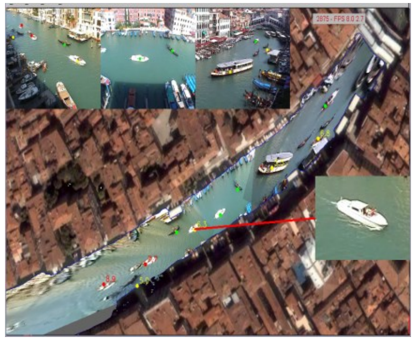
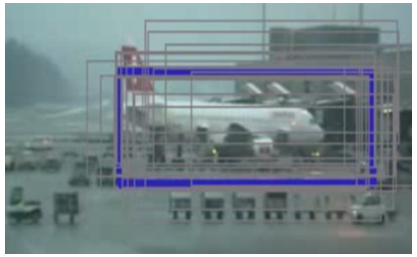
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA



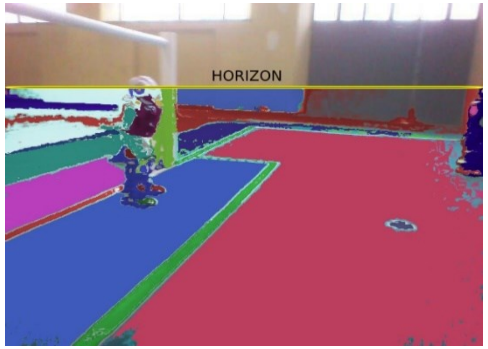
Corso di Sistemi Informativi
A.A. 2018/19

Docente
Domenico Daniele Bloisi

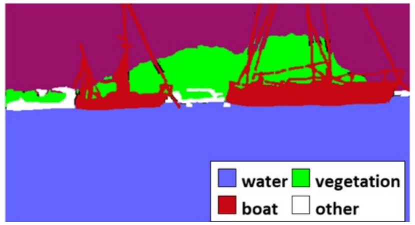



Feature Descriptors

Aprile 2019





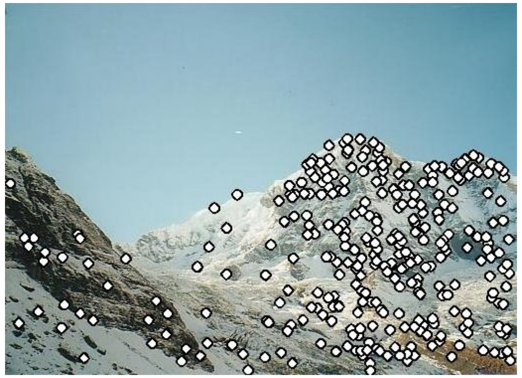
Riferimenti

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

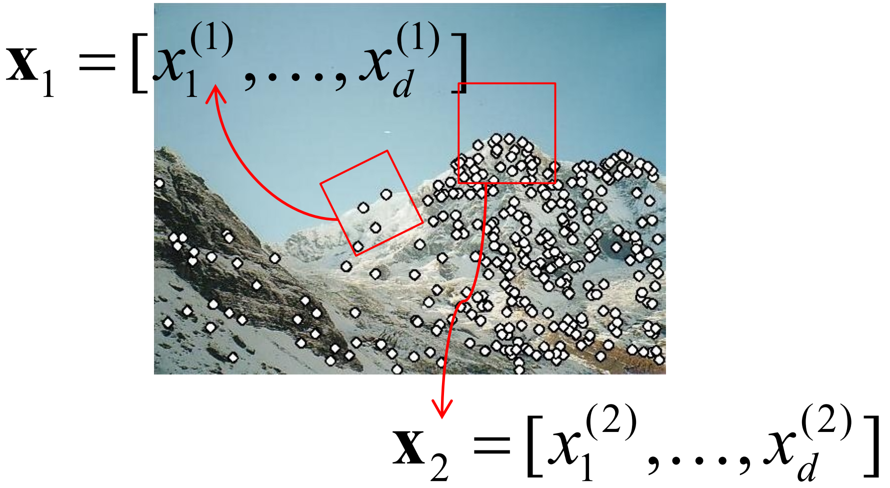
- I contenuti fanno riferimento al capitolo 4 del libro
  "Computer Vision: Algorithms and Applications"
  di Richard Szeliski, disponibile al seguente indirizzo
  http://szeliski.org/Book/

Local features: main components

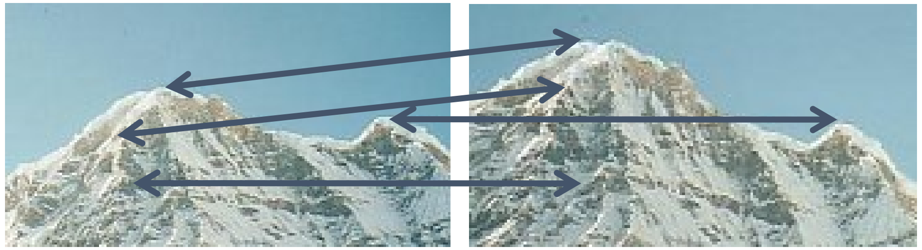
1) Detection: Identify the interest points



