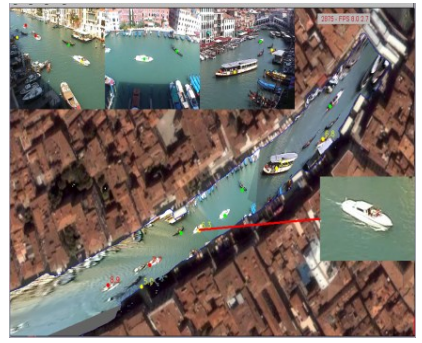
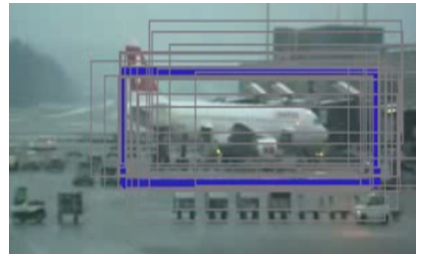
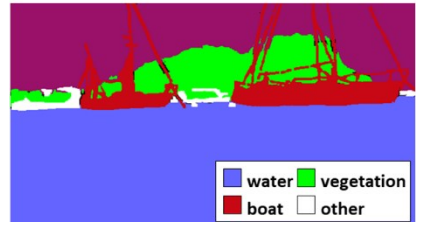
UNIVERSITÀ DEGLI STUDI DELLA BASILICATA



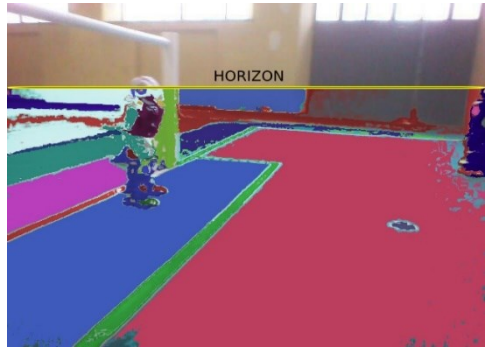




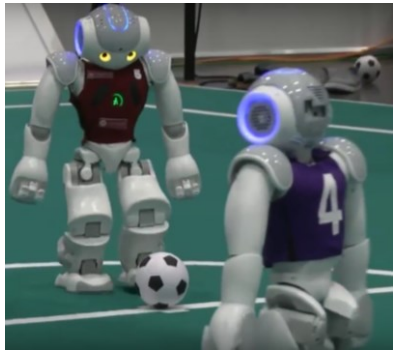


Corso di Visione e Percezione
A.A. 2019/2020

Docente
Domenico Daniele Bloisi

Omografie




Aprile 2020

Il corso

- Home page del corso
  http://web.unibas.it/bloisi/corsi/visione-e-percezione.html

- Docente: Domenico Daniele Bloisi

- Periodo: II 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 7: Transformations and warping“
  - M. Brown and D. G. Lowe
    Recognising Panoramas

- I contenuti fanno riferimento al capitolo 3 del libro
  "Computer Vision: Algorithms and Applications"
  di Richard Szeliski, disponibile al seguente indirizzo
  http://szeliski.org/Book/

Image alignment

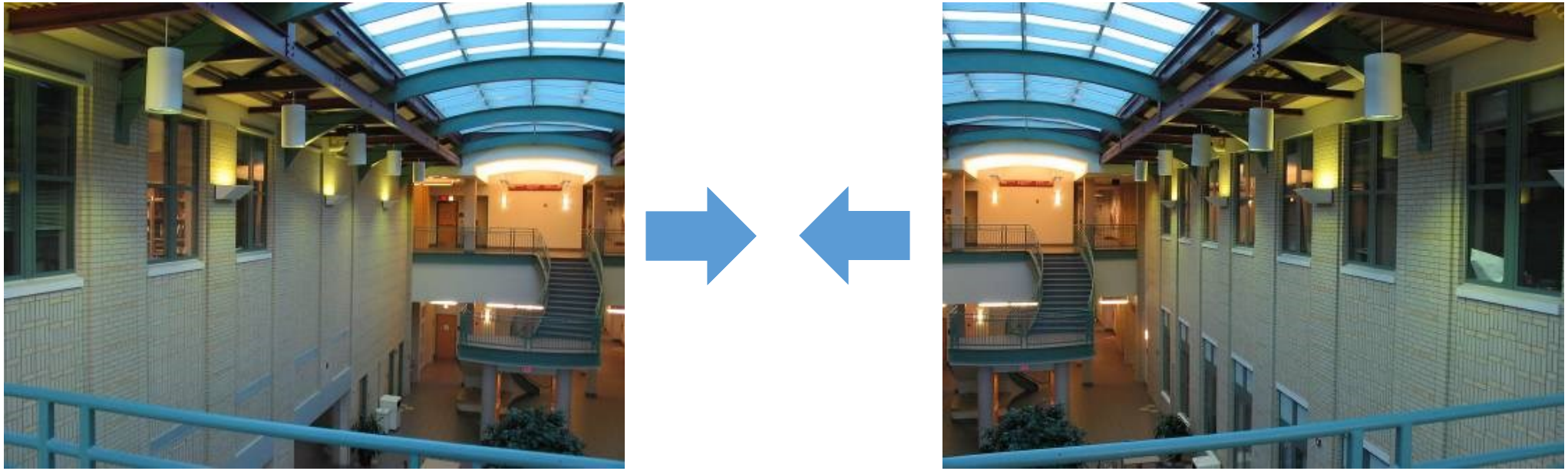


Image alignment

Similarity

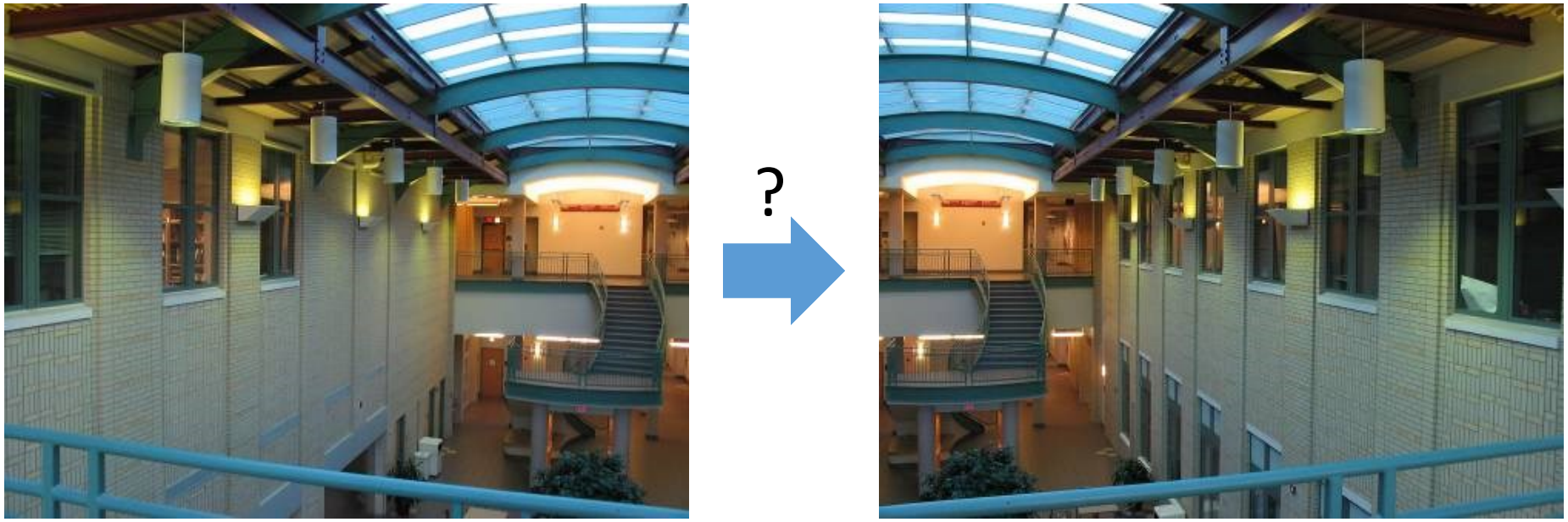
What is the geometric relationship between these two images?

?

Answer: Similarity transformation (translation, rotation, uniform scale)

Similarity?

What is the geometric relationship between these two images?



Similarity?

Very important for creating mosaics!

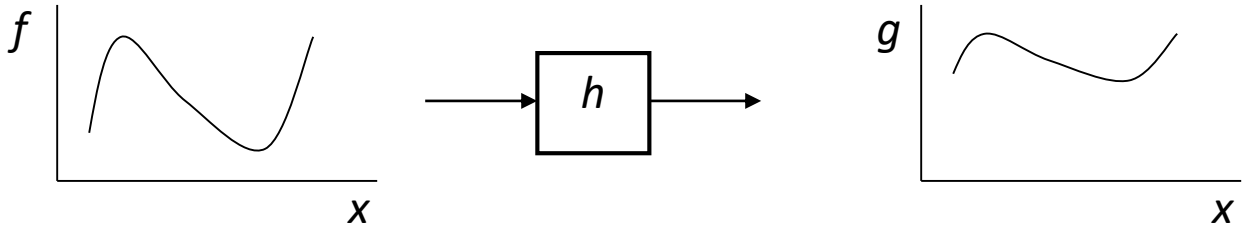
First, we need to know what this transformation is.

Second, we need to figure out how to compute it using feature matches.

