

Computer Vision SS22 Assignment 2: Camera Calibration

Felix Hamburger

Student ID: 35925

Computer Vision SS22

Computer Science Master

Ravensburg Weingarten University

Email: felix.hamburger@rwu.de

Mario Amann

Student ID: 35926

Computer Vision SS22

Computer Science Master

Ravensburg Weingarten University

Email: mario.amann@rwu.de

This report presents a usecase for a camera calibration. The explanations contained in this report will be divided into @@ steps. First a preprocessing step creates images from a video, then reads the created images and performs basic operations on the image. Then corner points of the document will be detected.

By performing a perspective transformation, the document will then be provided in a top-view. With the help of a filter the document will then be changed into a binary format.

1 INTRODUCTION

In everyday life, there are often situations in which it is advantageous to scan a document that has been received and keep it as a copy. Unfortunately not everyone has a ready-to-use scanner with them to scan the document. Though most of the people have a device capable of doing just this - a cell phone with a camera. Using various methods, it is possible to scan documents with the help of the cell phone camera, similar to a scanner, and save them in a suitable format. This report will give a step by step guide on how it is possible to implement a document scanner with Python and OpenCV[[Ope14](#)].

2 PREPROCESSING

Before any preprocessing can be applied, the images have to be created from the given video. Therefore each frame of the video is saved separately. This is done with the imread function[[Docj](#)] of OpenCV which outputs the image as numpy array with the shape (Height, Width, Channel). By default the decoded image is stored in the BGR-format. Which means, the first channel contains the blue

color channel, the second one contains the green color channel and the last the red one. In preparation for the detection of the checkerboard pattern, the images are converted to grayscale format. This conversion is made with the cvtColor function[[Docf](#)] of OpenCV. The conversion of colorspace is made with the following formula[[Doce](#)]:

$$Y = B * 0.114 + G * 0.587 + R * 0.299$$

3 CORNER DETECTION

After the preprocessing steps, the corners of the chessboard can be detected on the images. For this the OpenCV function findChessboardCorners[[Docg](#)] is used. As input parameters the function gets a grayscale image, the size of the pattern and already recognized corners. As size of the pattern the size 6 times 6 was chosen, because in about 500 pictures from 2000 the chessboard is recognized and the chessboard is recognized at completely different places. Since no corners are recognized before, nothing is passed at the already recognized corners. Furthermore, no flag is used and needed because the images were already converted to grayscale in preprocessing, incorrectly extracted squares were manually checked and did not occur, and there was no time pressure to speed up the detection of the corners. If corners of a chessboard are found in the image, the corner points are added to a list of corner points from all images.

4 CAMERA CALIBRATION

With the list of corner points the intrinsic camera parameters can be estimated. The OpenCV function calibrateCamera[[Docc](#)] is used, which is based on the algorithm of the paper by Z.Zhang[[Zha00](#)] from 2000. The

first parameter passed is a matrix consisting of vectors of the calibration points in the coordinate space. In this case this matrix is a multi-dimensional meshgrid. Furthermore the list of corner points from the [Corner Detection](#) section and the size of the image are passed to the function. The remaining passing parameters are passed as NoneType and no flags are set. The function returns the root mean squared re-projection error, the intrinsic camera matrix, the distortion coefficients and the rotation and translation vectors. The intrinsic camera matrix is described as follows:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The returned intrinsic camera matrix from the calibrateCamera function has the values:
 @

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The calibrateCamera function returns following distortion coefficients:
 @

$$\begin{aligned} k_1 &= s \\ k_2 &= s \\ p_1 &= s \\ p_2 &= s \\ k_1 &= s \\ k_1 &= s \end{aligned}$$

As next step the optimal camera intrinsic matrix has to be computed. For this purpose the getOptimalNewCameraMatrix function[[Doch](#)] is used. The function computes the optimal camera intrinsic matrix based on free scaling parameter alpha. The camera matrix, the distortion coefficients vector, the original and the new image size and the free scaling parameter alpha are passed as input. No further optional parameter are passed. The camera matrix and the distortion coefficients vector are the previously calculated values. As original and new image size the original size of the image is passed. The free scaling parameter alpha is set to one. Therefore all pixels are kept and no pixels are removed [[Docd](#)]. The optimal intrinsic camera matrix from getOptimalNewCameraMatrix has the following values:
 @



$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

With the optimal intrinsic matrix the next step the undistortion can take place.