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

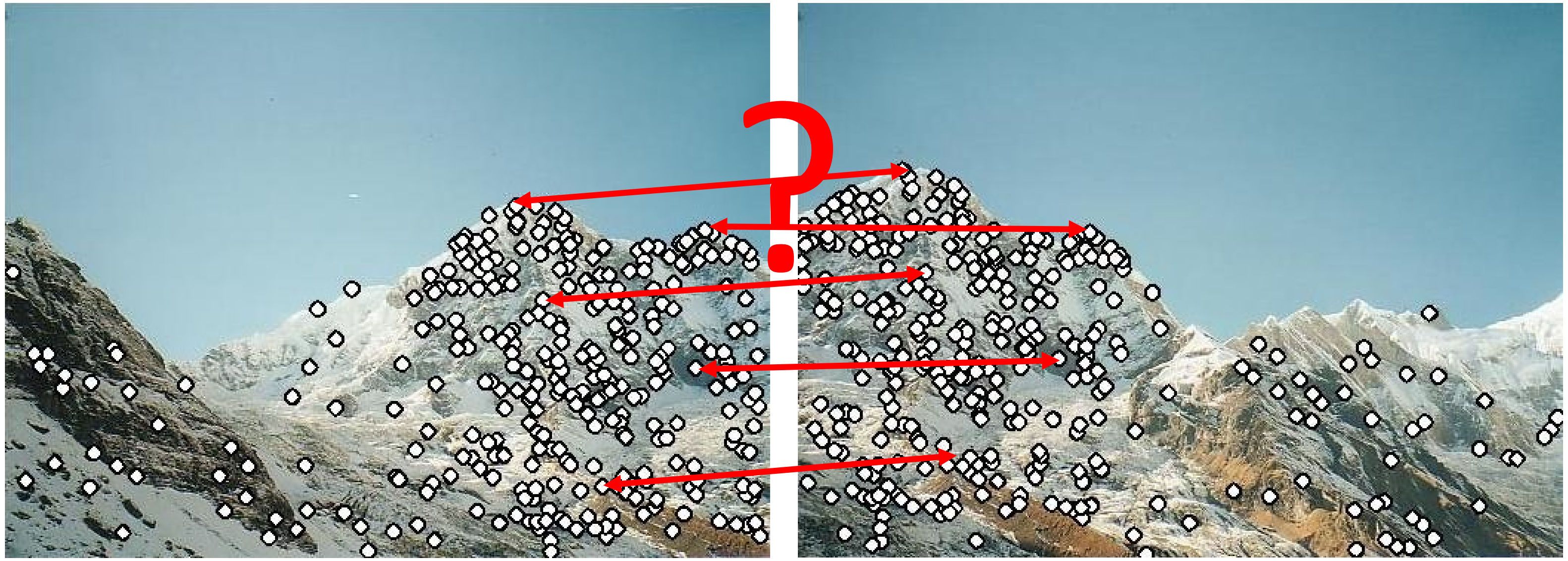


3) Matching: Determine correspondence between descriptors in two views



Feature descriptors

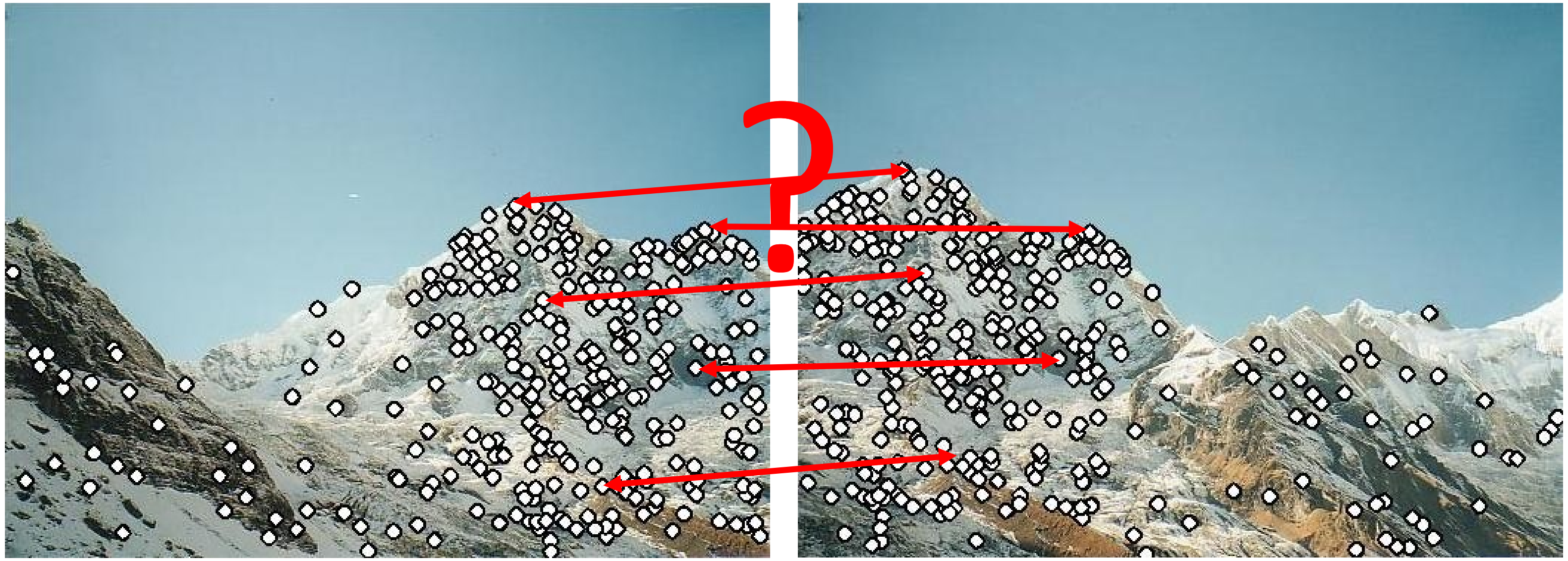
We know how to detect good points
Next question: **How to match them?**