Image Warping

- image filtering: change *range* of image

$$g(x) = h(f(x))$$



- image warping: change *domain* of image

$$g(x) = f(h(x))$$

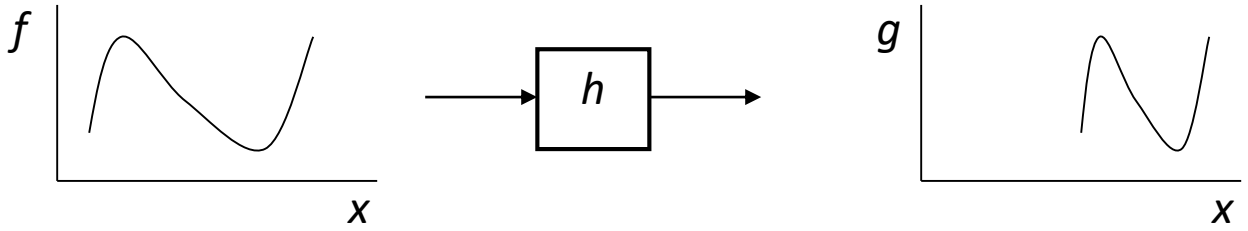
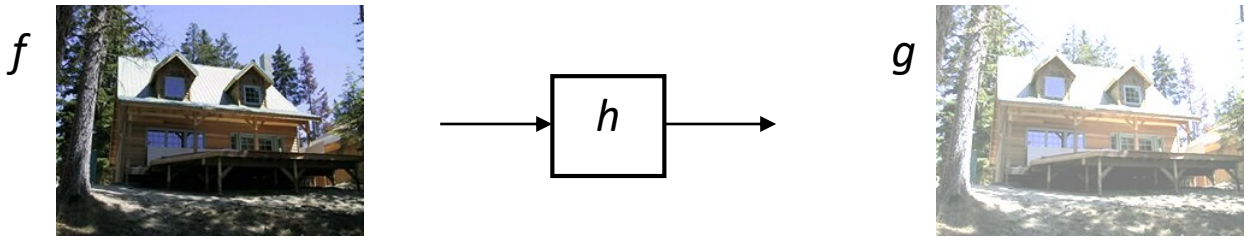


Image Warping

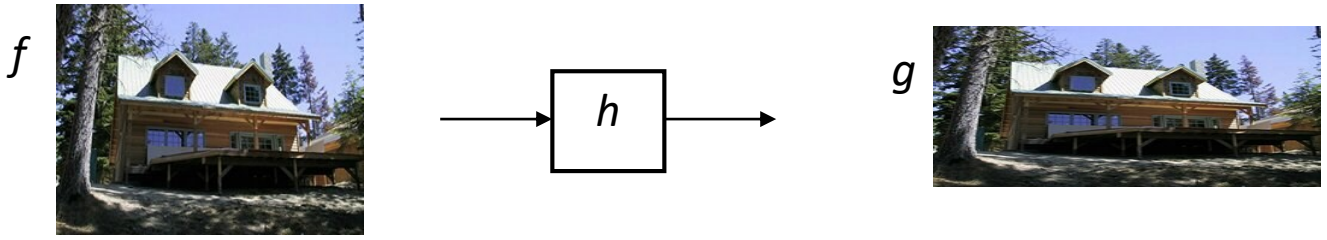
- image filtering: change *range* of image

$$g(x) = h(f(x))$$



- image warping: change *domain* of image

$$g(x) = f(h(x))$$



Parametric (global) warping

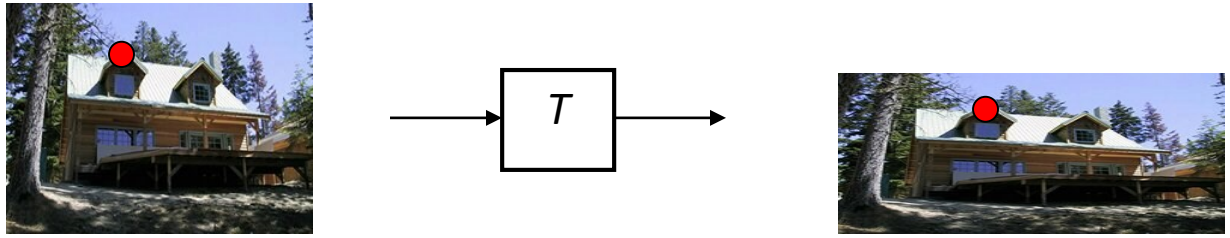
Examples of parametric warps:

translation

rotation

aspect

Parametric (global) warping



p = (x,y)

p’ = (x’,y’)

Transformation T is a coordinate-changing machine:

p’ = *T*(**p**)

What does it mean that *T* is global?

- is the same for any point **p**
- can be described by just a few numbers (parameters)

Linear transforms

Let's consider *linear* transforms

(can be represented by a 2x2 matrix):

$$\mathbf{p}' = \mathbf{T}\mathbf{p}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

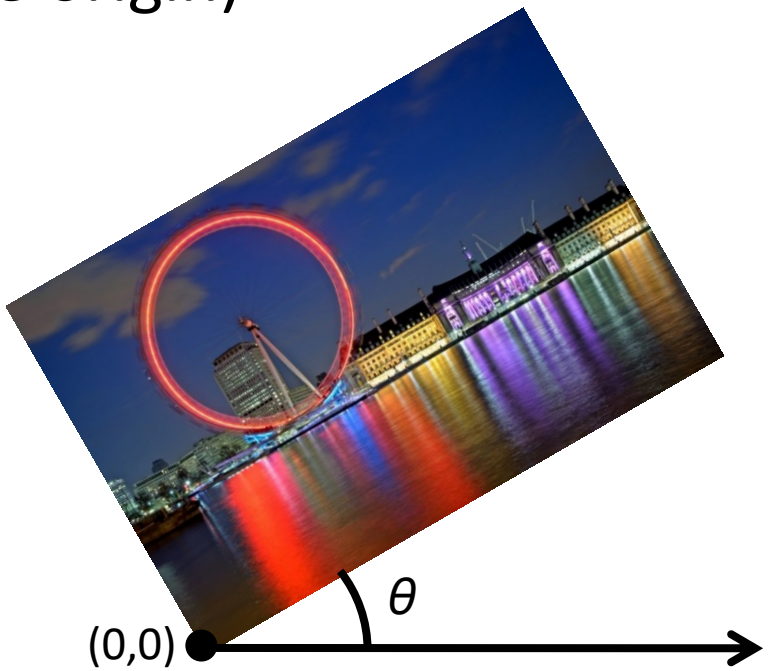
Scaling

Uniform scaling by *s*:

$$\mathbf{S} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

Rotation

Rotation by angle θ (about the origin)



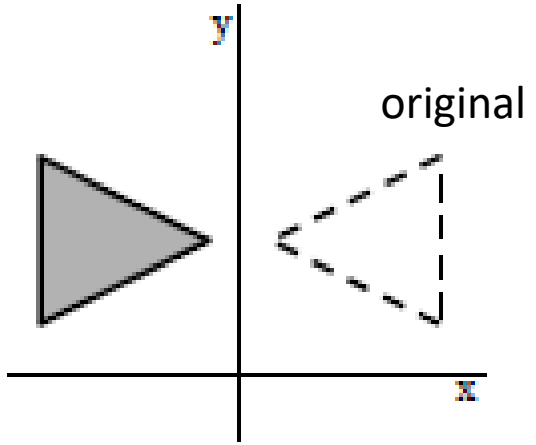
$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

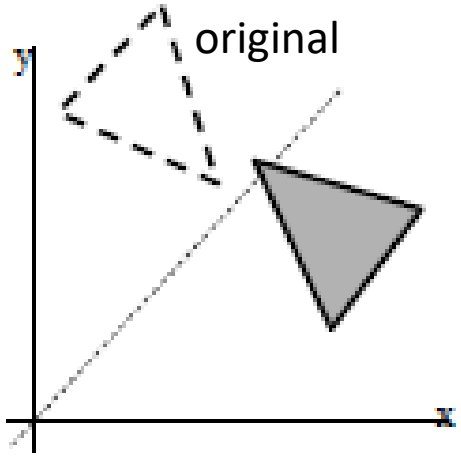
2D mirror about Y axis?

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$



2D mirror across line y = x?

$$\begin{aligned} x' &= y \\ y' &= x \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$



2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$x' = x + t_x$$
$$y' = y + t_y$$

NO!

Translation is not a linear operation on 2D coordinates

All 2D Linear Transformations

Linear transformations are combinations of …

- Scale,
- Rotation,
- Shear, and
- Mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

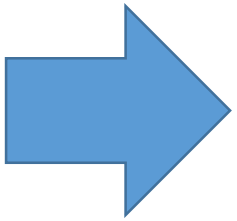
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Homogeneous Coordinates

How can we represent translation

$$x' = x + t_x$$
$$y' = y + t_y$$

as a 3x3 matrix?



