Movement and deformation analysis from images

OpenCV, Python

Z. Siki, B. Takács

21 April 2022

Automated Survey System
BME, Department of Geodesy and Surveying

Tutorials

OpenCV
https://docs.opencv.org/4.5.5/
OpenCV
https://www.programcreek.com/python/example/89361/cv2.Canny
OpenCV
http://www.bmva.org/bmvc/1989/avc-89-029.pdf
NumPy
https://github.com/OSGeoLabBp/tutorials/blob/master/english/python/numpy_tutor.ipynb
Further exercises
https://github.com/OSGeoLabBp/tutorials/tree/master/english/img_processing

Principles

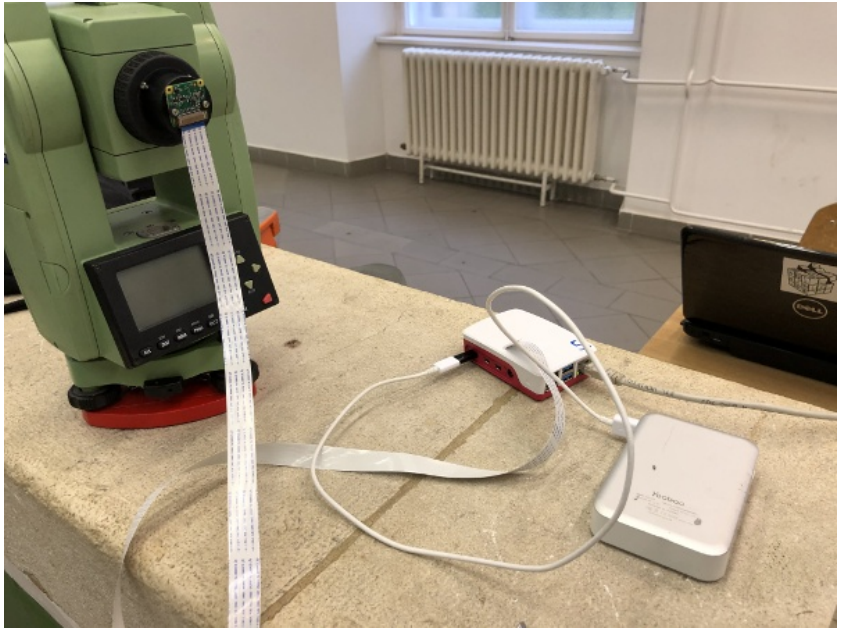
Stable camera records images/videos of moving objects

Resolution can be enhanced by
geodetic telescopes

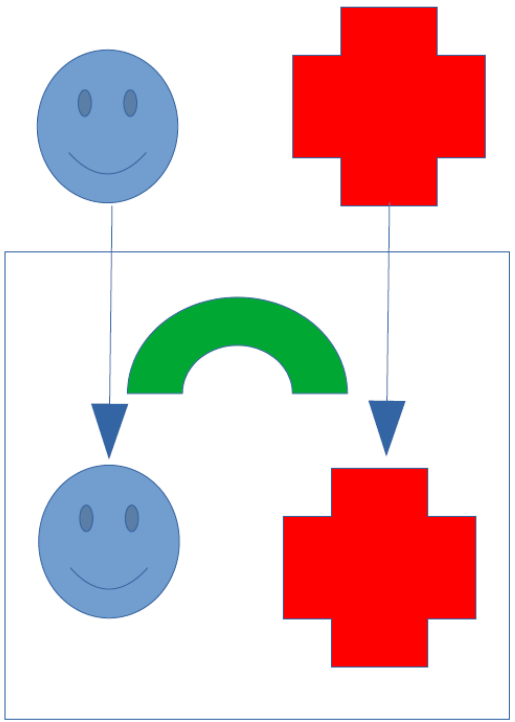
Camera calibrations

Automation

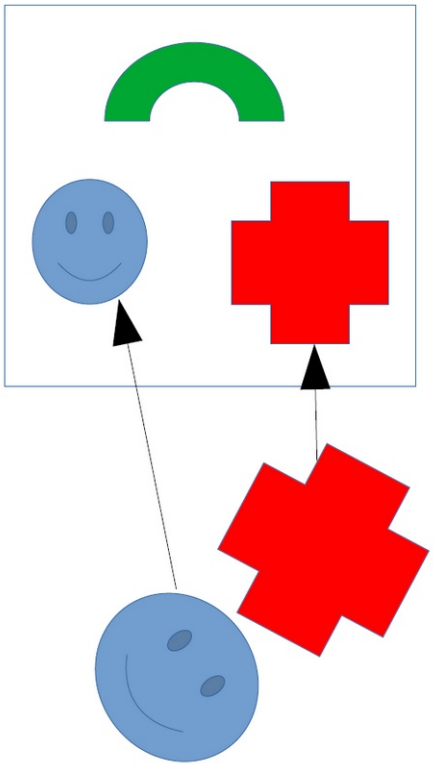
Methods
template matching – Movement in the picture plane
pattern recognition – Movement and rotation