5 UNDISTORTION

Before the transformation can be performed the orientation of the document in the image must be found out. Without knowing the content of the document, it is not possible to know the real orientation. Therefore a few assumptions are made:

1. All provided documents are in ISO 216 format
2. All provided documents are portrait format
3. All provided documents are only rotated 90 degrees right or left, otherwise the document is scanned upside down.

These assumptions define the output of the scanner: A document in portrait format. The first task is to find out if the document is landscape or portrait in the image. Four examples of how the document could have been recorded can be seen in Figure ?? . What is known from the list of the given four corners from the previous section is that the first element  is the most upper point in the image. Next, the distance of this point must be calculated with the second and fourth elements  in the list and these two distances compared. For this purpose, the Euclidean distance calculation is used. If the distance to the second element is higher than to the fourth element, the document is recorded portrait. Otherwise it was recorded landscape. Relative to the previous result, the list of found points must be put into this form:

[upper_left, upper_right, lower_right, lower_left]

This is used to calculate the perspective transformation from the given points to the destination points with the getPerspectiveTransform[[Doci](#)] function from OpenCV. The target points are the set points of the document in ISO 216 format.

$$\begin{aligned} &[[0, 0], [\text{width}, 0], [\text{width}, \text{height}], [0, \text{height}]] \\ &\text{height} = 3000 \\ &\text{width} = \frac{\text{height}}{\sqrt{2}} \end{aligned}$$

getPerspectiveTransform(src_pts, dst_pts)

The last step of this section is to apply the perspective transformation to the grayscale image. For this purpose the warpPerspective function[[Dock](#)] from OpenCV is used:

warpPerspective(grayScale_image, M, (width, height))

The function uses the grayscale image, the calculated perspective Transformation(M), and the size of the output image to perform the transformation. The optional parameters are set to default. The output of the transformation can be seen in Figure ??.

6 SUMMARY

In the last step of the process, the transformed image is converted into a binary image. In various tests, it has been found that an adaptive threshold[Doca] is best suited. Both an adaptive gaussian threshold[Docb] and an adaptive mean threshold show great performance, though an adaptive mean threshold tends to generate less noise in the outputs.

```
adaptiveThreshold(transformation, 255,
    ADAPTIVE_THRESH_MEAN_C,
    THRESH_BINARY, 7, 6)
```

The chosen adaptive mean threshold value is a mean of the blocksize $7 * 7$ minus the constant $c = 6$. The image generated after this process can be seen in figure ??

REFERENCES

- [Doca] OpenCV Documentary. adaptivethreshold. https://docs.opencv.org/4.x/d7/d1b/group__imgproc_misc.html#ga72b913f352e4a1b1b397736707afcde3. Accessed: 2022-05-12.
- [Docb] OpenCV Documentary. Adaptivethresholdtypes. https://docs.opencv.org/4.x/d7/d1b/group__imgproc_misc.html#gga42a3e6ef26247da787bf34030ed772cad0c5189c637calibration62f1ee78. Accessed: 2022-05-12.
- [Docc] OpenCV Documentary. calibratecamera. https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d. Accessed: 2022-06-15.
- [Docd] OpenCV Documentary. Camera calibration. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. Accessed: 2022-06-15.
- [Doce] OpenCV Documentary. Color conversion rgb to gray. https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray. Accessed: 2022-05-09.
- [Docf] OpenCV Documentary. cvtcolor. https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab. Accessed: 2022-05-09.
- [Docg] OpenCV Documentary. findchessboardcorners. https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a. Accessed: 2022-06-14.
- [Doch] OpenCV Documentary. getoptimalnewcameramatrix. https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga7a6c4e032c97f03ba747966e6ad862b1. Accessed: 2022-06-15.
- [Doci] OpenCV Documentary. getperspectivetransform. https://docs.opencv.org/3.4/da/d54/group__imgproc_transform.html#ga15302cbff82bdcddb70158a58b73d981. Accessed: 2022-05-12.
- [Docj] OpenCV Documentary. imread. https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fcea07b3bbd3a56. Accessed: 2022-05-09.
- [Dock] OpenCV Documentary. warp_perspective. https://docs.opencv.org/4.x/da/d54/group__imgproc_transform.html#gaf73673a7e8e18ec6963e3774e6a94b87. Accessed: 2022-05-12.
- [Ope14] OpenCV. *The OpenCV Reference Manual*, 2.4.13.7 edition, April 2014.
- [Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000. Accessed: 2022-06-15.