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

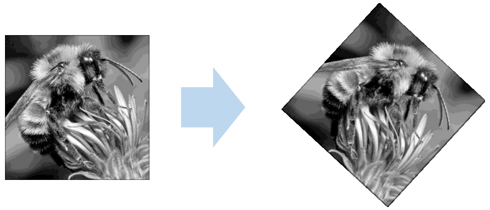
Discriminability:

- Descriptor should be highly unique for each point

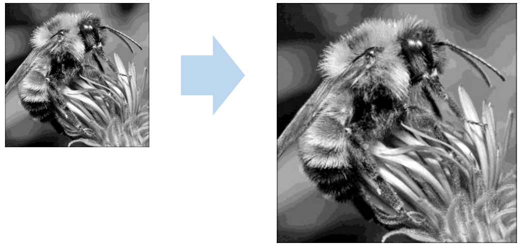
Image transformations

Geometric

Rotation

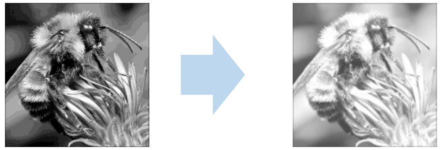


Scale



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

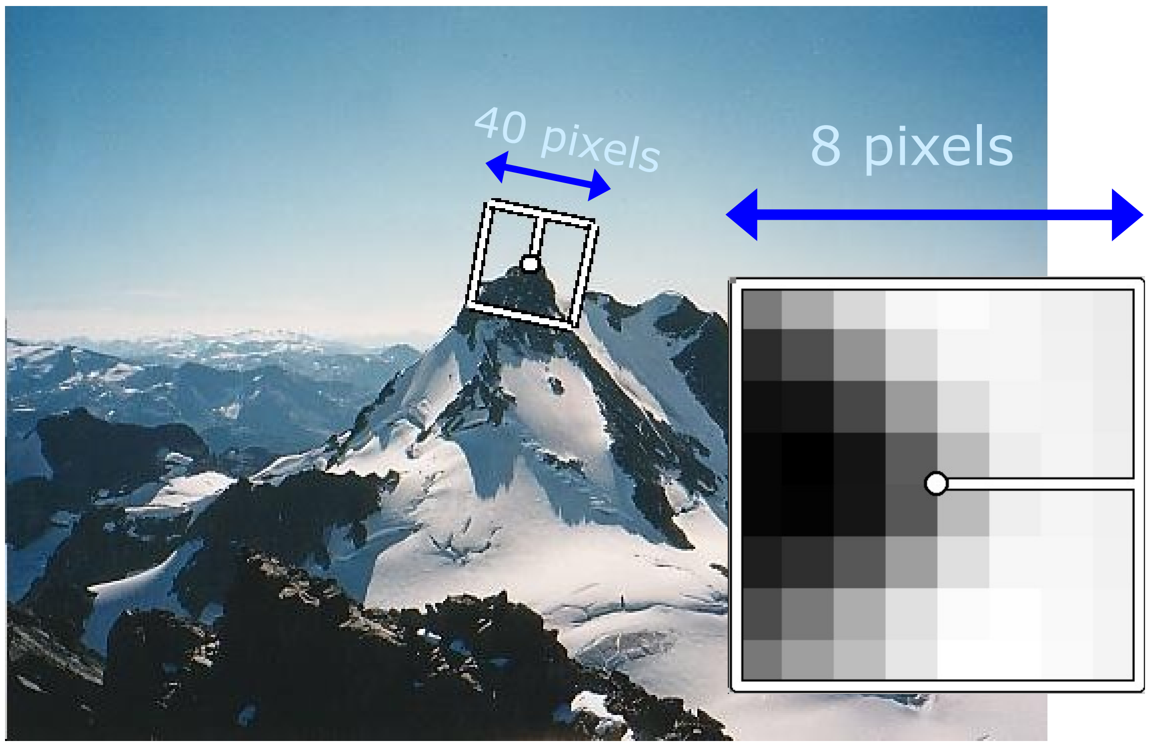
- Find dominant orientation of the image patch
- Rotate the patch according to this angle