Methods



Template matching



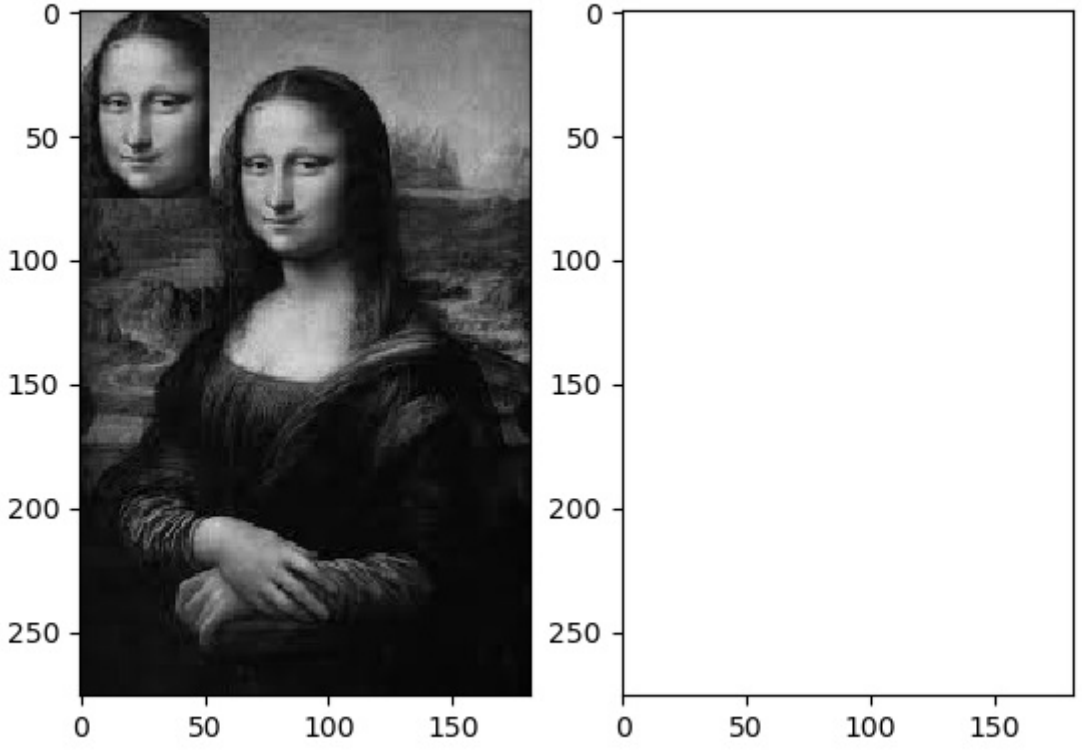
Pattern recognition

Methods

	Template matching	Pattern recognition
Pros	There is always a match	Marker can rotate
	Simple algorithm	Scale factor is allowed
	No special markers needed	Normal of the marker can be estimated
Cons	Chance for false match	Special markers needed
	No rotation	More sensitive for light conditions
	No scale factor	

Template matching

Searching for the most probable position of the template in the source image



Statistics

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2 \quad \text{min}$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \quad \text{min}$$

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y')) \quad \text{max}$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \quad \text{max}$$

Template matching

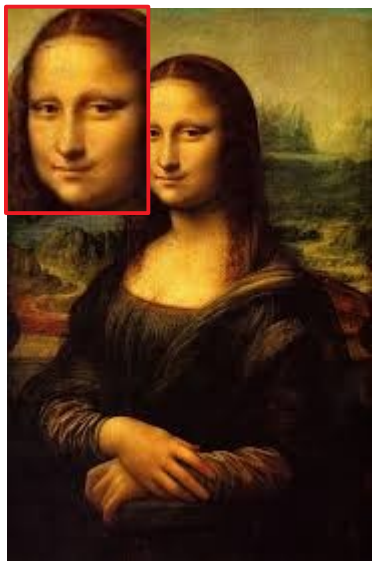
Searching for the most probable position of the template in the source image

source image

template

Template matching

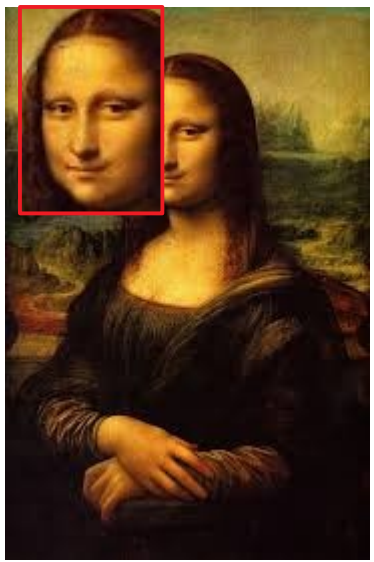
Searching for the most probable position of the template in the source image



Move the template in the source image and
compute statistical values from the colors of the same pixels

Template matching

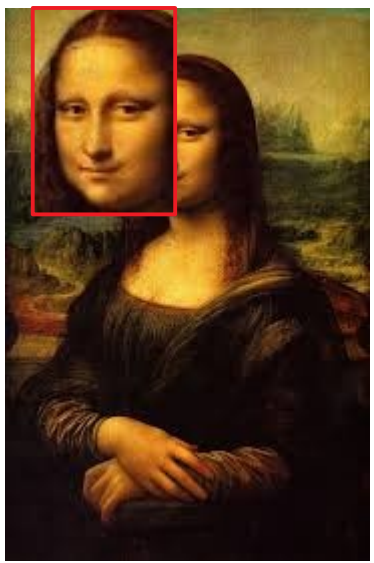
Searching for the most probable position of the template in the source image



