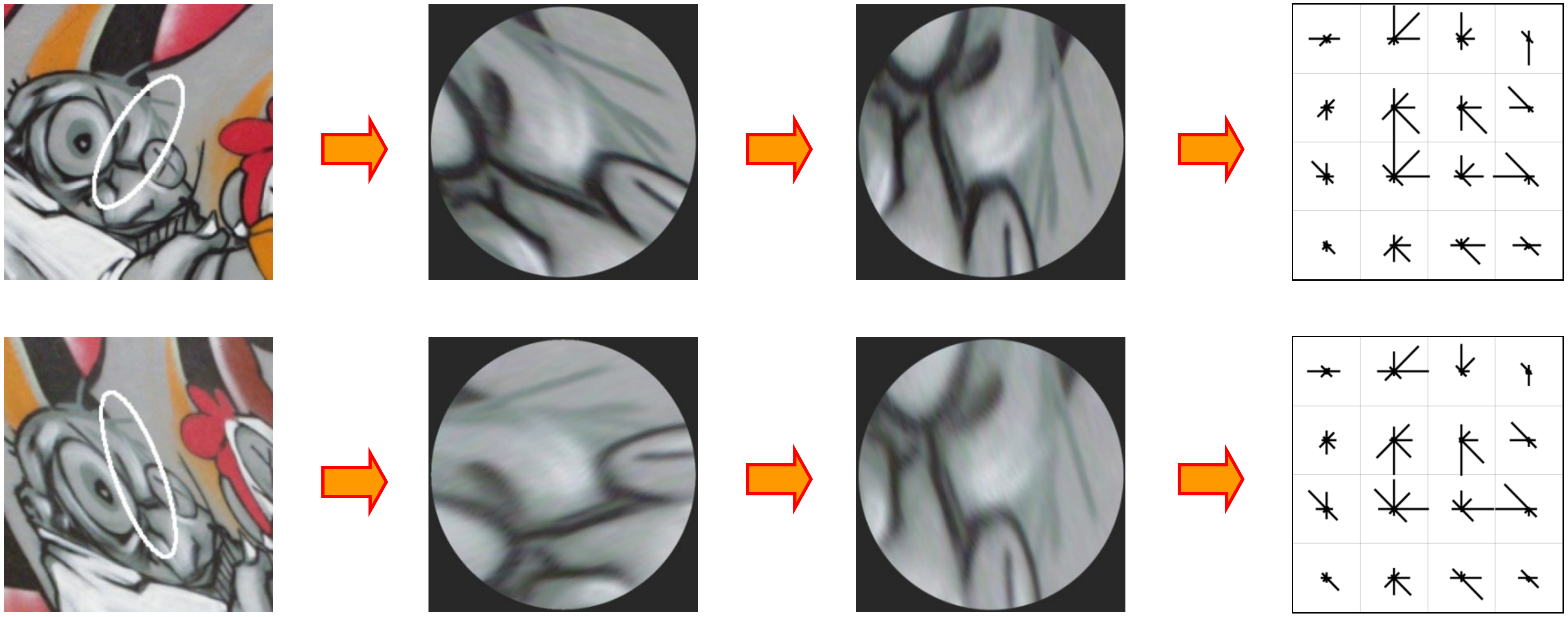
Iterative Affine Adaptation

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

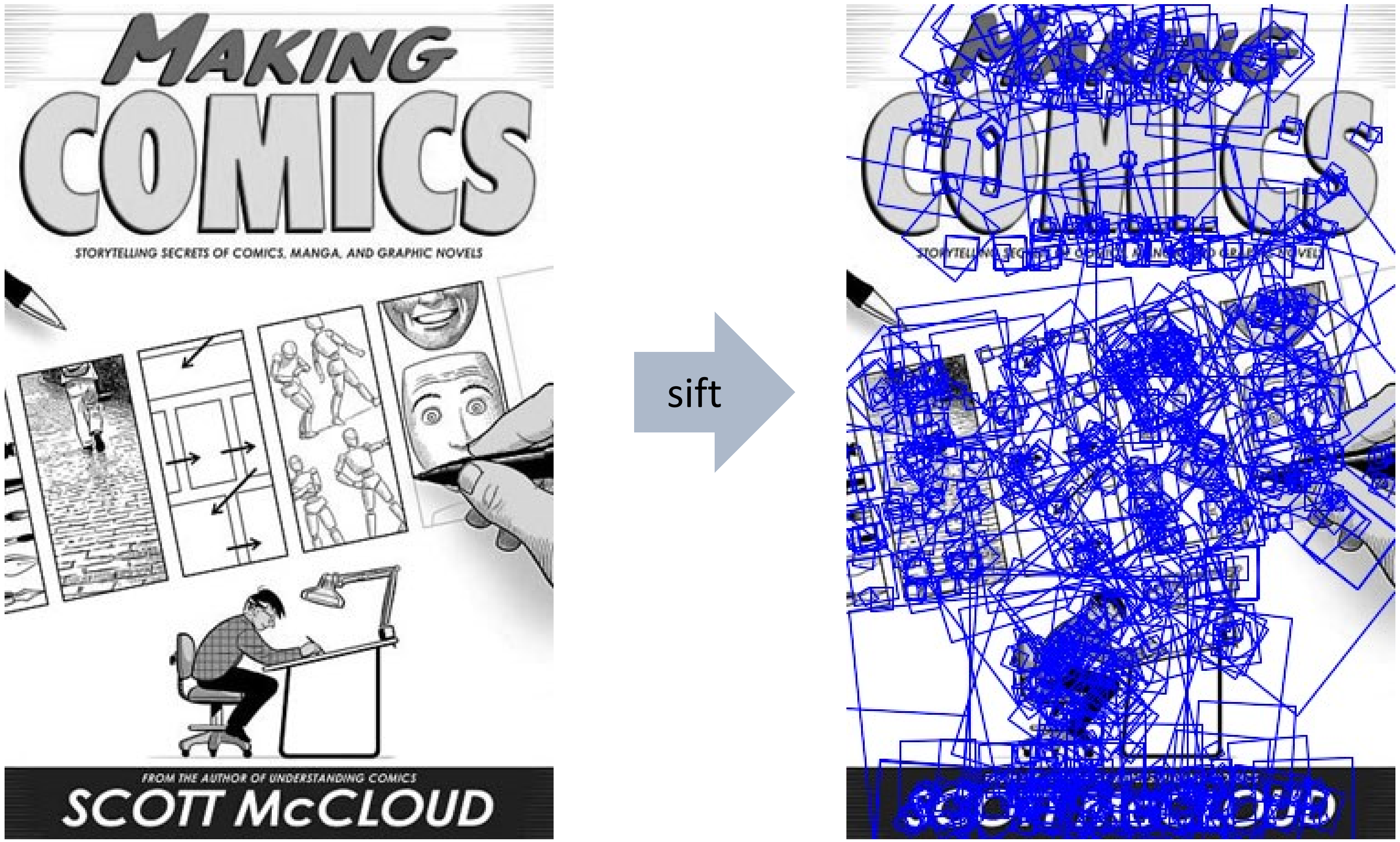
Source: Fei-Fei Li

Summary: Affine-Inv. Feature Extraction

Extract affine regions | Normalize regions | Eliminate