$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{homogeneous coords}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

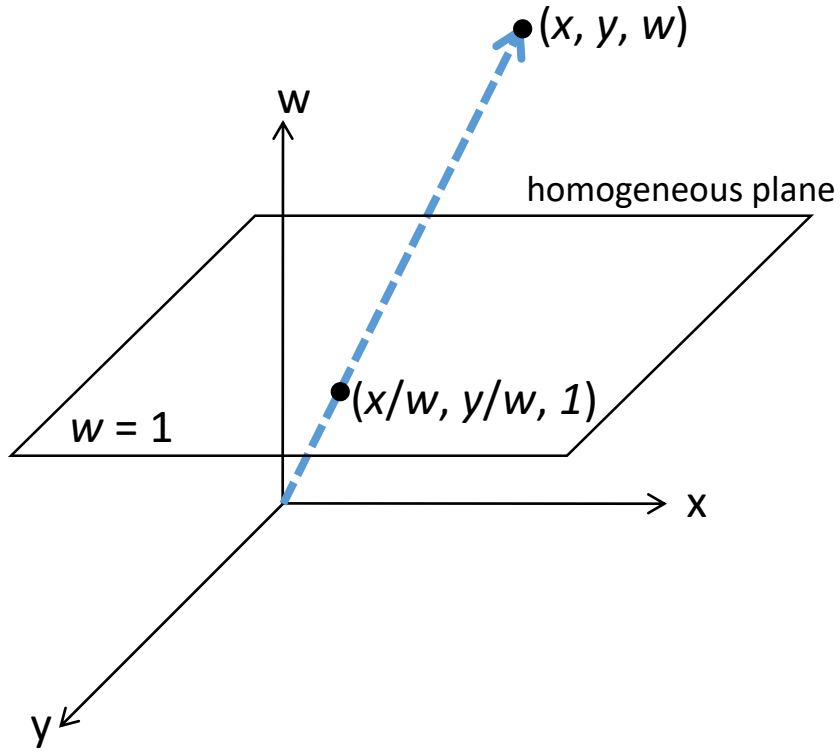
represent coordinates in 2 dimensions with a 3-vector

Homogeneous coordinates

Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates



Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Translation

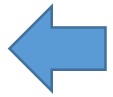
Solution using homogeneous coordinates

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

Affine transformations

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



any transformation represented by a 3x3 matrix with last row [0 0 1] we call an *affine transformation*

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

Affine transformations

Affine transformations are combinations of …

- Linear transformations, and
- Translations

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of affine transformations:

- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

Basic affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

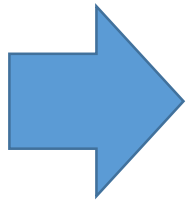
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D *in-plane* rotation

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

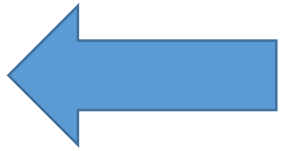
É una trasformazione affine?



Trasformazioni non affini

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

affine transformation



Cosa accade se cambiamo gli elementi della terza riga?

Omografie

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

H is a *homography* (or *planar perspective map*)