Move the template in the source image and
compute statistical values from the colors of the same pixels

Template matching

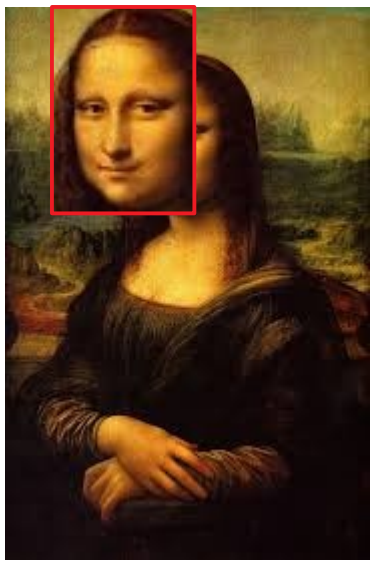
Searching for the most probable position of the template in the source image



Move the template in the source image and
compute statistical values from the colors of the same pixels

Template matching

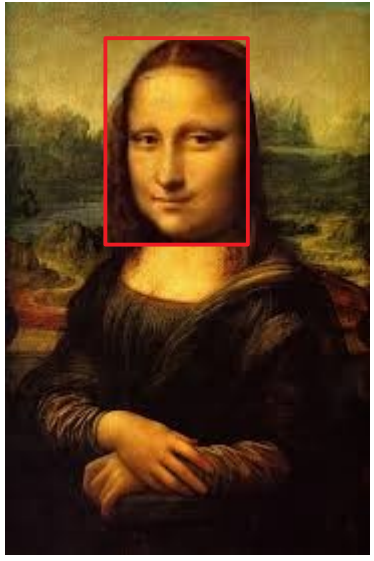
Searching for the most probable position of the template in the source image



Move the template in the source image and
compute statistical values from the colors of the same pixels

Template matching

Searching for the most probable position of template in an image



Move the template in the source image and
compute statistical values from the colors of the same pixels

Satistical values

TM_SQDIFF (0):
Squared difference (min)

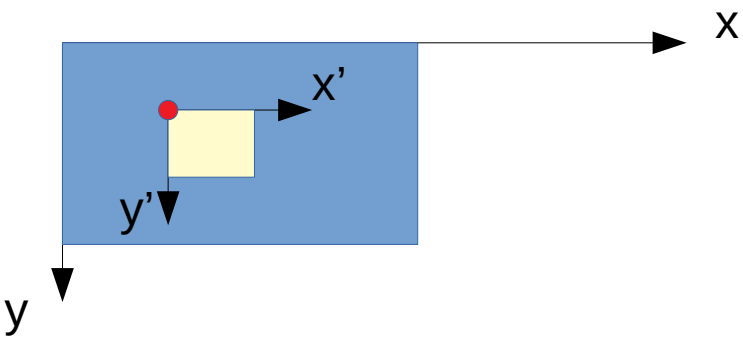
TM_SQDIFF_NORMED (1):
Normalized squared difference (min)

TM_CCORR (2):
Cross correlation (max)

TM_CCORR_NORMED (2):
Normalized cross correlation (max)

TM_CCOEFF (4):
Correlation coefficient (max)

TM_CCOEFF_NORMED (5):
Normalized correlation coefficient (max)