Figure by Matthew Brown

Multiscale Oriented PatcheS descriptor

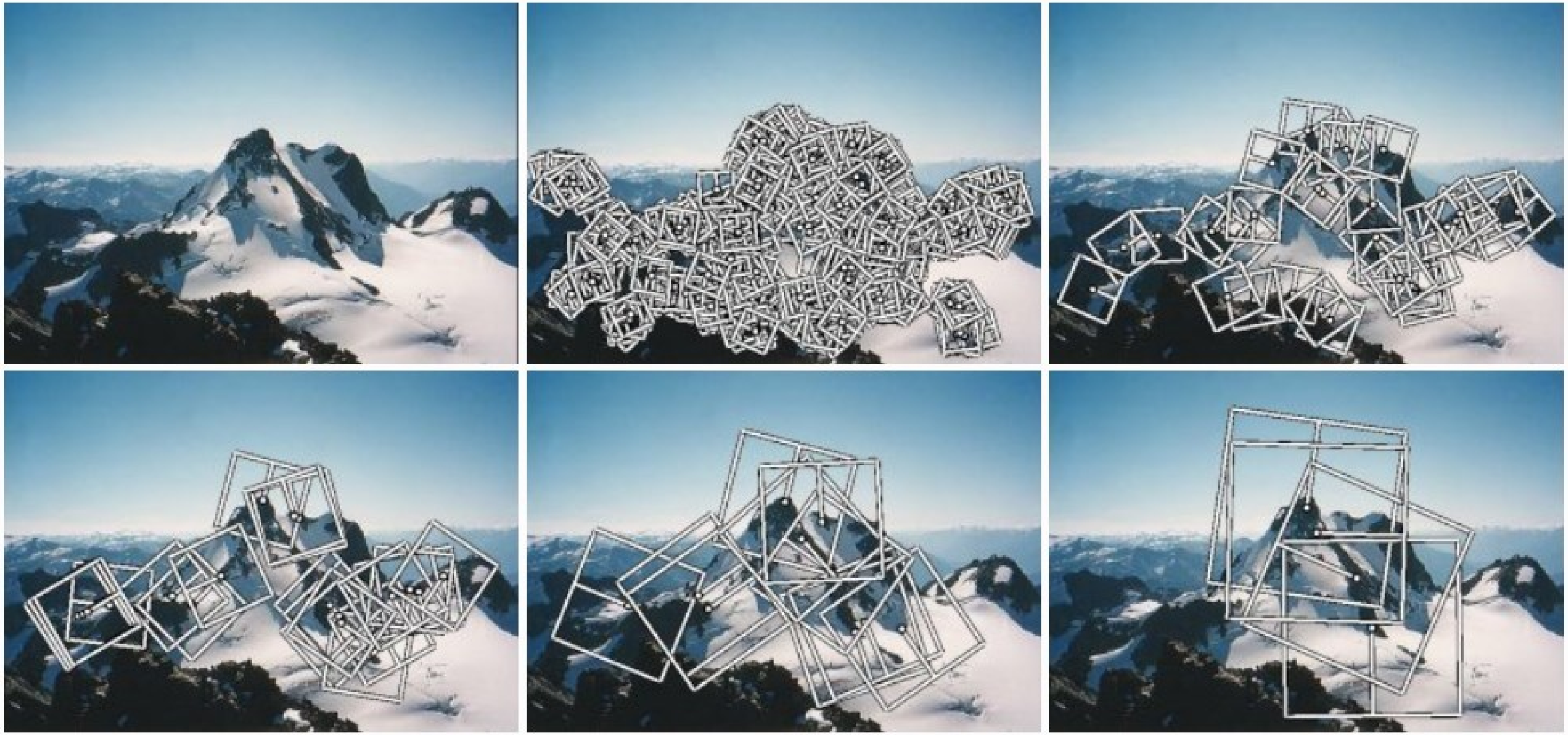
Take 40x40 square window around detected feature

- 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)



Adapted from slide by Matthew Brown

Detection at multiple scales

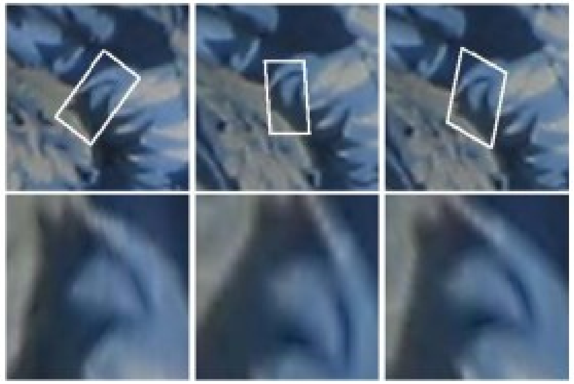


Multi-scale Oriented Patches (MOPS) extracted at 5 pyramid levels

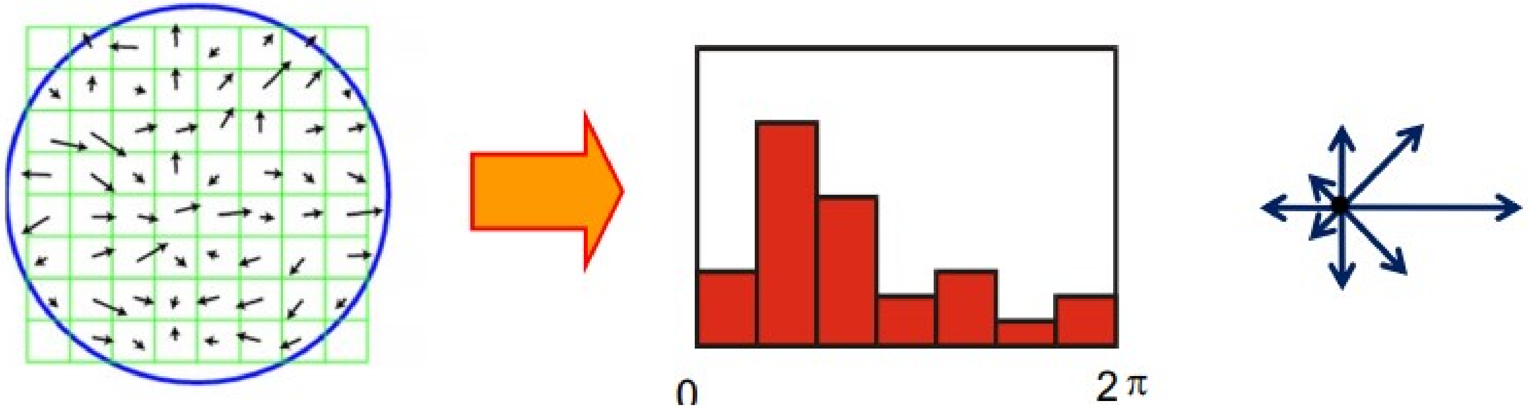
http://matthewalunbrown.com/mops/mops.html