rotational ambiguity | Compare descriptors



SIFT example



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):

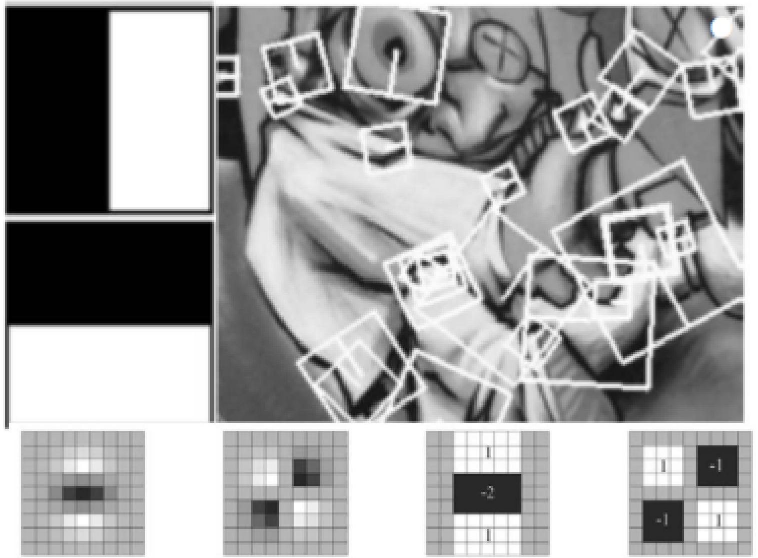
- 1999 Scale Invariant Feature Transform (Lowe)
- 2006 Speeded Up Robust Features (Bay, Tuytelaars, Van Gool)