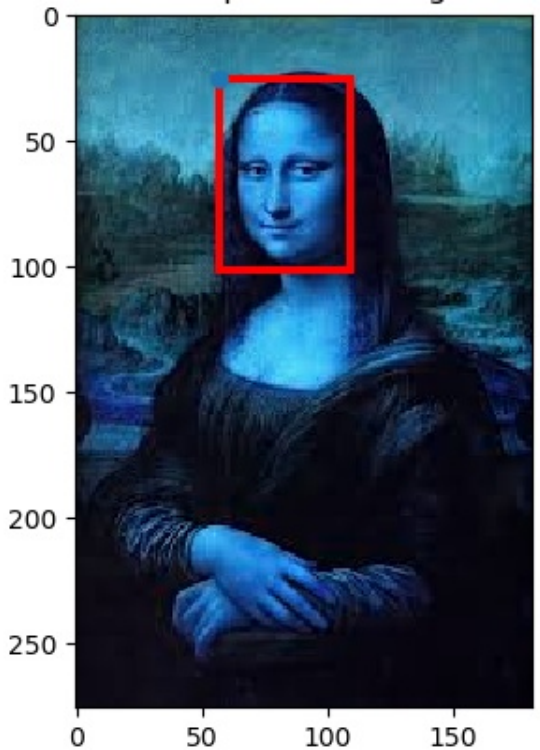


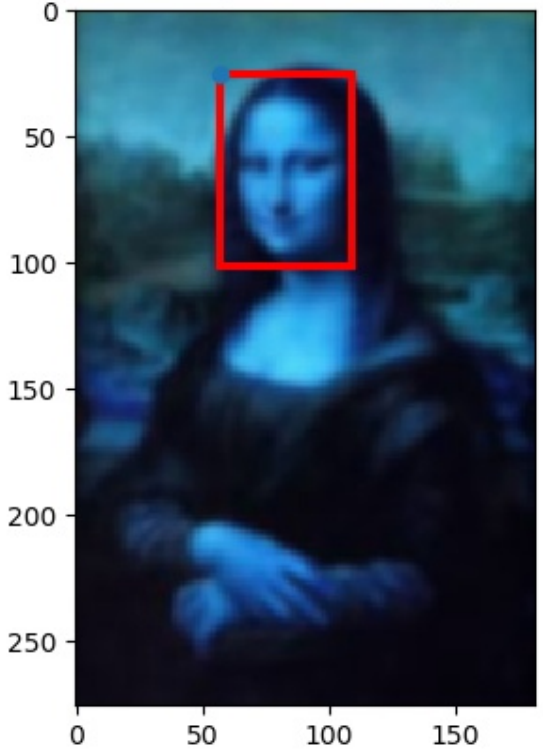
In order to boos efficiency, let's convert images to grayscale ones

https://docs.opencv.org/3.4/de/da9/tutorial_template_matching.html

Use Ulyxes apps

```
python3 img_correlation.py 0 mona_temp4.png monalisa.jpg
25 57
python3 img_correlation.py 0 mona_temp4.png monalisa_blur.jpg
25 57
```

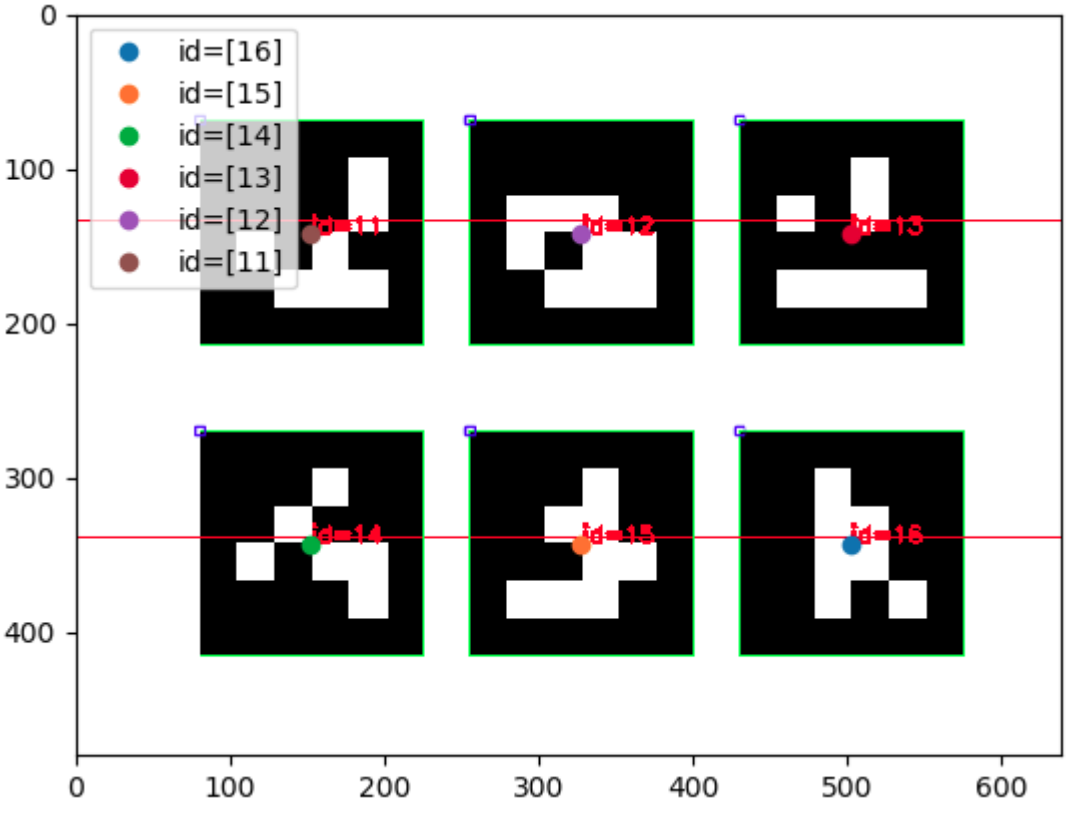




Running time [seconds]	Octave	Python	C++	Python OpenCV	C++ OpenCV
Mona 182 x 276	1.7	0.7	0.2	0.4	0.1
Lab1 ~2k x 3k	466	235	28	1.2	0.8
Lab ~5.5k x 4k	2880	703	109	2.8	1.8