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°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

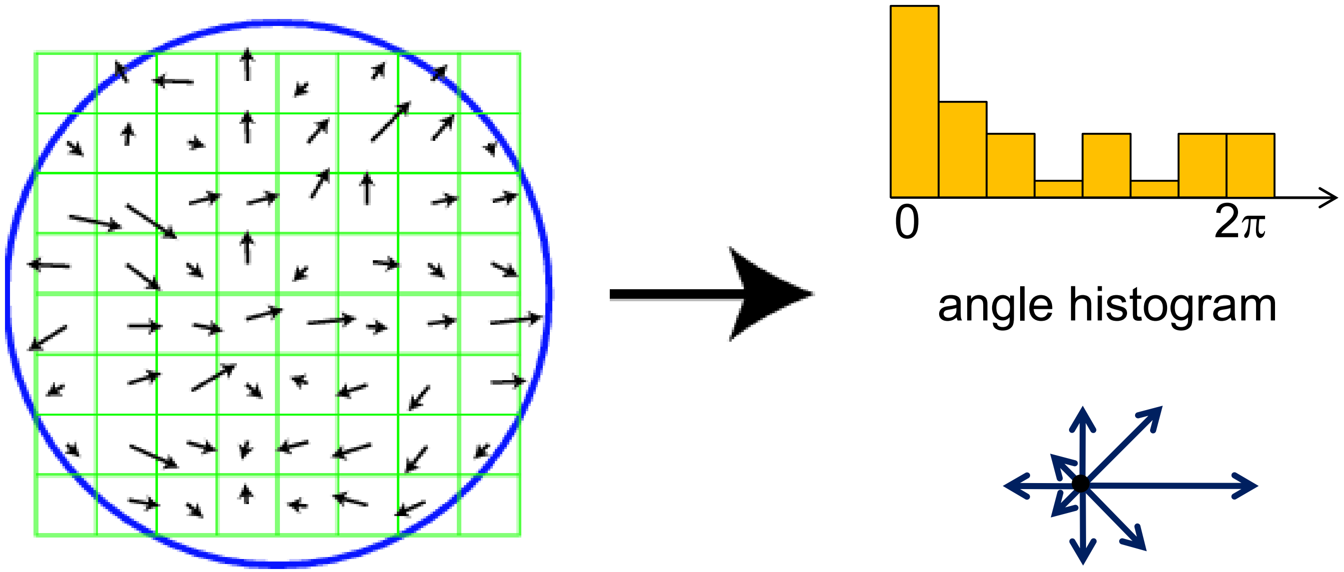
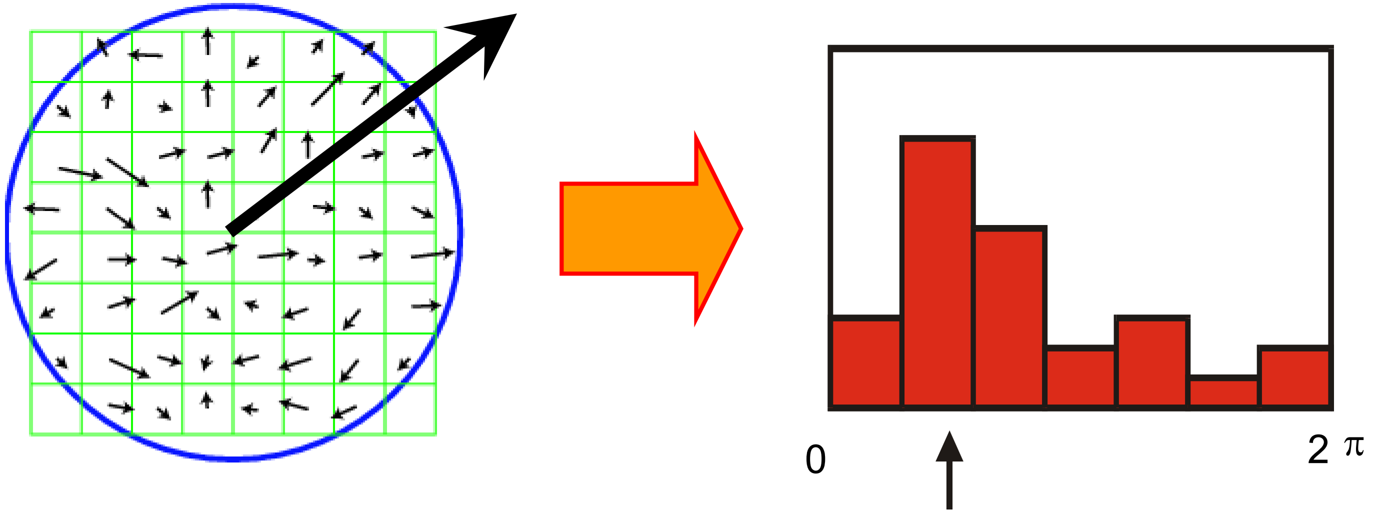


