Computer Vision 2018/2019

Final Project – “DIY 3D Laser Scanner”

The final project consists in the development of a simple 3D laser scanner using off-the-shelf components, namely:

- A webcam
- A laser line projector
- A few sheets of paper and a printer.

The idea is to print a pair of rectangular shapes using an inkjet or laser printer. The two shapes should be thick enough to allow a reliable recognition under noise and different illumination conditions. The setup is built by taping the rectangular shapes on two (non-parallel) planar surfaces (a good solution can be one horizontal on your desk and the other hanging on the wall close to the desk). The object to reconstruct is placed between the two rectangles and the camera is positioned as close as possible to the object but far enough to observe the whole scene (Fig 1.a).

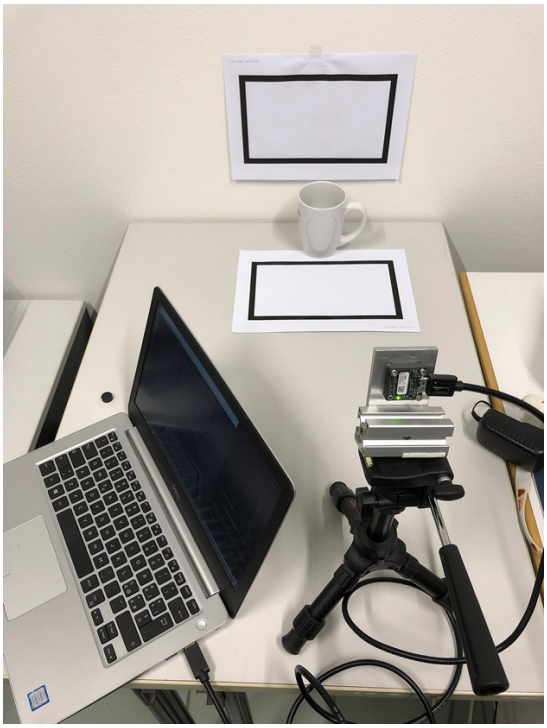

Fig. 1 (a)

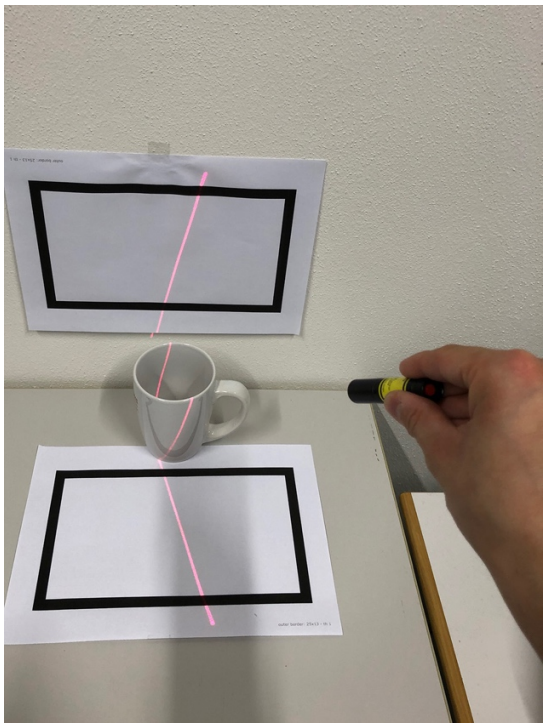

Fig. 1 (b)

The acquisition is performed by swiping the laser line on the object (Fig 1.b), ensuring that the same line is also visible inside the rectangular figures. By analyzing the shape of the line in each frame, the software should output a colored 3D point cloud consistent with the 3D geometry of the scene.

Calibration

The camera must be calibrated before usage to perform an accurate 3D reconstruction. Together with the 3D scanning software, you are also asked to create a program to calibrate a camera using multiple exposures of the given target shown in Fig. 2a.

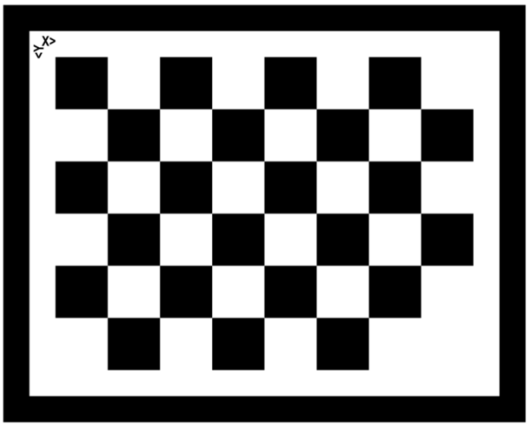

Fig. 2 (a)

