Geometric and 3D Computer Vision
Assignment 2

Filippo Bergamasco

14/11/2022

Marker detection

In the first assignment you focused on the background/foreground segmentation part of the last year's course final project (see **this page**).

The goal of this second assignment is to detect the circular marker placed on the rotating turntable.

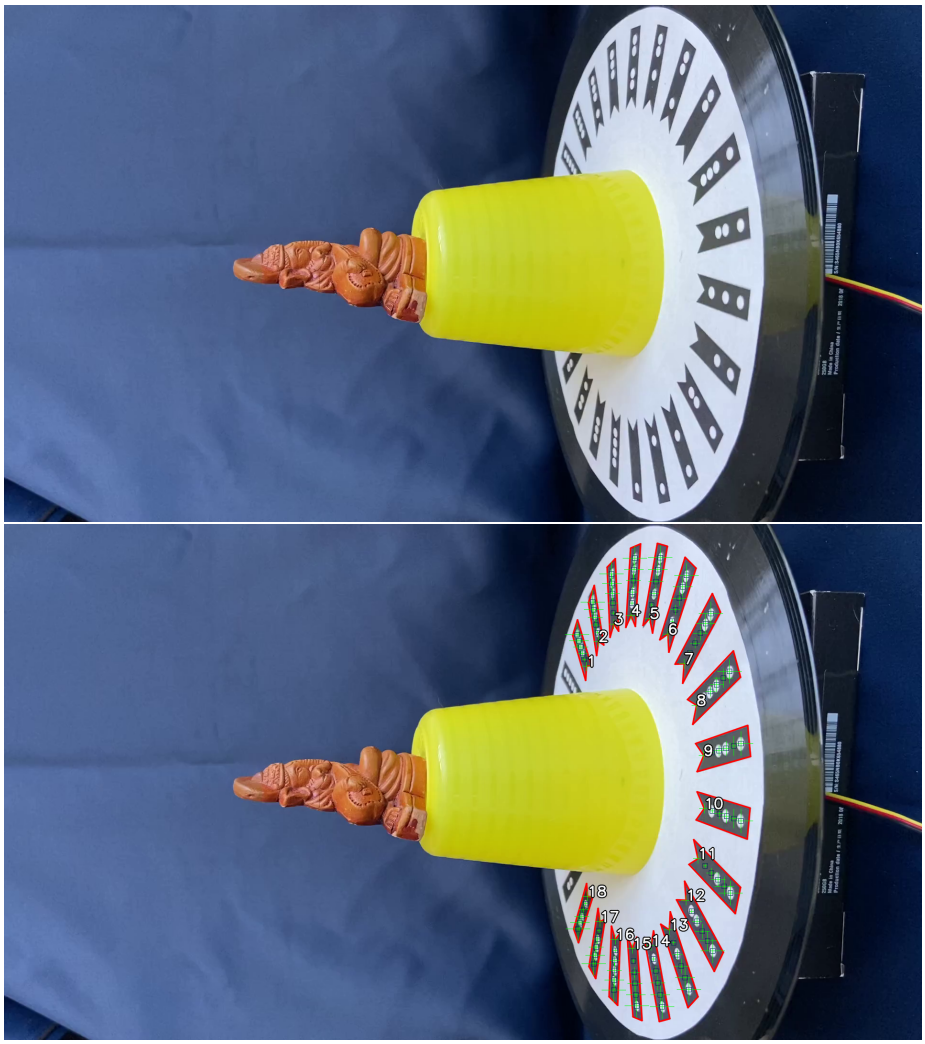
You'll use the same dataset of the first assignment, downloadable at this URL:

https://www.dais.unive.it/~bergamasco/teachingfiles/G3DCV2022/data.7z

Your task is to create a Python program (or notebook) that:

- opens one of the video files cited above
- detects the marker in the turntable, by identifying all the visible *marks* to create a list of coordinates of each point in the image and in the marker reference system (more details follows)
- save all the debug information into new video files named `obj1_marker.mp4` ... `obj4_marker.mp4`
- save point coordinates in CSV files `obj1_marker.csv`, ... ,`obj2_marker.csv`.

See the following Figure for an example of how a debug image might look like. You are not required to copy exactly the same style, it is just a reference for the kind of information to provide:



Marker design

The circular marker is composed by 24 *marks* distributed radially around the center with angle increments of 15°. Each *mark* is a black polygon with 5 vertexes, 1 concave and 4 convex. In the internal area, 5 black and white dots are evenly distributed to binary encode the *mark index* (black dot=1, white dot=0).

Coordinates of each mark vertex in the marker reference frame are as follows:

Point	Coordinates (x,y,z)
O	0, 0, 0

Point	Coordinates (x,y,z)
A	70, 0, 0
B	65, 5, 0
C	98, 5, 0
D	98,-5,0
E	65,-5, 0
D1	75, 0, 0
D2	79.5, 0, 0
D3	84, 0, 0
D4	88.5, 0, 0
D5	93, 0, 0

Expected outputs

For each frame you must detect all the possible (non-occluded) marks, recover the *mark index* from its internal binary encoding and store:

- the pixel coordinate (on the image)
- the 3D coordinate (wrt. the marker reference frame)

of point A (i.e. the concave one) of each mark.