Image gradients

Adapted from slide by David Lowe

Finding a reference orientation

Assign reference orientation at peak of smoothed histogram

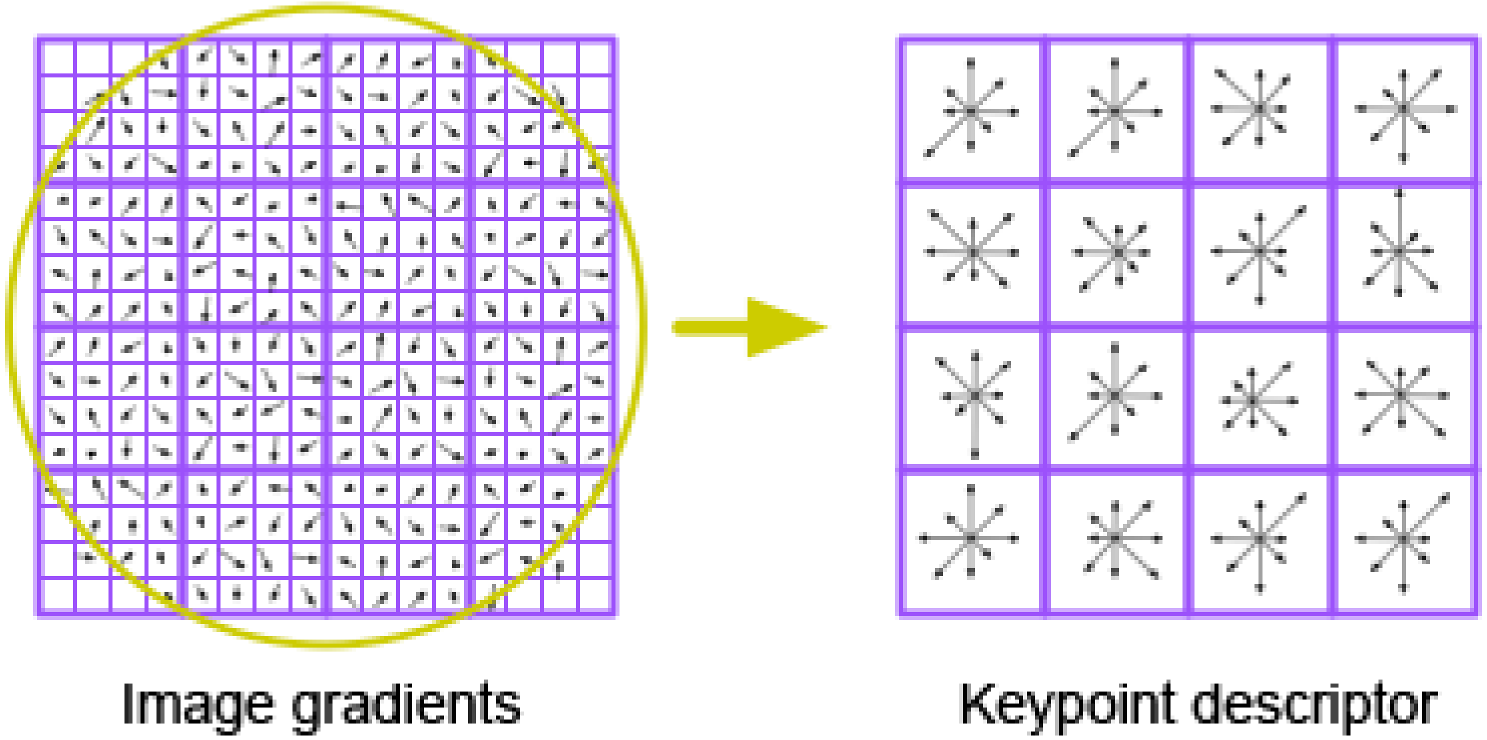


Source: Svetlana Lazebnik

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



What about 3D rotations?

