Geometric and 3D Computer Vision
Assignment 3

Filippo Bergamasco

14/12/2022

Augmented reality

The goal of this third assignment is to use the marker's point coordinates detected in the previous assignment to:

1. Compute the marker pose R, T with respect to the camera
2. Use the pose to draw a virtual cube on top of the rotating object like in classical augmented reality applications

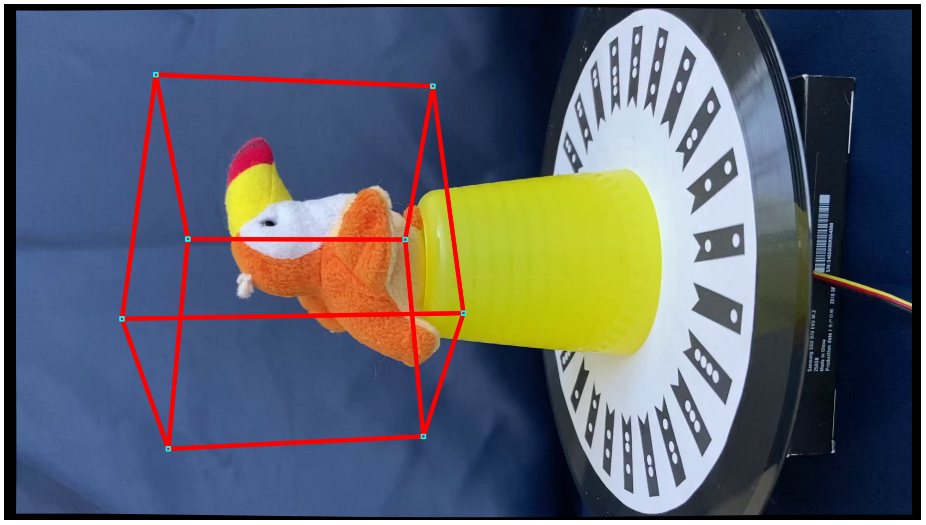

Figure 1: Output frame showing a cube rendered on top of the rotating object.

You should submit a single Python script (or Notebook) that opens one of the video files in the dataset together with the associated CSV file produced in the previous assignment. For each i^{th} frame where more than 6 marks were localized, the pose R_i, T_i should be computed as described in the next section. Then, a cube can be drawn on top the frame and saved in a new video file named `obj01_cube.mp4`, `obj02_cube.mp4`, etc. No other output is required except the video.

Camera pose

The rotation matrix R and translation vector T to transform points from the world (marker) reference frame to the camera reference frame can be computed from the set of 3D-2D point-point correspondences that have been collected in the previous assignment.

Since the marker is planar, you can try two different ways to compute the pose:

1. Use the OpenCV function solvePnP providing in input the object points (ie. the 3D coordinates of marker points in marker reference frame), the image points (ie. the location of marker points in pixel), and the camera intrinsic parameters.
2. Use the OpenCV function findHomography to compute the homography H mapping points from the marker plane to the image plane. Then factorize the Homography into $H = \alpha K[r_1\ r_2\ T]$ as seen in class.

Notes

- I suggest to use the "Infinitesimal Plane-Based Pose Estimation" method implemented in solvePnP by setting the flag `SOLVEPNP_IPPE`
- solvePnP returns a rotation vector instead of a matrix. You can obtain the rotation matrix from a rotation vector with the Rodrigues function.
- Since you'll use more than 4 point correspondences, there is no unique solution for the Homography H. I suggest to specify `RANSAC` in the find-Homography method parameter

Augmented reality

Once the pose R_i, T_i have been computed for the i^{th} frame, the i^{th} projection matrix is obtained as $P_i = K[R_i\ T_i]$. The projection matrix allow you to project any 3D point (defined in the marker reference frame) onto the image.

To draw the cube, start by defining a set of 8 points $p_1 \dots p_8 \in \mathbb{P}^3$ corresponding to the 8 vertexes of the cube to be placed on top of the object. For doing that, it is fine to just guess their coordinates according to the dimension of the underlying marker (see the previous assignment for details). Then, project each point onto the image using the projection matrix and draw lines to connect the edges. The final aspect (size, color, etc.) of the cube is not important. The goal is to observe if the poses have been accurately estimated.

Notes

- Camera must be undistorted before doing this operation. Use the function undistort providing the input image and the camera intrinsic parameters.

Camera intrinsic parameters

Camera has been already calibrated for you. The resulting matrix of intrinsic parameters K and distortion coefficients are the following:

```
K:
[[1.66750771e+03 0.00000000e+00 9.54599045e+02]
 [0.00000000e+00 1.66972683e+03 5.27926123e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]

distortion:
[ 1.16577217e-01 -9.28944623e-02  7.15149511e-05 -1.80025974e-03
  -1.24761932e-01]
```

Submission

Package the produced source code in a zip file named `name_surname_assignment3.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: 14/12/2022`