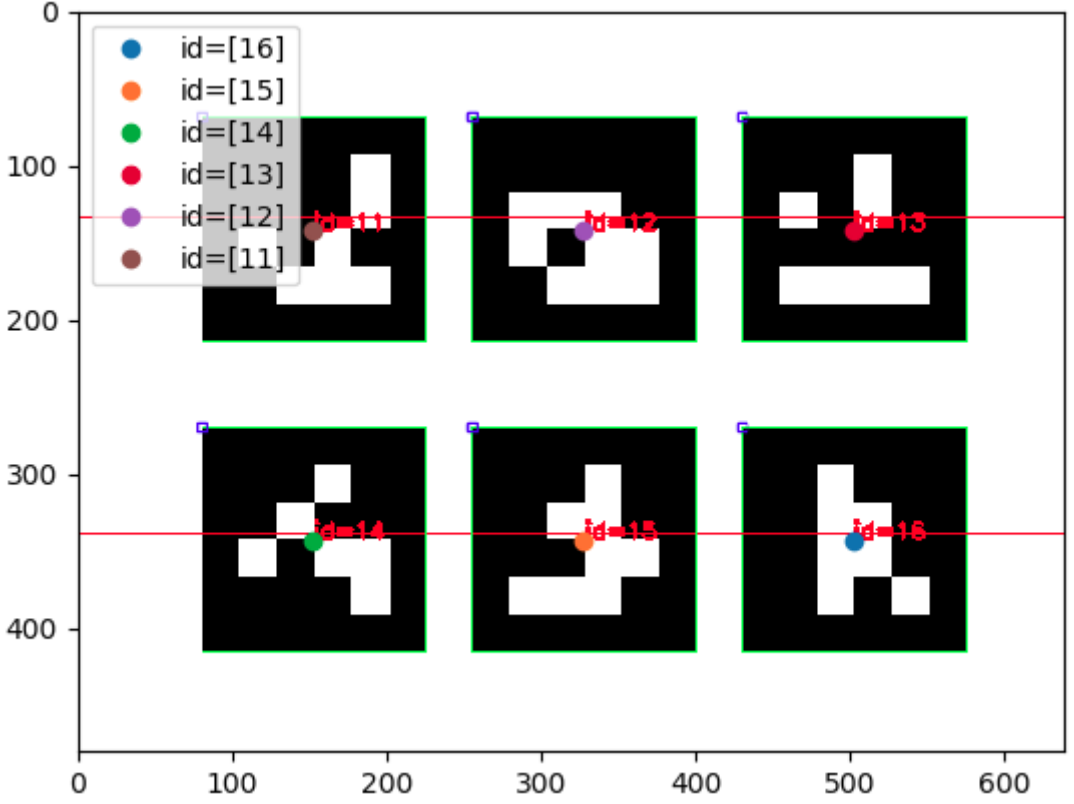
ArUco codes

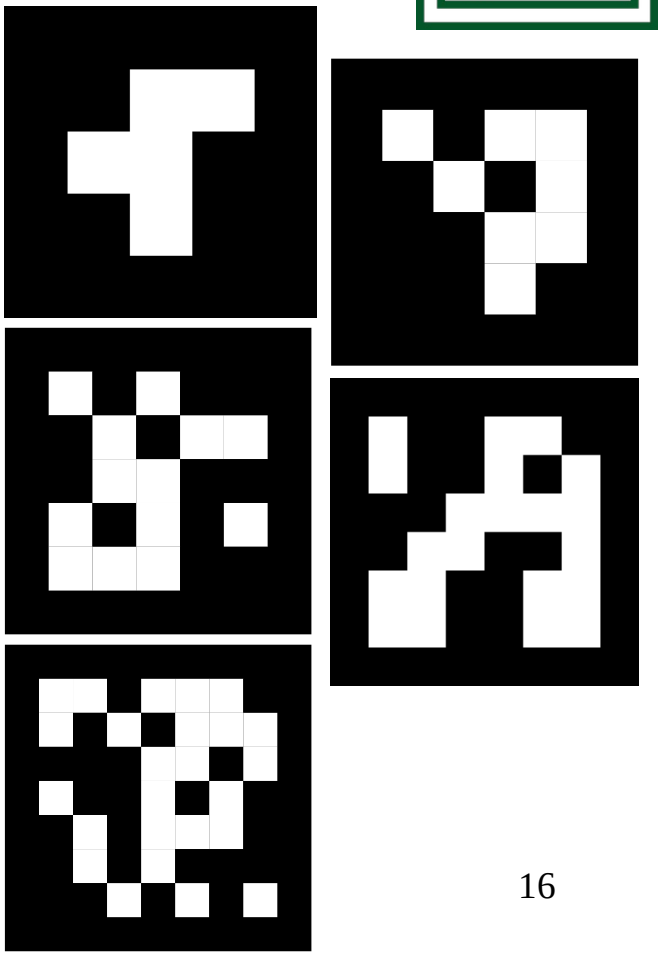
3 x 3, 4 x 4, 5 x 5 , … grid + black frame