Punti all’infinito

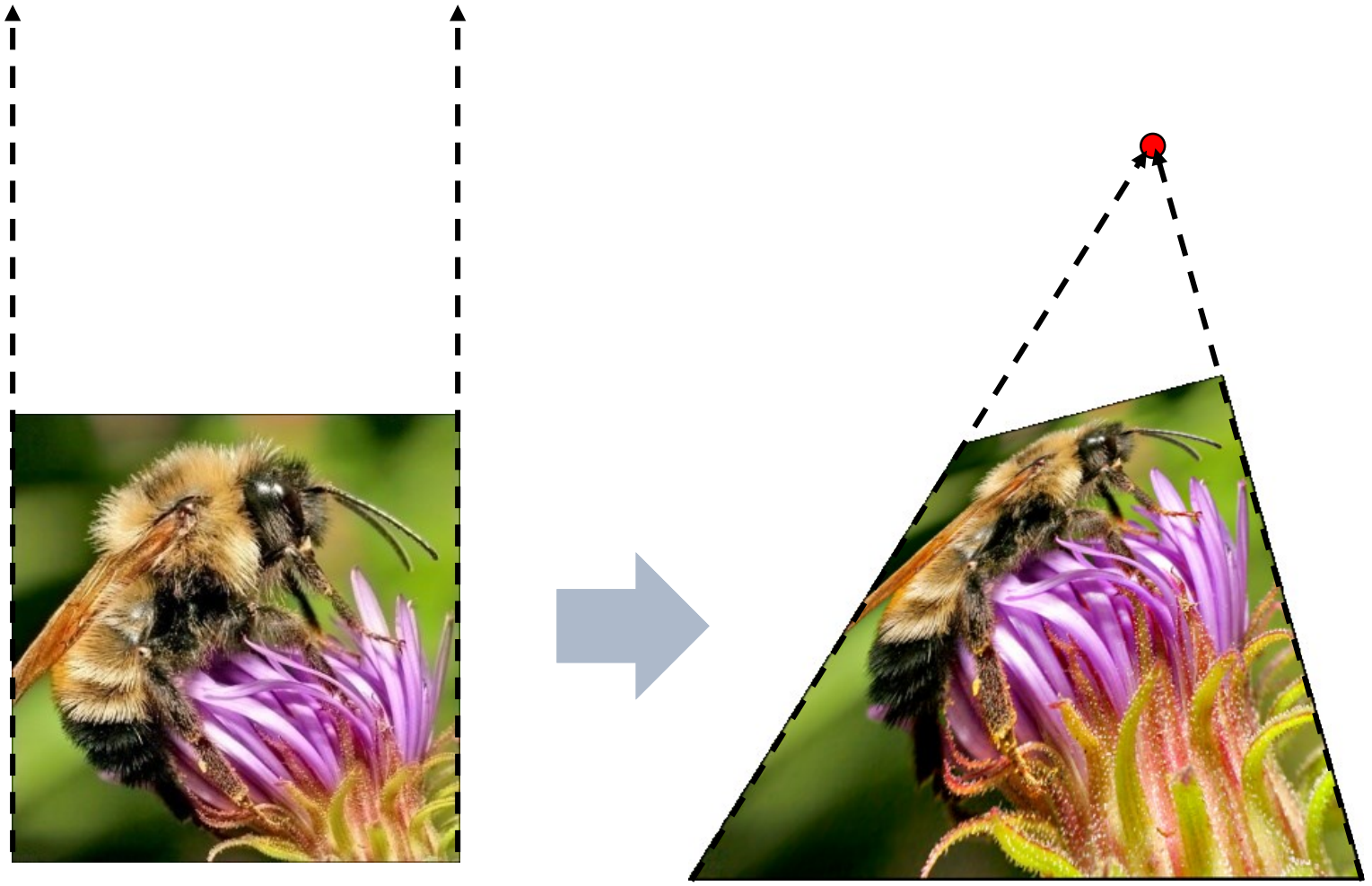


Image warping con omografie

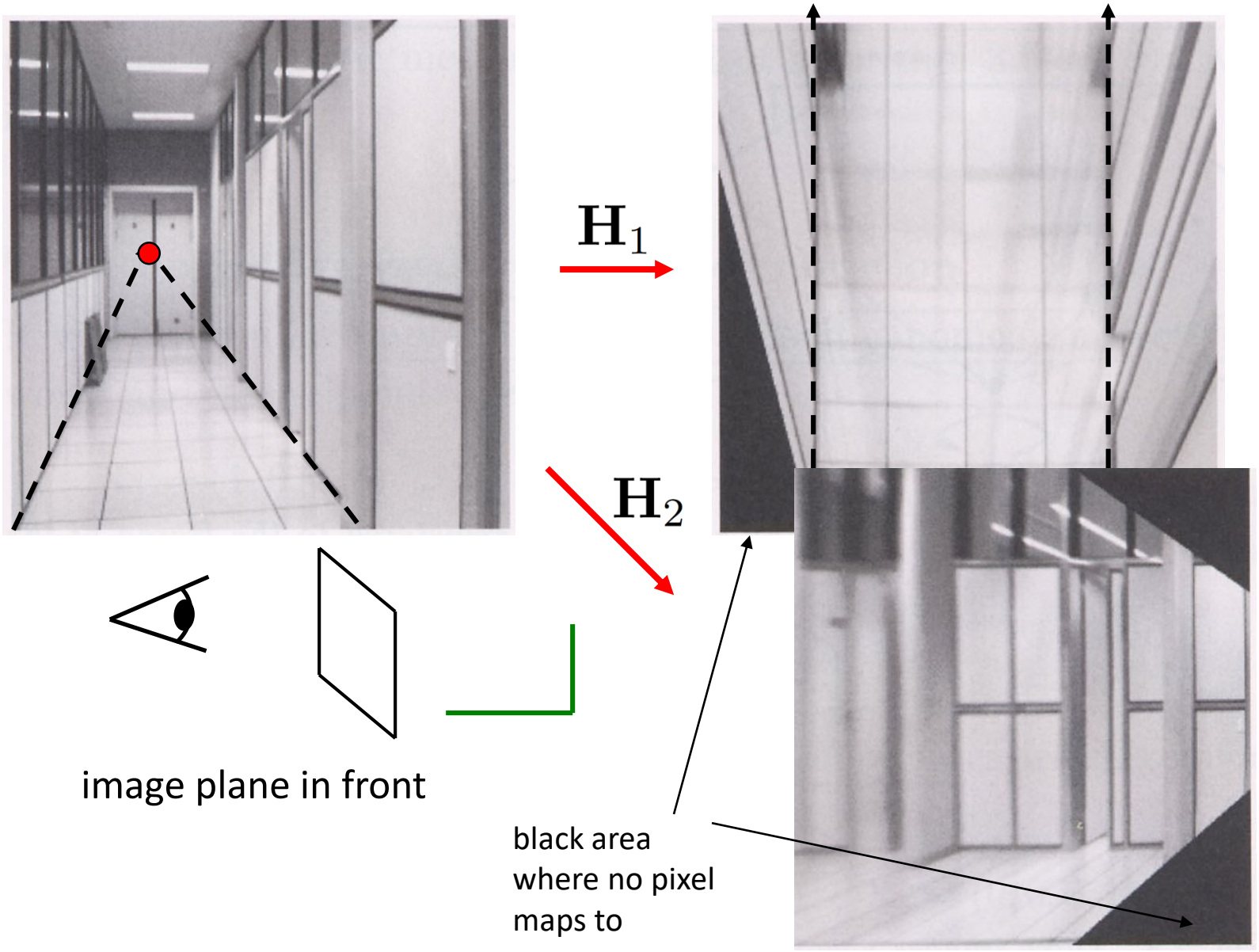
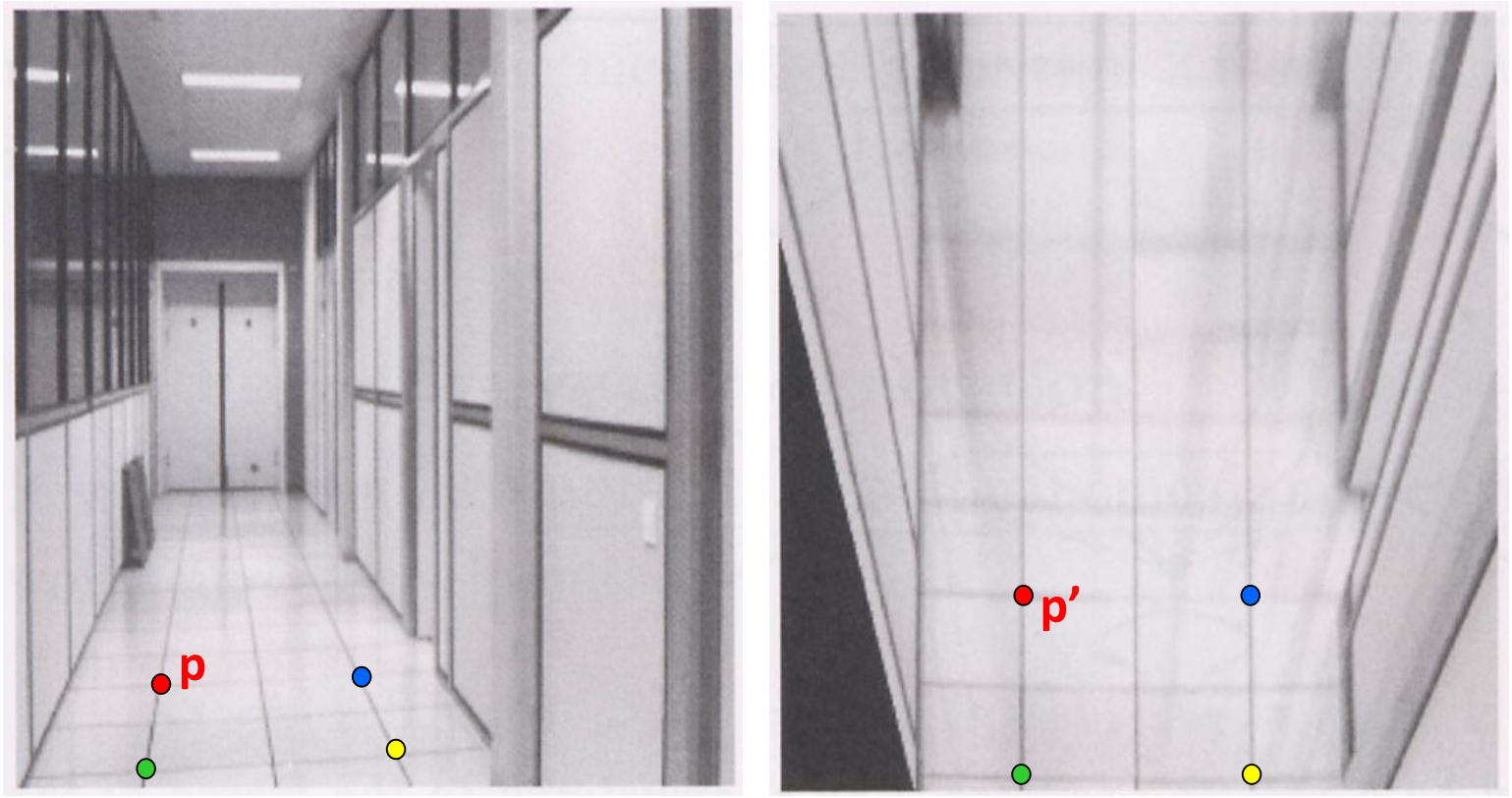
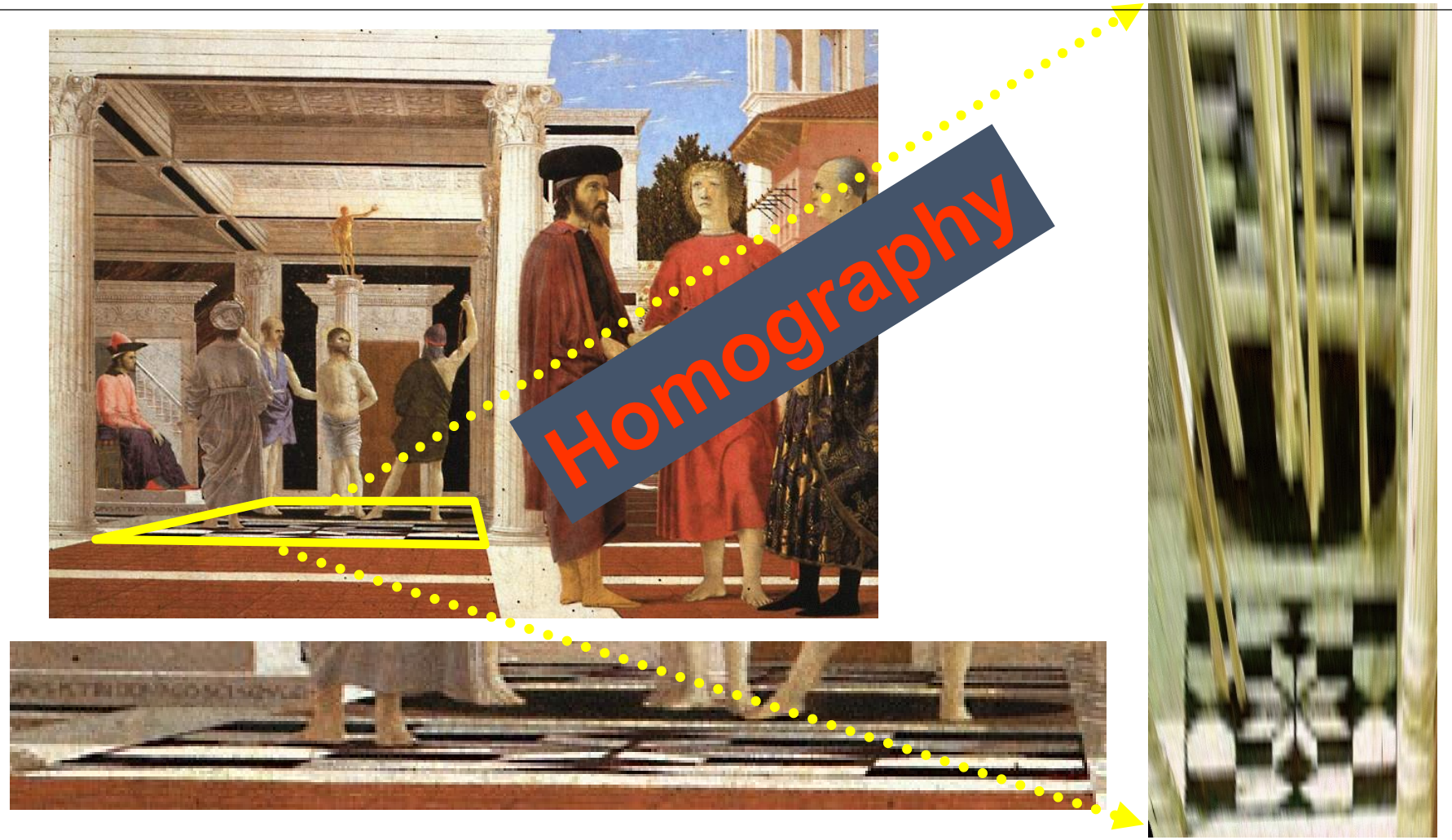


Image warping con omografie



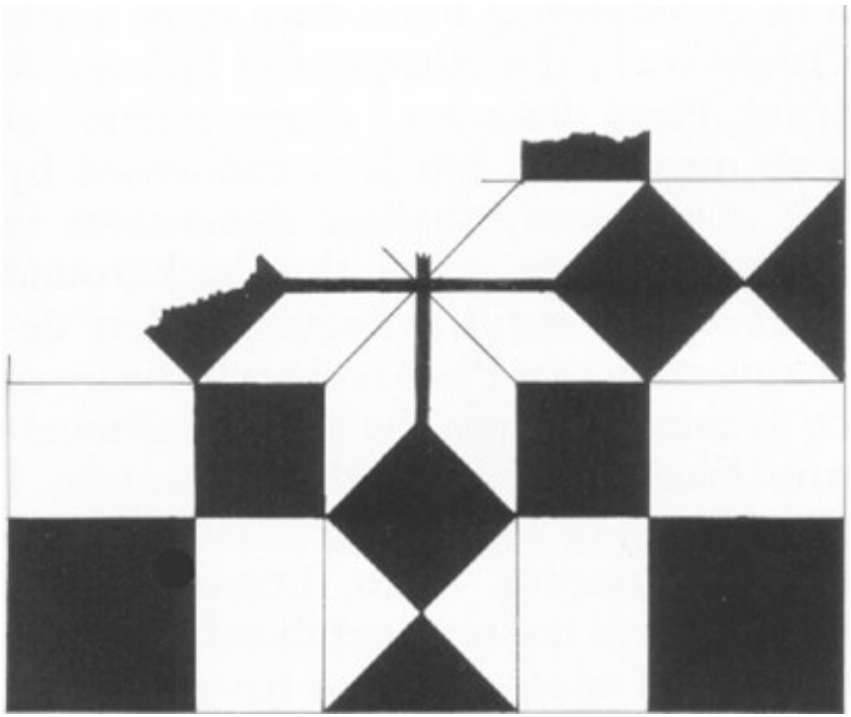
What is the shape of the b/w floor pattern?