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

Source: Svetlana Lazebnik

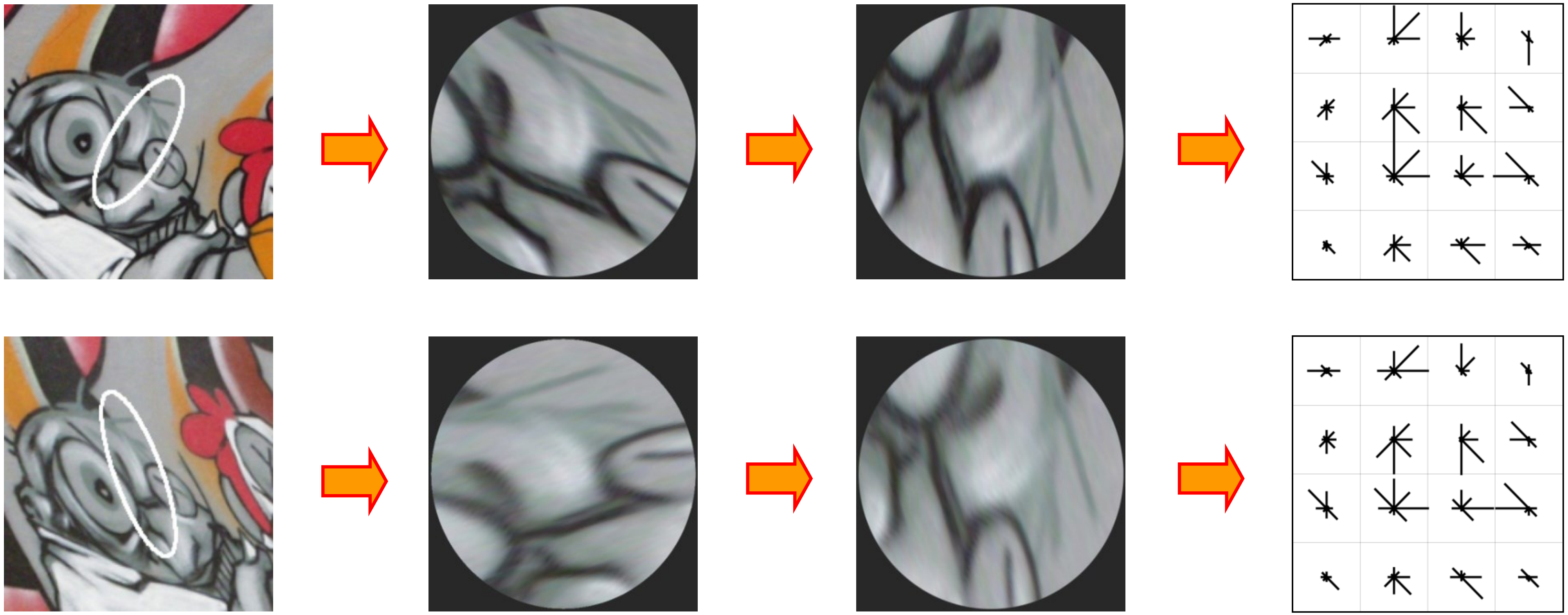
Summary: Affine-Inv. Feature Extraction

Extract affine regions

Normalize regions

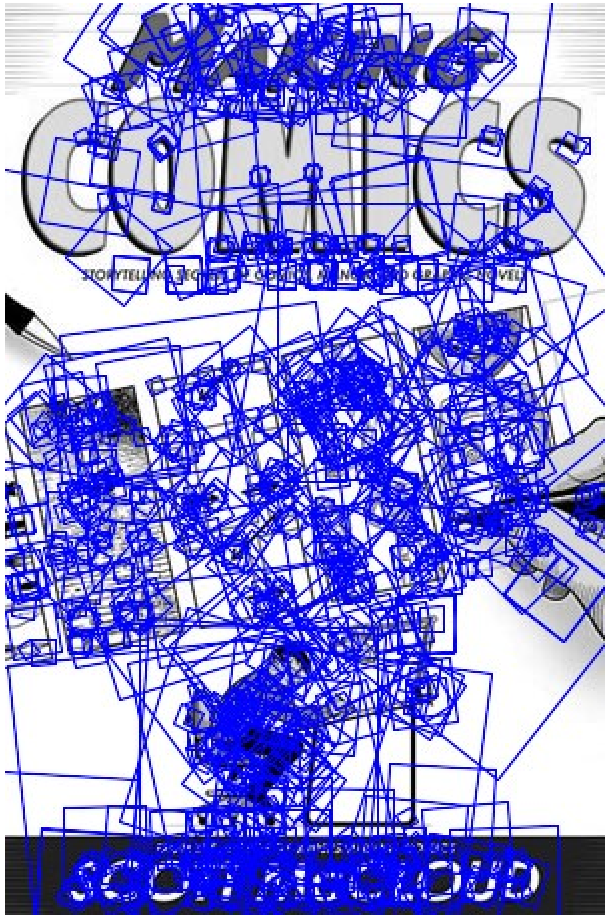
Eliminate rotational ambiguity

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):

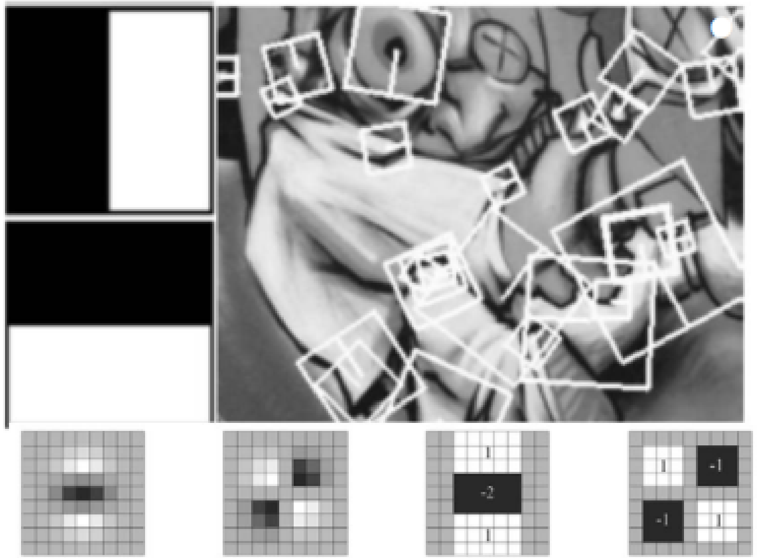
- 1999 Scale Invariant Feature Transform (Lowe)
- 2006 Speeded Up Robust Features (Bay, Tuytelaars, Van Gool)

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)

Source: Stefan Haller

SURF Speeded Up Robust Features



Fast approximation of SIFT idea

- Efficient computation by 2D box filters & integral images
  $\Rightarrow$ 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/