ArUco codes

3 x 3, 4 x 4, 5 x 5 , … grid + black frame





ArUco codes

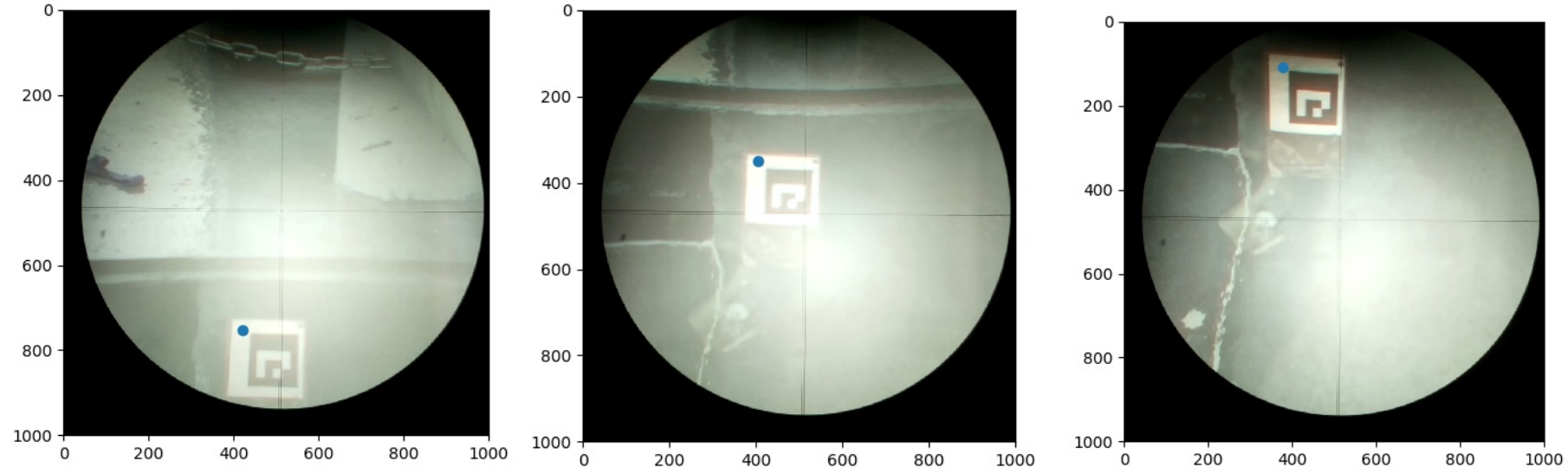
3 x 3, 4 x 4, 5 x 5 , … grid + black frame

As GCP in photogrammetry

GCP = Ground Control Point

Movement detection

- Detecting movements in a plane (no rotation, offset only)
- Processing fast movements from videos
- Improve efficiency – process just a part of the images (minor differences between images)
- https://github.com/zsiki/ulyxes/tree/master/camera
- Metric units, pixels / mm
- http://152.66.5.8/tbence/uszomu/5_20191207_142831_20_v2.mp4



Practical

1) Go through jupyter notebook

   https://github.com/OSGeoLabBp/tutorials/blob/master/english/data_processing/lessons/img_def.ipynb

2) Use your own laptop and laptop camera
   1) Calibrate the camera of your laptop
   2) Take an image with the camera of your laptop of an AruCo code and find it
   3) Take an image with the camera of your laptop of more AruCo codes and find them
   4) Move an AruCo code, take a video using the camera of your laptop and find the position of the AruCo code in real time. Use Ulyxes.

Practical

1) Go through jupyter notebook

   https://github.com/OSGeoLabBp/tutorials/blob/master/english/data_processing/lessons/img_def.ipynb

2) Use your own laptop and laptop camera

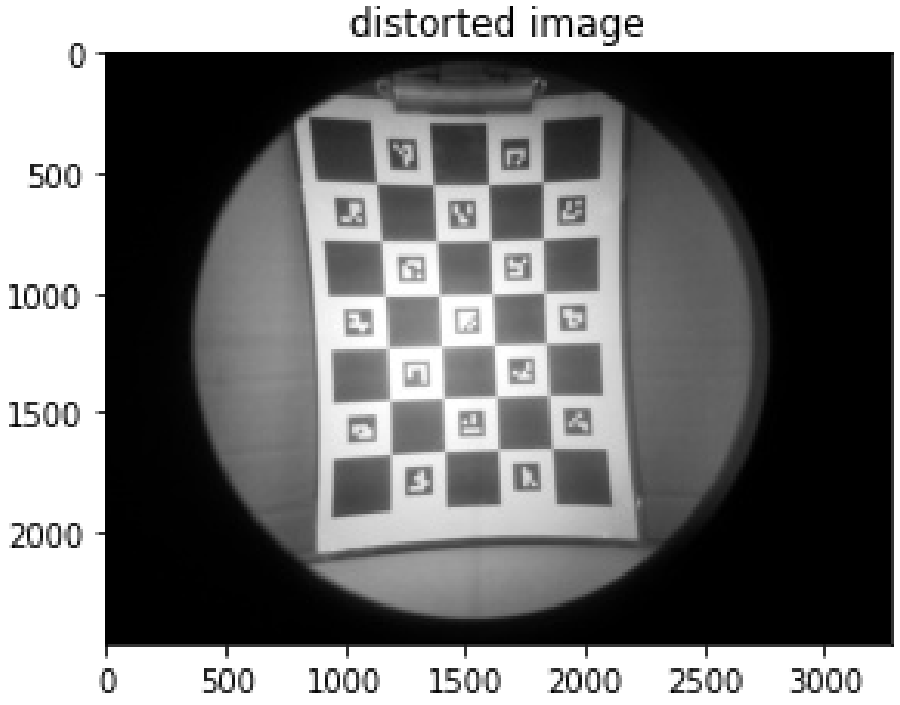
   1) You might have to install python modules: matplotlib, opencv, aruco, yaml