Slide from Criminisi

Analyzing patterns and shapes

Automatic rectification



From Martin Kemp *The Science of Art*
(manual reconstruction)

2 patterns have been discovered !

Slide from Criminisi

Analyzing patterns and shapes

***St. Lucy Altarpiece,* D. Veneziano**

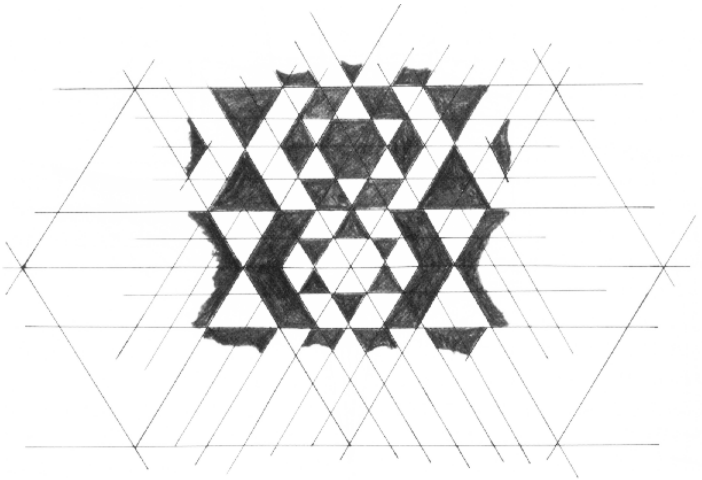
What is the (complicated) shape of the floor pattern?

Automatically rectified floor

Slide from Criminisi

Analyzing patterns and shapes

Automatic rectification



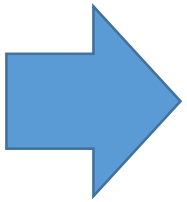
From Martin Kemp, *The Science of Art*
(manual reconstruction)

Slide from Criminisi

Analyzing patterns and shapes

The Ambassadors by Hans Holbein the Younger, 1533

É una omografia



Omografie

Homographies …

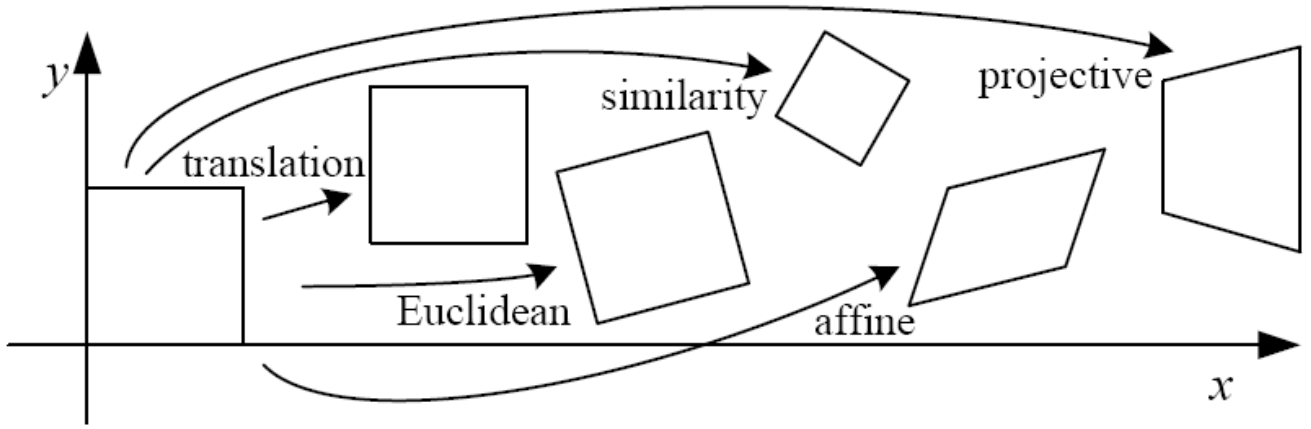
- Affine transformations, and
- Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of projective transformations:

- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition

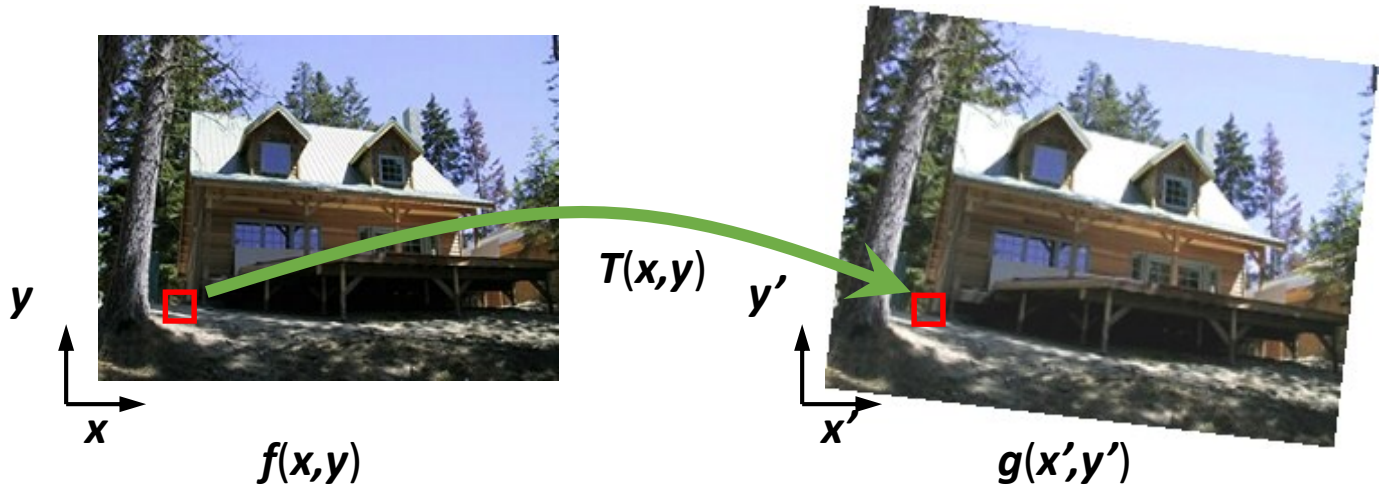
Ricapitolando



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times3}$	8	straight lines	

Implementing image warping

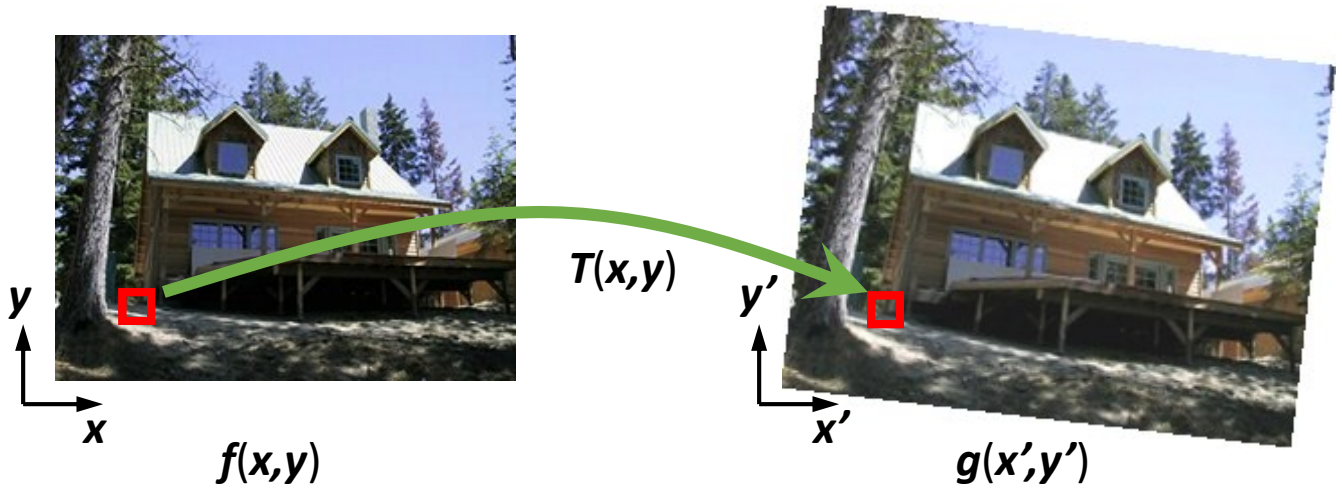
Given a coordinate transform $(x',y') = T(x,y)$ and a source image $f(x,y)$, how do we compute a transformed image $g(x',y') = f(T(x,y))$?



Forward Warping

Send each pixel $f(x)$ to its corresponding location
$(x',y') = T(x,y)$ in $g(x',y')$

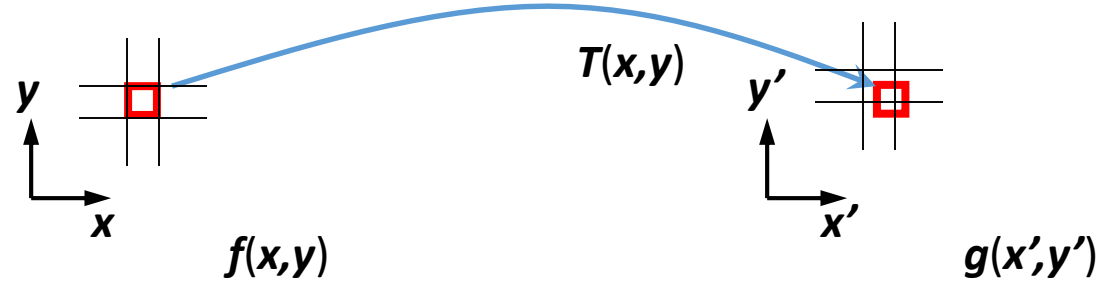
- What if pixel lands "between" two pixels?



