Computer vision course

Proff. Stefano Ghidoni | Matteo Terreran

Lab 8 - Camera calibration

Task 1 | Acquire a calibration dataset

You need to acquire calibration images using a camera (e.g., your smartphone). Calibration patterns are provided during the lab, but if you work on your own you can get it from https://calib.io/pages/camera-calibration-pattern-generator and print it. Once you selected your pattern, you have to count the number of square intersections per row and column and measure the size of the square side.
You should place the calibration pattern on a flat surface (e.g., your desk) and take several pictures with different orientations and distances. The pattern should appear in many different positions in the images.
Once you have acquired the pictures, you should move them to the computer where you are going to develop your code.
Please note: if you place the pattern at different distances from the camera, the autofocus will automatically adjust the focus distance, but this will slightly change the intrinsics of your camera. For a perfect result, you should go to advanced settings (if available) and freeze the focus. However, by doing so, images will get out of focus if you change the camera-pattern distance too much. Given all these requirements, you shall find the best trade-off.

If you do not have a camera, you can download some pictures here:
https://drive.google.com/file/d/1TFOZcr8P6g6vkRtrkwk5qcqoI2FigSyy/view?usp=sharing
(the squares in the checkerboard have 11cm size).

Task 2 | Calibrate the camera

Create a program that:

- loads the images of the dataset;
- detects the checkerboard intersections in each image - use the cv::findChessboardCorners() function. Optionally, you can use the cv::cornerSubPix() function to refine the detections (check the documentation);
- calibrates the camera using the points of the pattern - use the cv::calibrateCamera() function;
- computes the mean reprojection error - check how it is calculated and describe this in the report;
- undistorts and rectifies a new image acquired with the same camera - you can do that using the maps created by the cv::initUndistortRectifyMap() - check the documentation to understand how this map can be applied to the image;
- shows the distorted and undistorted images in the same window.