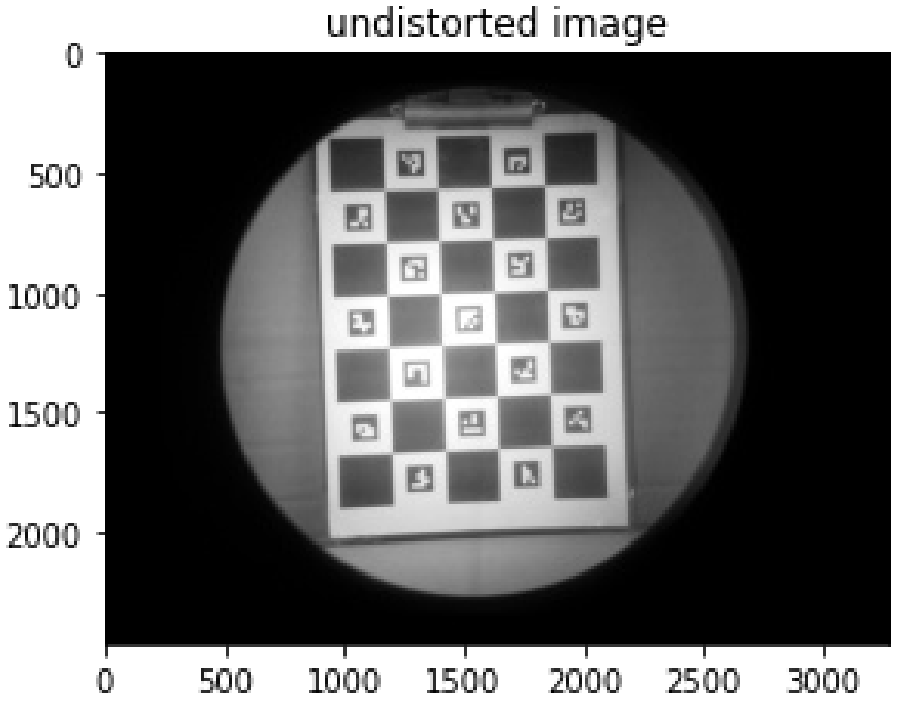
      pip3 install matplotlib
      pip3 install opencv-python
      pip3 install opencv-contrib-python
      pip3 install pyyaml

   2) You might have to install ulyxes

      download ulyxes as zip file (https://github.com/zsiki/ulyxes)
      uncompress into e.g. c:\ulyxes

1. task: camera calibration





Camera calibration

- Radial distortion

$$x' = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y' = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