Forward Warping

Send each pixel $f(x)$ to its corresponding location
$(x',y') = T(x,y)$ in $g(x',y')$

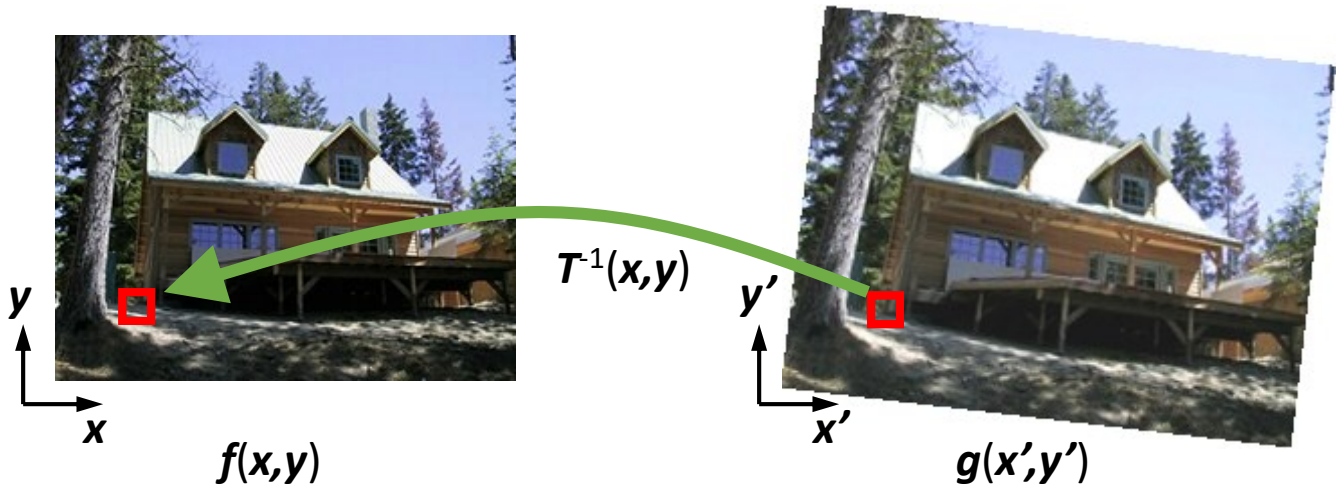
- What if pixel lands "between" two pixels?
- Answer: add "contribution" to several pixels, normalize later (splatting)
- Can still result in holes



Inverse Warping

Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x,y)$ in $f(x,y)$

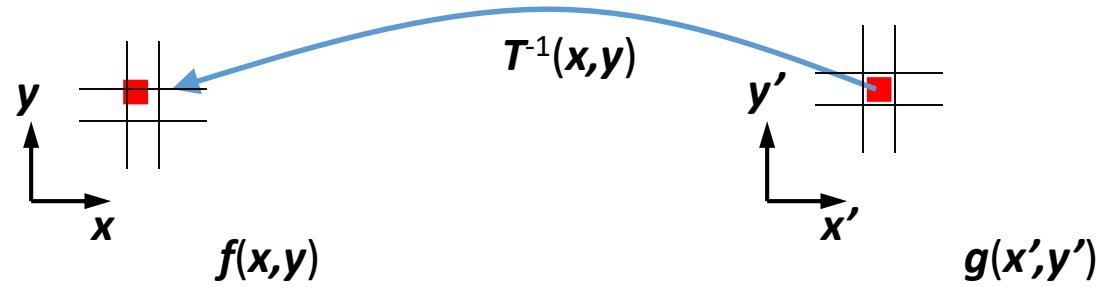
- Requires taking the inverse of the transform
- What if pixel comes from "between" two pixels?



Inverse Warping

Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$

- What if pixel comes from "between" two pixels?
- Answer: *resample* color value from *interpolated* (*prefiltered*) source image



Interpolation

Possible interpolation filters:

- nearest neighbor
- bilinear
- bicubic
- sinc

Needed to prevent “jaggies” and “texture crawl”

(with prefiltering)

Forward vs. inverse warping

Q: Which is better?

A: usually inverse—eliminates holes

- however, it requires an invertible warp function—not always possible...

Esempio omografia+warping

```
import cv2 as cv
from google.colab.patches import cv2_imshow
from urllib.request import urlopen
import numpy as np

req_left = urlopen('https://dbloisi.github.io/corsi/images/montagna-1.jpg')
arr_left = np.array(bytearray(req_left.read()), dtype=np.uint8)
img_left = cv.imdecode(arr_left, -1)
cv2_imshow(img_left)
```

Esempio omografia+warping

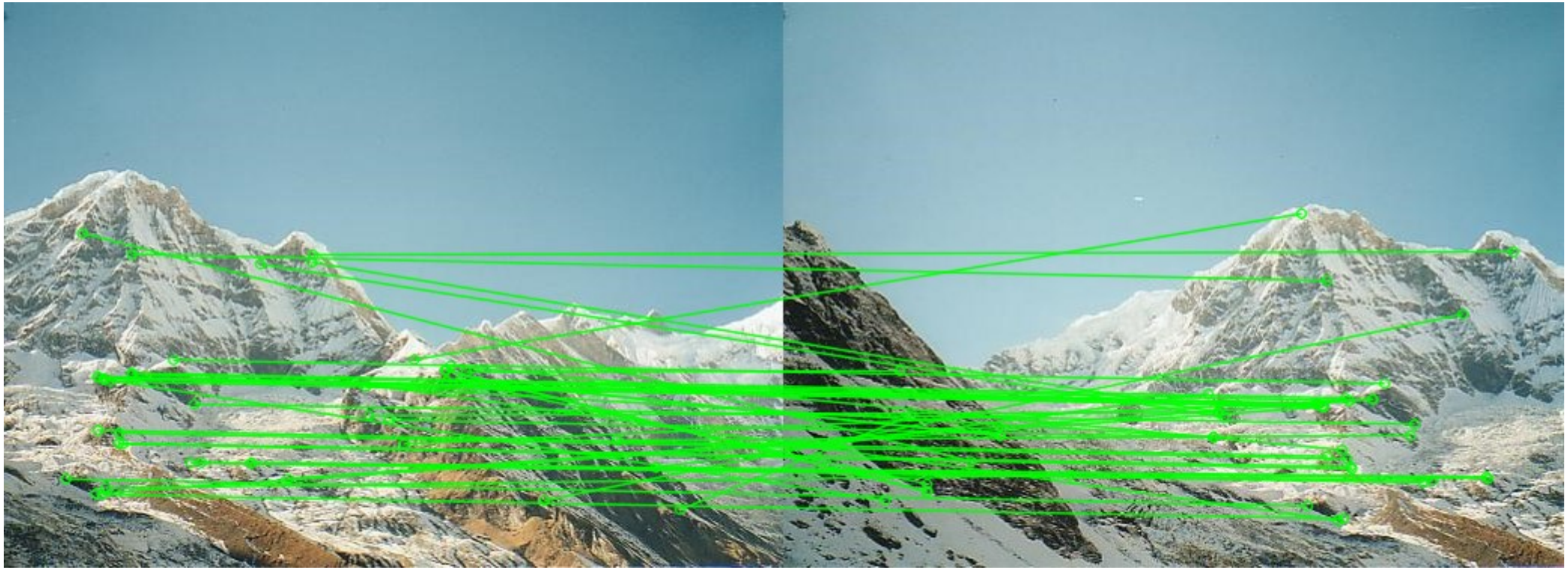
```
req_right = urlopen('https://dbloisi.github.io/corsi/images/montagna-2.jpg')
arr_right = np.array(bytearray(req_right.read()), dtype=np.uint8)
img_right = cv.imdecode(arr_right, -1)
cv2_imshow(img_right)
```

Esempio omografia+warping

```python
# orb descriptor
orb = cv.ORB_create()
# find key points
kp1, des1 = orb.detectAndCompute(img_right, None)
kp2, des2 = orb.detectAndCompute(img_left, None)
# brute force matching
match = cv.BFMatcher()
matches = match.knnMatch(des1,des2,k=2)
# distance ratio
# "Distinctive Image Features from Scale-Invariant Keypoints"
# by David G. Lowe
good = []
for m,n in matches:
    if m.distance < 0.85*n.distance:
        good.append(m)

# drawing good matches
draw_params = dict(matchColor=(0,255,0),
                   singlePointColor=None,
                   flags=2)
matches_img = cv.drawMatches(img_right,kp1,img_left,kp2,good,None,**draw_params)
cv2_imshow(matches_img)
```

