HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Camera Calibration

Leopold Gaube - 34948, IN

June 10, 2022

Contents

1 Introduction

When taking images or video with a camera, a 3D scene gets projected onto a 2D image. A simple pinhole-camera model is still often used explain the concept of this projection, but nowadays modern cameras use lenses to capture more light. This gives rise to a problem where some supposed straight lines, e.g. a door, do not appear straight in the resulting image. This unwanted effect is called lens distortion and is usually most pronounced with ultrawide lenses. (The provided video footage has been captured with a ultrawide lens, thus it suffers from Barrel Distortion.) (radial and tangential distortion)

Problem: We usually don't manufacture the sensor ourself How to calculate K and (R, t)?

[Naturally, this is accompanied with a loss in information as the distance of an object to the camera is lost. Therefore, a perfect reconstruction of the original 3D scene from a 2D image is impossible.]

The code and all frames used for this camera calibration task are available in this GitLab repository [Gau22].

2 Task Overview

The following report presents an algorithm to calibrate a camera and undistort images based on this calibration. A calibration video depicting a checkerboard pattern in various positions has been provided. The algorithm first extracts appropriate frames from this video. Then the checkerboard is localized in all calibration frames and its corresponding positions are used to retrieve the intrinsic matrix of the camera. The intrinsic matrix allows to undistort any image taken with this camera, which is demonstrated on the calibration frames.

- easily recognisable patterns, OpenCV uses a simple checkerboard pattern. (printed, flat)

- multiple images (minimum of 3?) of a known object - best results if taken from various positions - compute camera matrix, distorition parameters?, rotation and translational vectors (extrinsics?) - finally, remove distortion

3 Frame Selection

In theory, we only need three frames to retrieve the intrinsic camera matrix. (why?) In practise, more frames give better results and it is also advisable to use frames with the checkerboard present in different parts of the image. It is possible to manually extract appropriate frames from the calibration video, but we chose to automate this process in order to minimize work for the user for future calibrations.

First we have to specify how many calibration frames should be used. For our algorithm we have settled on 15, as this creates good results in a reasonable execution time. Naturally, we do not want to use consecutive frames, but frames that span the entire video. This way, we get varying checkerboard positions (assuming the checkerboard is being moved around in the calibration video).

Some of these frames may have a lot of motion blur which can be problematic for accurately localizing the checkerboard corners. So instead, for each evenly spaced time step of the video, we consider 50 consecutive candidate frames (corresponding to two seconds of video) from which we only choose the least blurred one. (ToDo: Grafik?) The extend of blur in an image can be calculated using the Variance of Laplacian. [BRC16] With OpenCV it is simple to obtain a focus measure for any greyscale image:

```
focus_score = cv2.Laplacian(frame, cv2.CV_64F).var()
```

The absolute value is of no interest, but the highest value of our 50 candidate frames corresponds to the sharpest one.

4 Intrinsic Camera Matrix

In order to estimate the instrinsic camera matrix, a predefined object, e.g. a checkerboard pattern, has to be detected and localized in the calibration frames. The provided calibration video shows a person moving a flat checkerboard throughout the frame.

With OpenCV we can detect and localize this checkerboard pattern using the cv2.findChessboardCorners() method. In our particular video, OpenCV only recognizes the full 7x7 checkerboard pattern in a handfull of frames, therefore, we chose to detect a 6x6 pattern instead, which works more consistently.

So far, the algorithm only localizes the checkerboard pattern in terms of pixel coordinates, which is inprecise. With the cv2.cornerSubPix() method we can refine these coordinates to sub-pixel precision.

In addition to the 2D image positions of the checkerbord pattern of all calibration frames,

the 3D real-world object positions are also needed. The world-coordinate system can be set to any position, canonically, we choose an outer corner of our checkerboard as the world-coordinate origin. Furthermore, the distance between each checkerboard square is set to be one unit in both x and y direction. The checkerboard is flat, which allows all object points to sit on the z=0 plane. This makes it possible to set identical real-world object positions for all 15 calibration frames.

ToDo: labeled checkerboard

OpenCV provides a functionality to calculate the intrinsic camera matrix and distortion coefficients automatically from the object positions and corresponding image positions.

$$K = \begin{pmatrix} f * k_1 & 0 & s_1 \\ 0 & f * k_2 & s_2 \\ 0 & 0 & 1 \end{pmatrix}$$

K is an upper triangular matrix, therefore, the determinant of K is the product of its trace, which is non-zero because the focal point f, as well as k_1 and k_2 (and theta?), are always greater than 0!!

5 Undistortion

Our Algorithm retrieve the following intrinsic matrix

$$K = \begin{pmatrix} 2157.65015388 & 0. & 993.94771538 \\ 0. & 2189.00413434 & 511.18730768 \\ 0. & 0. & 1. \end{pmatrix}$$

with distortion coefficients

$f = -0.83698399, k_1 = 0.69455783, k_2 = -0.01109099, s_1 = -0.01136698, s_2 = 0.0078521$

We save the matrix K to a file, as it only has to be calculated once and can be reused in a different scene assuming the same camera is being used.

The error of our calibration is 0.09529720894585678.

List of Acronyms

References

[BRC16] Raghav Bansal, Gaurav Raj, and Tanupriya Choudhury. Blur image detection using laplacian operator and open-cv. In *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*, pages 63–67, 2016.

[Gau22] Gaube. Camera calibration. `https://gitlab.com/gaubeleo/camera-calibration`, 07.06.2022.