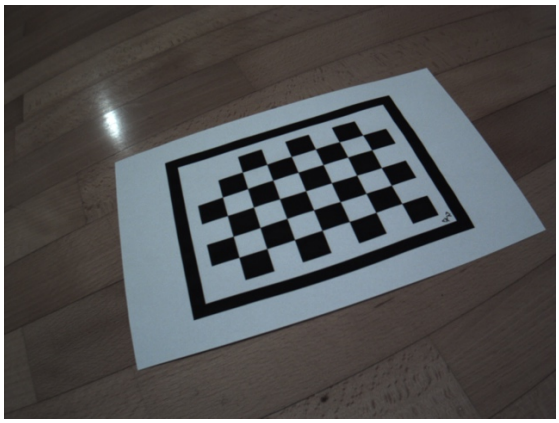

Fig. 2 (b)

The calibration software should work without any user intervention. In other words, it should load all the images found in a certain directory, detect all the corners of the target (if possible) and proceed with the calibration using the OpenCV function `calibrateCamera`. After the calibration, the camera parameters should be written to a file ad used for the subsequent scanner operations. I suggest using a 5-parameters polynomial distortion model.

Image data

Building the scanner hardware is not required for the project. Pre-acquired videos of 4 sample scenes can be downloaded from:

http://www.dais.unive.it/~bergamasco/teachingfiles/cvstuff2019/cv2019_data.7z

The package contains the following files:

- 50 images of the custom calibration target.
- 4 videos of different objects swiped by the laser line: `cup1.mp4`, `cup2.mp4`, `puppet.mp4`, `soap.mp4`.
- The rectangular shape used during the acquisition (`plane.pdf` and `plane.svg`).
- The calibration target used during the calibration (`chessboard.pdf`. and `chessboard.svg`).

Your project should fulfill all the requirements described below considering at least the data provided in the package. In other words, parameters and algorithms can be tuned to “work well” with the given samples.

Requirements

The project must contain two different programs:

1. A *"camera calibrator"* that loads all the 50 calibration images provided and computes the intrinsic parameters of the camera (the intrinsic matrix K and the lens distortion vector assuming a 5-parameters model) **without any user intervention**.
2. A *"3D scanner"* that reads one of the 4 videos provided and process it on a frame-by-frame basis. After processing all the frames (or during the processing), it computes a 3D colored point cloud with all the reconstructed 3D points of the scene that have been illuminated by the laser line. This program can optionally present an interactive interface to the user with the only requirement that the user cannot "help" the scanner to identify the laser line nor the planar rectangles.

The two programs **must be written in Python3** and should use NumPy, OpenCV and all the additional libraries you consider useful for the project.

Point clouds **must be saved in PLY file format** (see http://paulbourke.net/dataformats/ply/) so that they can be visualized with MeshLab http://www.meshlab.net/. Please check that the produced files are valid by verifying if they can be opened with MeshLab.

Optionally, the Open3D library (http://www.open3d.org/) can be used to visualize the reconstructed data in real-time during the processing.

Additional notes:

There are no particular constraints on the number of function/classes/py files produced. You are highly encouraged to give any visual feedback to better understand each algorithm involved. For example, the program could show the boundary of each planar rectangle on the scene, the detected line points, the reconstructed 3D point cloud etc. Any debug information useful to explain the operations involved are welcome and will contribute to the final evaluation of the project during the oral exam.

A reasonable real-time processing speed will be evaluated as a positive feature of your work (although not mandatory to pass the exam). You are invited to report the processing time of each frame and, if appropriate, of each different processing step.

Comment your code whenever possible. Since no additional report is required, comments are a good way to clarify what your code is supposed to do.

Exam Information

Final course exam consists in an **oral discussion about the project.**

When ready to take your final exam, package the source code in a zip file named <name>_<surname>_exam.zip and submit it via Moodle. Then, notify me via mail (filippo.bergamasco@unive.it) so that we can arrange a date (usually within a couple of weeks from the submission). Please, **do not submit any data related to the project (video files, images, calibration data, etc.)**.

During the exam, I'll ask you to explain all the interesting parts of the source code, focusing on the implementation details and the algorithms involved. You are supposed to discuss strengths and limitations of any choice you made during the development. During the discussion, you'll also have to show a demo of your project either on my laptop or yours, based on your preference.

I'll consider the exam passed with full marks if you demonstrate proficiency on the computer vision techniques we have seen during the course in relation to the project you have developed. This means that the actual final performance of the produced code is not critical if the behavior is properly justified and discussed.

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