Esempio omografia+warping

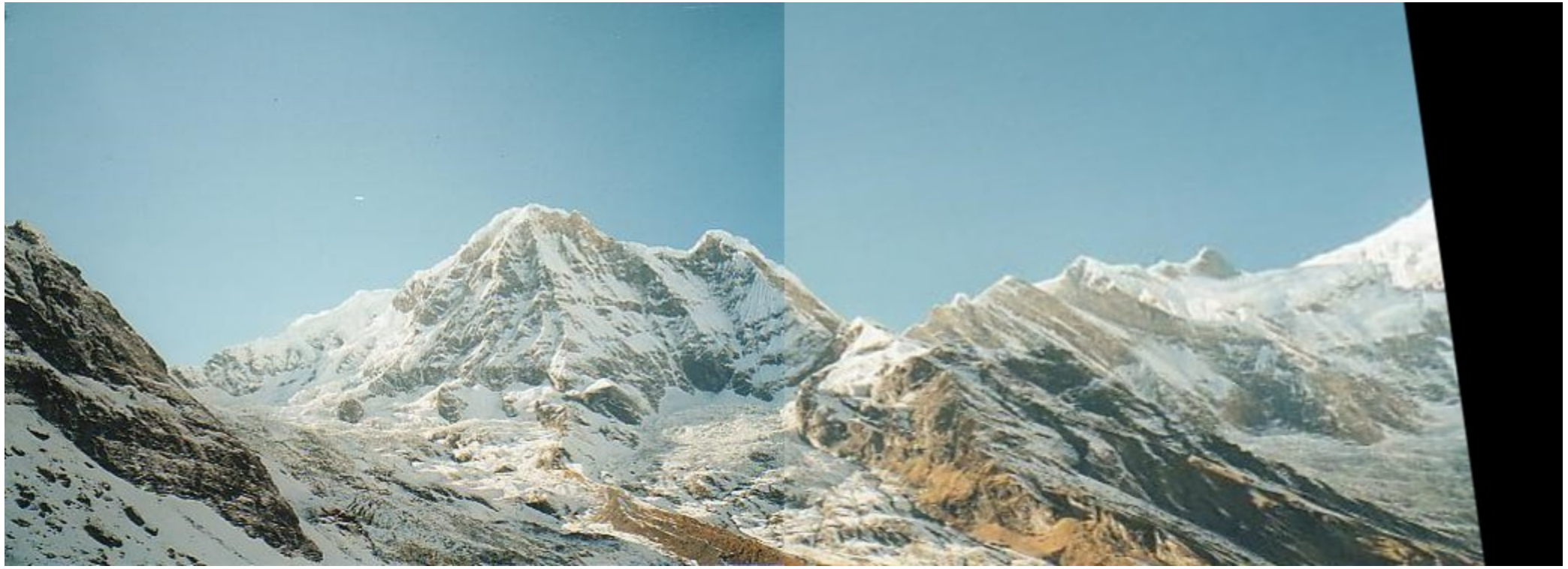


Esempio omografia+warping

```python
# homography computation
MIN_MATCH_COUNT = 5
if len(good) > MIN_MATCH_COUNT:
    src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2)
    dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)
    M, mask = cv.findHomography(src_pts, dst_pts, cv.RANSAC, 5.0)
    h,w,c = img_right.shape
    pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
    dst = cv.perspectiveTransform(pts, M)
else:
    print("Not enought matches are found - %d/%d", (len(good)/MIN_MATCH_COUNT))
```

Esempio omografia+warping

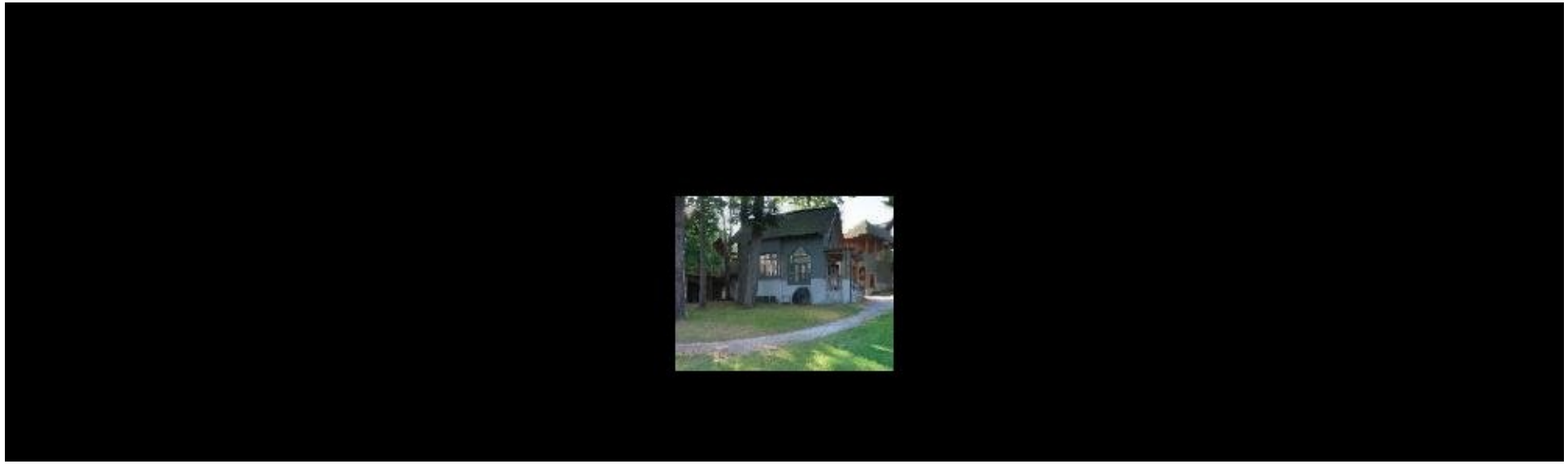
```
# creating panorama image
panorama_img = cv.warpPerspective(img_right,M,(img_left.shape[1] + img_right.shape[1], img_left.shape[0]))
panorama_img[0:img_left.shape[0],0:img_left.shape[1]] = img_left
cv2_imshow(panorama_img)
```



Panoramas

Are you getting the whole picture?

Compact Camera FOV = 50 x 35°

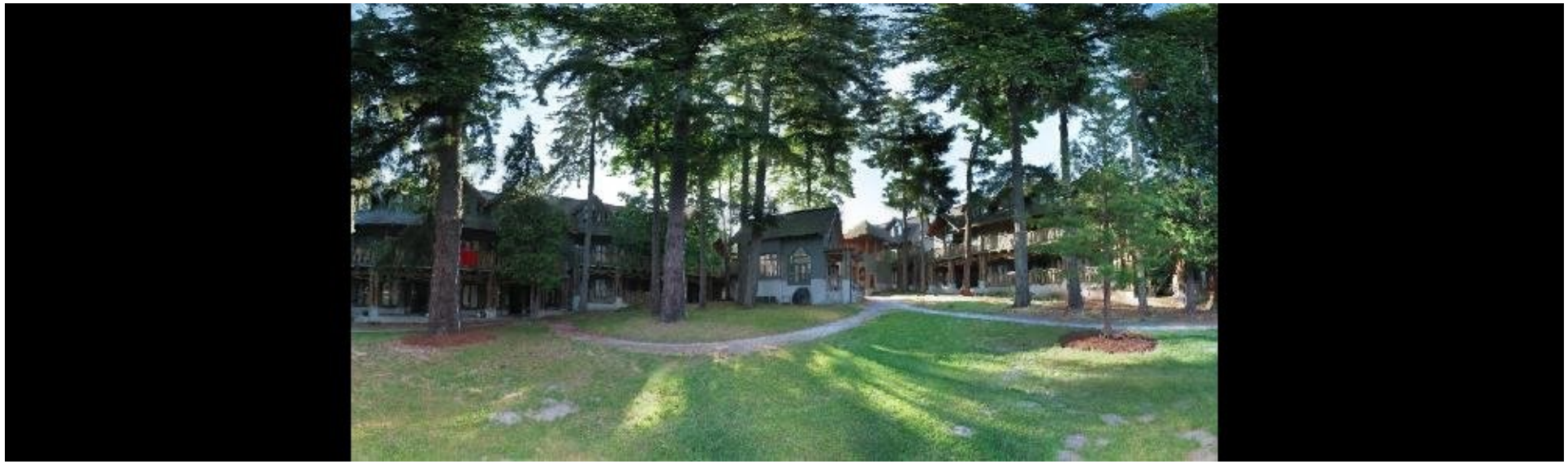


Panoramas

Are you getting the whole picture?

Compact Camera FOV = 50 x 35°

Human FOV = 200 x 135°



Panoramas

Are you getting the whole picture?

Compact Camera FOV = 50 x 35°

Human FOV = 200 x 135°

Panoramic Mosaic = 360 x 180°

Why “Recognising Panoramas”?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

Why “Recognising Panoramas”?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

Why “Recognising Panoramas”?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

- 2D Rotations (θ, ϕ)
  - Ordering $\not\Rightarrow$ matching images

Why “Recognising Panoramas”?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