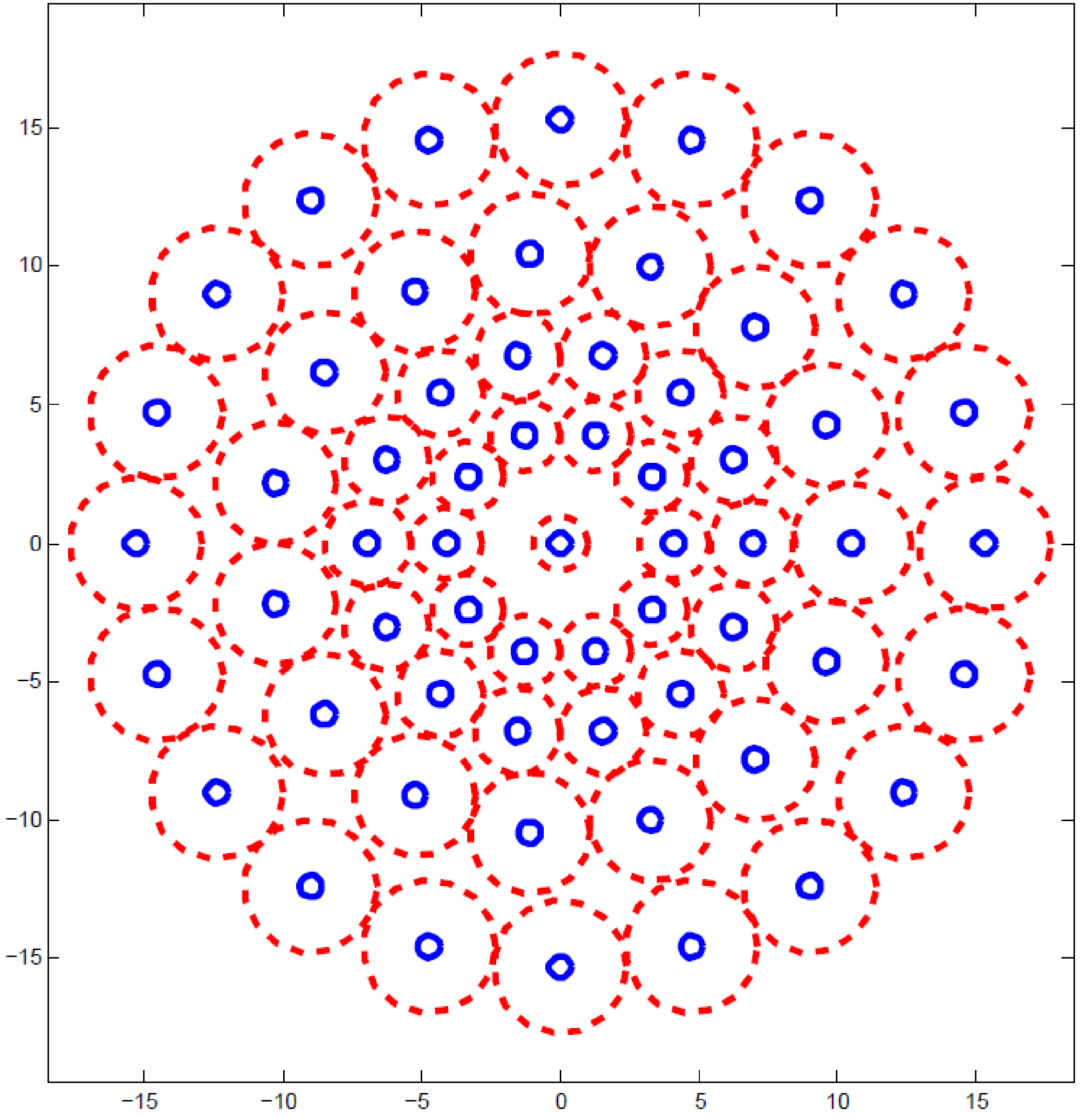
[Bay, ECCV'06], [Cornelis, CVGPU'08]

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

Source: Fei-Fei Li

BRISK – Sampling pattern

- Number of samples: N = 60
- Points $\boldsymbol{p_i}$ are evenly separated around concentric circles.
- $I(\boldsymbol{p_i}, \sigma_i)$: Intensity of points smoothed with Gaussian proportional to separation of points on circle.
- i.e., points on each concentric circle use a different kernel.
  - Avoids aliasing effects
- Size of pattern based on scale (pattern shown for *t=1*)



Source: Stefan Haller

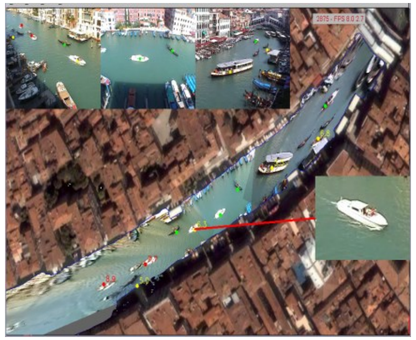
BRISK – Example

```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)
```

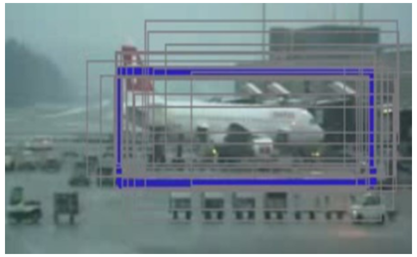
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

Corso di Sistemi Informativi
A.A. 2018/19

Docente
Domenico Daniele Bloisi



Feature Descriptors



Aprile 2019