The output CSV file should look as follows:

```
FRAME, MARK_ID,    Px,    Py,    X,     Y, Z
  0,      0,    200.4, 100.5, 70.0,     0, 0
  0,      1,    230.7, 150.1, 67.6, -18.1, 0

...

100,      1,    430.7, 250.1, 67.6, -18.1, 0

...
```

Additionally, you have to produce a video file showing the detection result. I expect to see the points of each mark highlighted together with the *mark index*. All other debug information that might be useful to evaluate your method are welcome.

Approach

You can use any OpenCV function you like to solve the problem. The only requirement is that you understand how each specific algorithm is implemented.

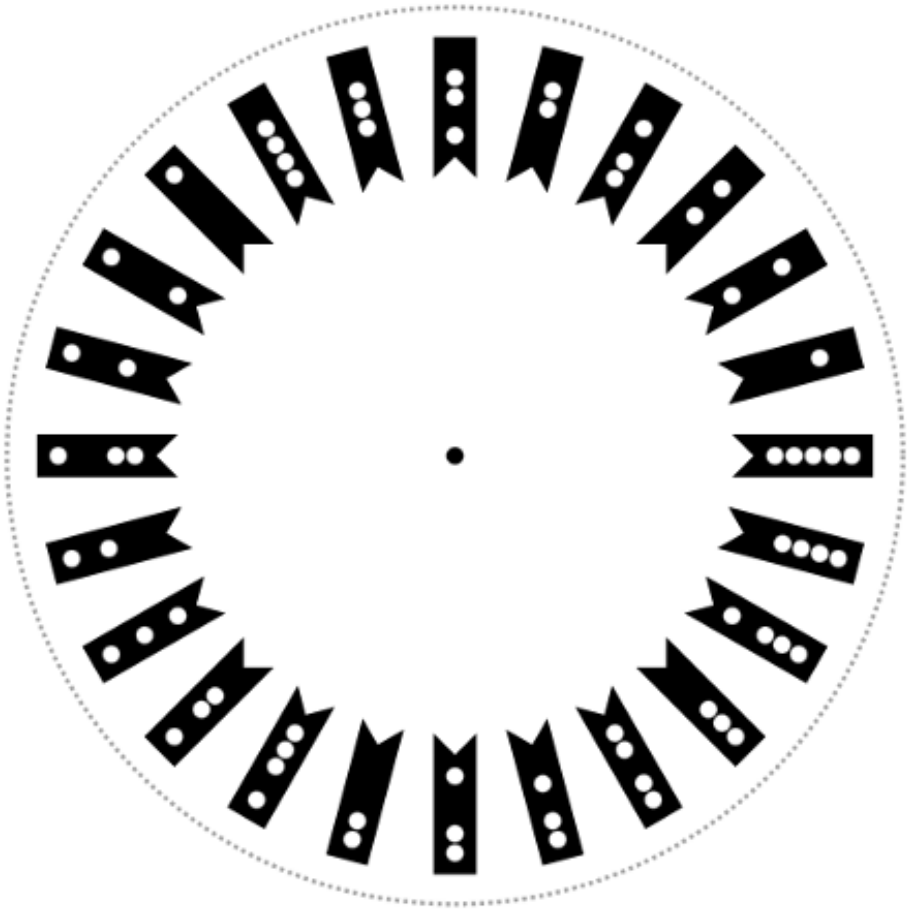


Figure 1: Custom circular marker used in video sequences

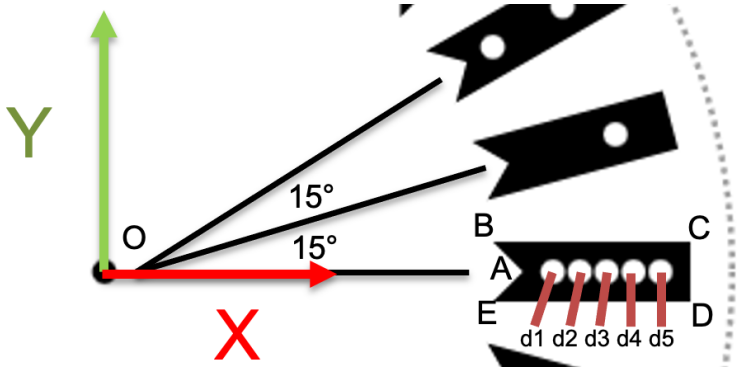


Figure 2: Closeup of a single mark

It is highly recommended that you take advantage of the similarities between subsequent video frames to speedup the process. On average, the whole process should be real-time.

I suggest you to look at these functions:

- findContours: retrieves contours from a binary image
- approxPolyDP: approximates a polygonal curve
- calcOpticalFlowPyrLK: Lukas-Kanade sparse optical flow
- putText: Draws a text string inside an image
- cornerSubPix: Iterative function to find the sub-pixel accurate location of corners or radial saddle points

Submission

Package the produced source code in a zip file named `name_surname_assignment2.zip` and submit it via Moodle at the official course page. Please, **do not submit any data related to the project, like video files, images, etc.**

Report is not required, just submit the source code. Please comment every function so that I can clearly understand what you are doing. **I suggest to add a Readme file if you need to provide additional instructions on how to use/run your programs**.

Notes for the exam

During your final oral exam, you will be asked to run your code and discuss the results. As previously sad, you are free to use any OpenCV function, as long as you fully understand the related algorithms.

For any question, feel free to mail me at filippo.bergamasco@unive.it.

`Last revision: 13/12/2022`