- 2D Rotations (θ, ϕ)
  - Ordering $\not\Rightarrow$ matching images

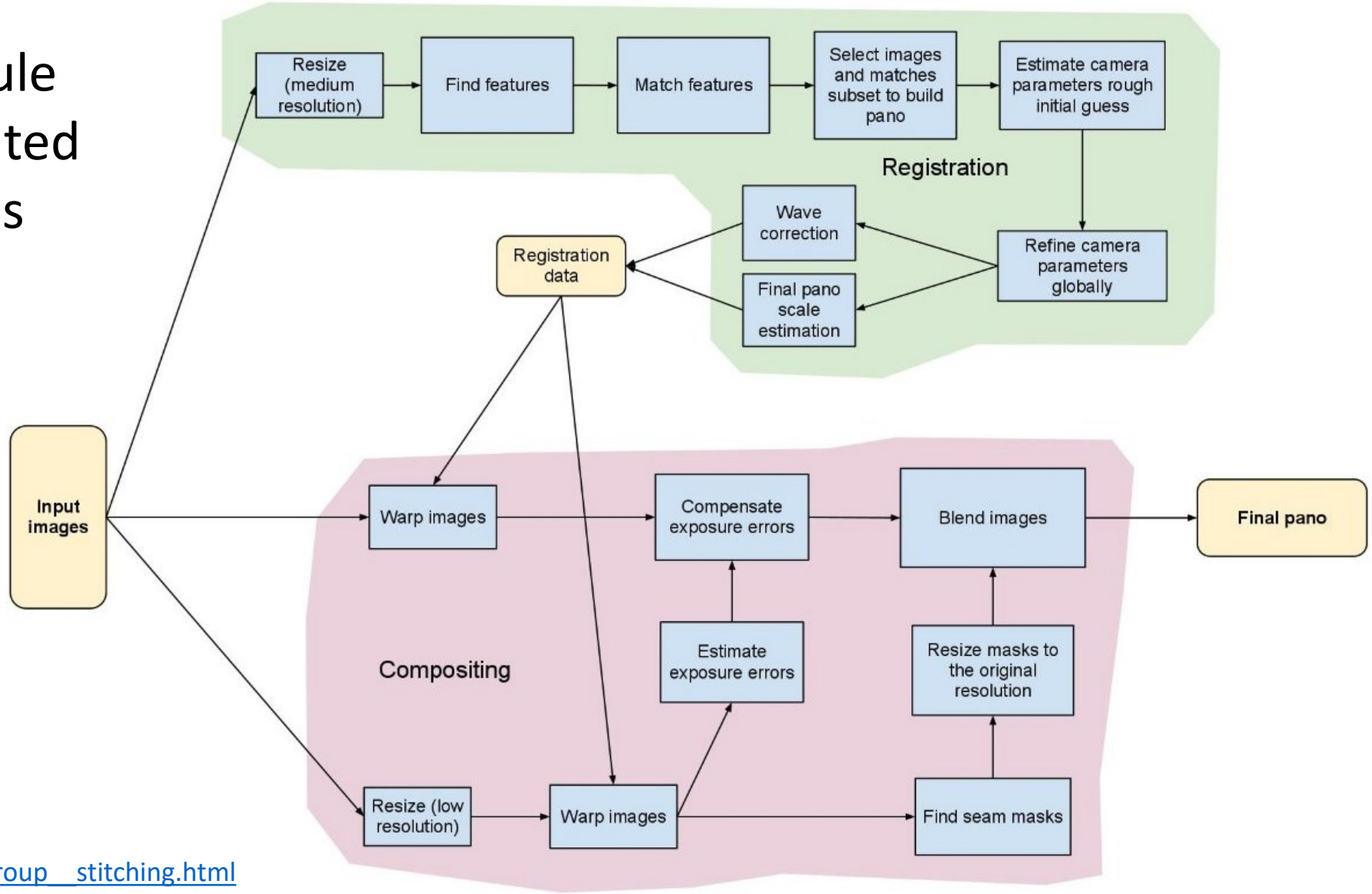
Why “Recognising Panoramas”?

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

- 2D Rotations (θ, ϕ)
  - Ordering $\not\Rightarrow$ matching images

Stitching in OpenCV

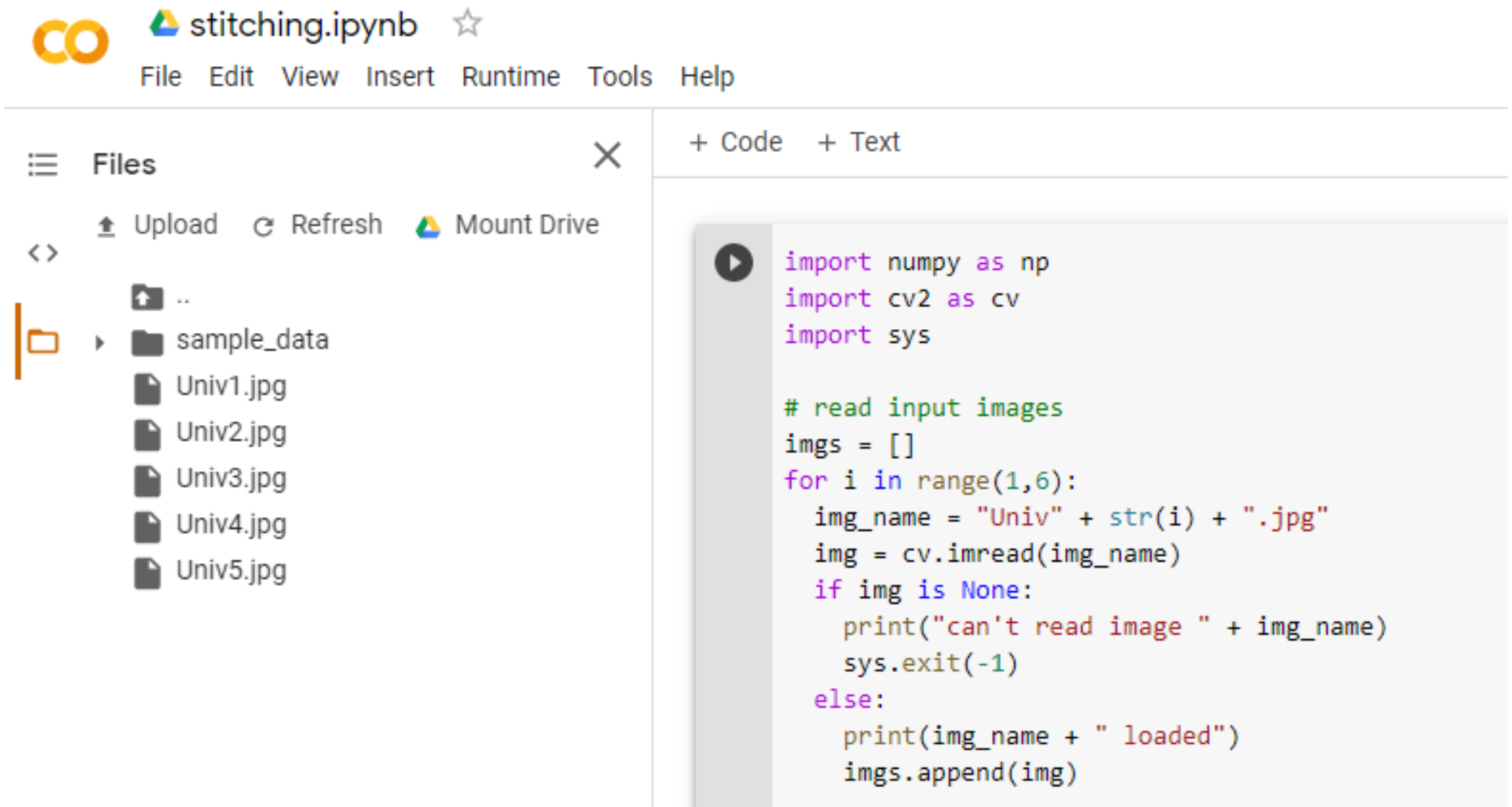
The stitching module pipeline implemented in the **Stitcher** class



https://docs.opencv.org/4.1.2/d1/d46/group__stitching.html

Stitching in OpenCV

Stitching in OpenCV



https://docs.opencv.org/4.1.2/d5/d48/samples_2python_2stitching_8py-example.html

Stitching in OpenCV

```
stitcher = cv.Stitcher.create(cv.Stitcher_PANORAMA)

status, pano = stitcher.stitch(imgs)

if status != cv.Stitcher_OK:
  print("Can't stitch images, error code = %d" % status)
  sys.exit(-1)

cv.imwrite("panorama.jpg", pano)

print("stitching completed successfully.")
```

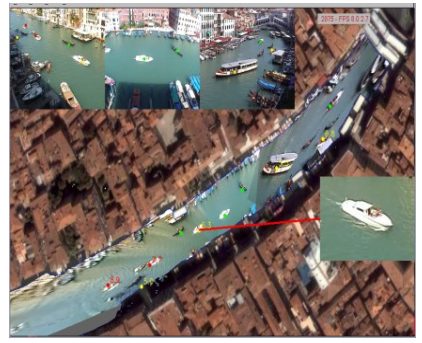
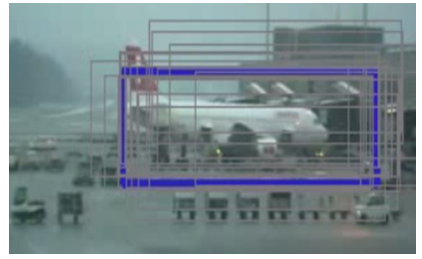
```
stitching completed successfully.
```

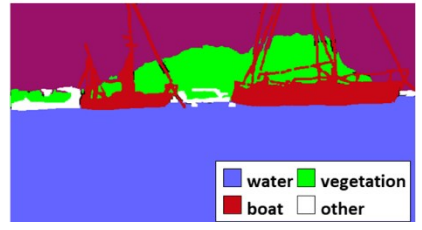
https://docs.opencv.org/4.1.2/d5/d48/samples_2python_2stitching_8py-example.html

Stitching in OpenCV

UNIVERSITÀ DEGLI STUDI DELLA BASILICATA



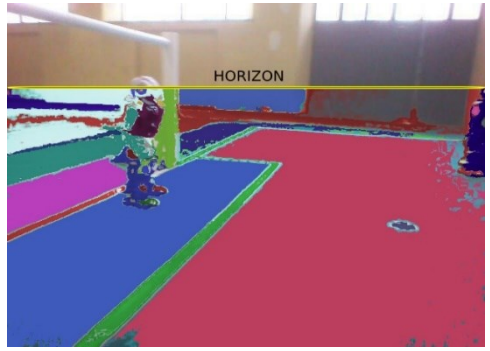




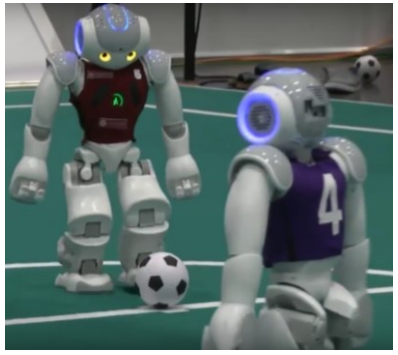


Corso di Visione e Percezione
A.A. 2019/2020

Docente
Domenico Daniele Bloisi

Omografie




Aprile 2020