- Tangetial distortion

$$x' = x + (2p_1 xy + p_2(r^2 + 2x^2))$$

$$y' = y + (p_1(r^2 + 2y^2) + 2p_2 xy)$$

- Camera matrix

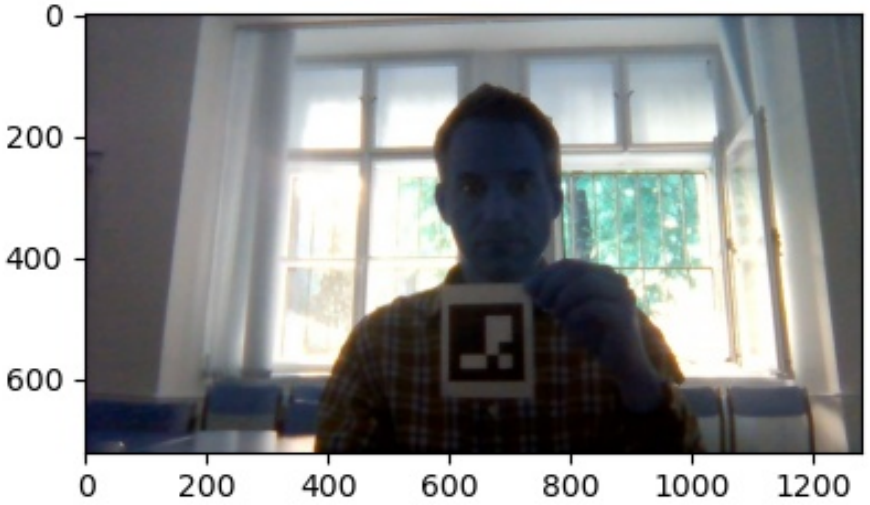
f_x	0	c_x
0	f_y	c_y
0	0	1

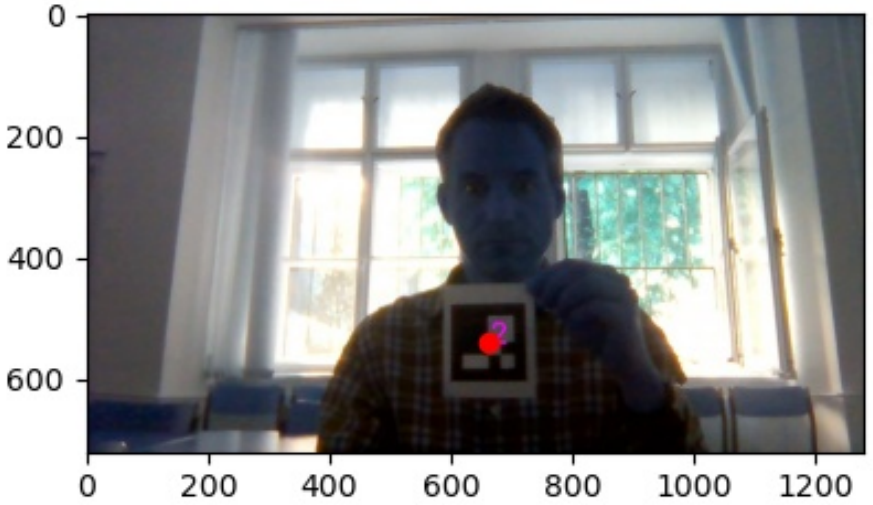
Result in yaml file:

```
camera_matrix:
- - 969.8222790468571
  - 0.0
  - 655.5889488154544
- - 0.0
  - 968.9001026243365
  - 340.2349215331474
- - 0.0
  - 0.0
  - 1.0
dist_coeff:
- - 0.02083211518580173
  - -0.4849224738004900 3
  - -0.011447843509347154
  - 0.003355223089517 9517
  - 1.9246460948 2782
```

2. task: find an AruCo code

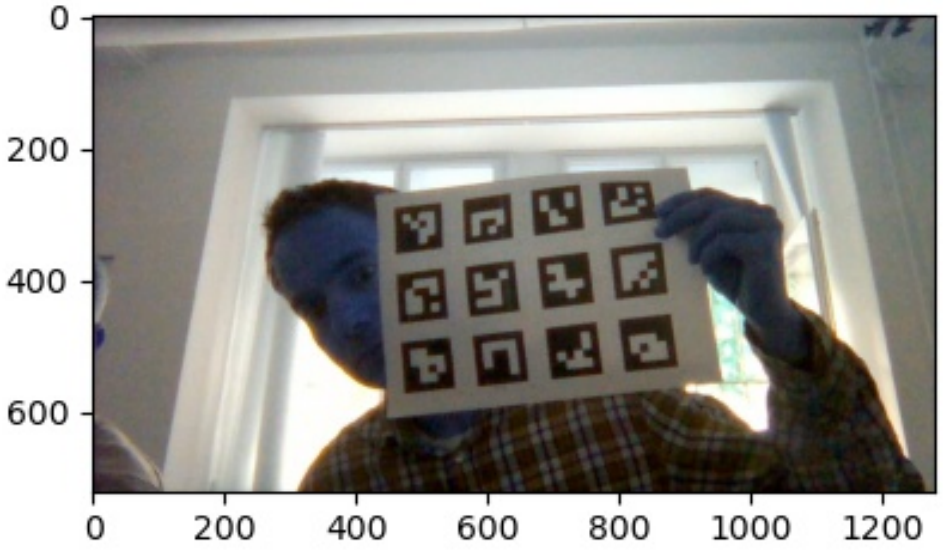
- Take an image with an AruCo code and find it

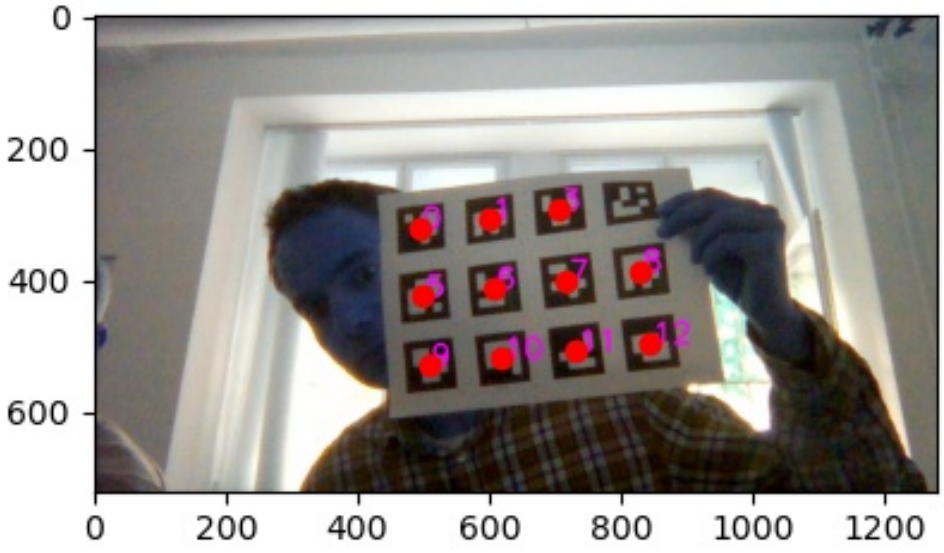




3. task: find AruCo codes

- Take an image with AruCo codes and find them





4. task find moving AruCo code

- Move AruCo code and find its position in a video
- Do it in real-time
- Use video_aruco.py in ulyxes, e.g.

```
python3 c:\ulyxes\camera\video_aruco.py --debug 5 0
```