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

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

Source: Stefan Haller

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/

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

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

Source: Fei-Fei Li

(some) Features descriptors in OpenCV

- [SIFT (Scale Invariant Feature Transform)]
- [SURF (Speeded Up Robust Features)]

NON FREE
but the patent expired in the year 2020

- [BRISK (Binary Robust Invariant Scalable Keypoints)]
- [BRIEF (Binary Robust Independent Elementary Features)]
- [ORB (Oriented FAST and Rotated BRIEF)]

SIFT – Example

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

url = "https://web.unibas.it/bloisi/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)

# Initiate SIFT detector
sift = cv.SIFT_create()
# find the keypoints and descriptors with SIFT
kp = sift.detect(img, None)

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


cv.imwrite("surf.png", img)
plt.axis('off')
plt.imshow(img),plt.show()
```

OpenCV
Version 4.5.1

surf.png

BRIEF – Example

```python
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

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)
```

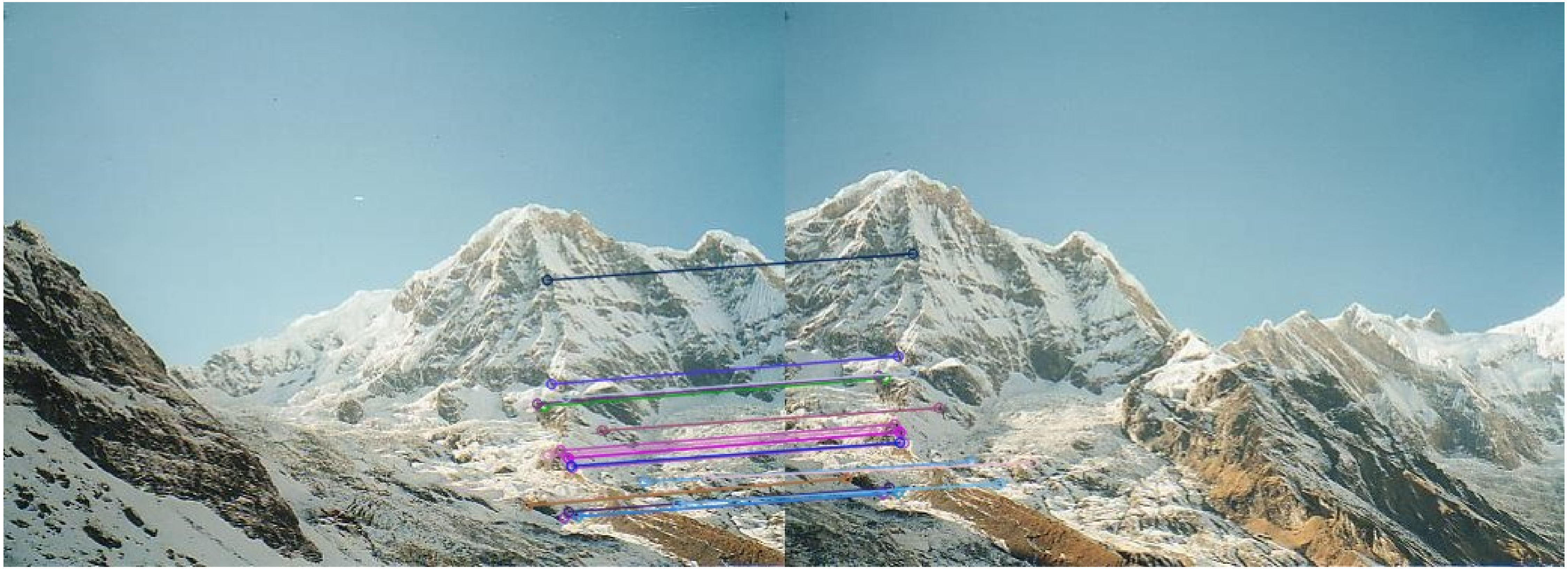
brisk.png

ORB – Example

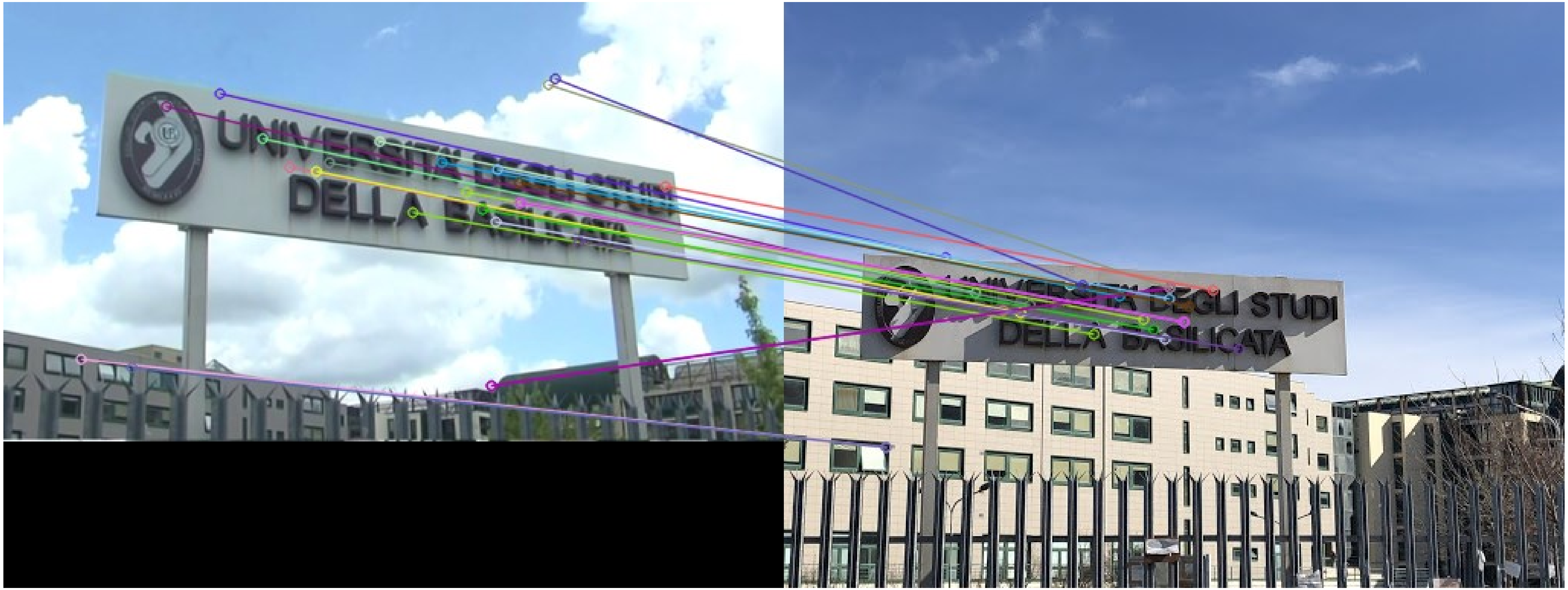
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import urllib.request

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



Feature matching

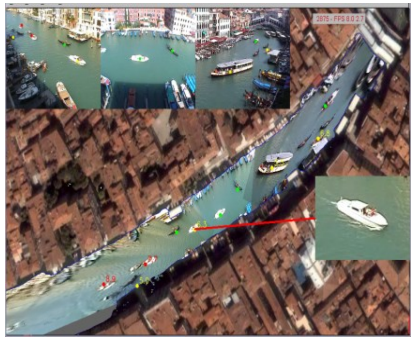


UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA



Corso di Visione e Percezione

Docente
Domenico D. Bloisi



Feature Descriptors