Silesian University of Technology

Department of Computer Graphics, Vision and Digital Systems

Silesian
University
of Technology

Advanced Vision, Motion and Image Analysis

Lab 1

Camera calibration and image rectification

Gliwice 2025

1 Introduction

The camera intrinsic matrix A is composed of the focal lengths f_x and f_y, which are expressed in pixel units, and the principal point (c_x,c_y), that is usually close to the image center [3]:

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

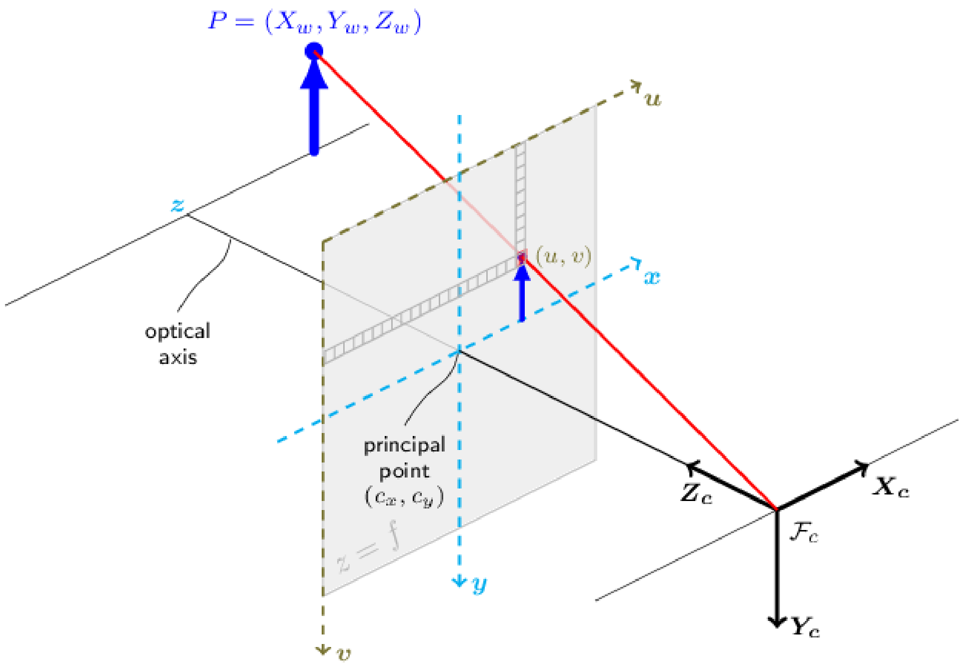

Figure 1: Pinhole camera model [3]

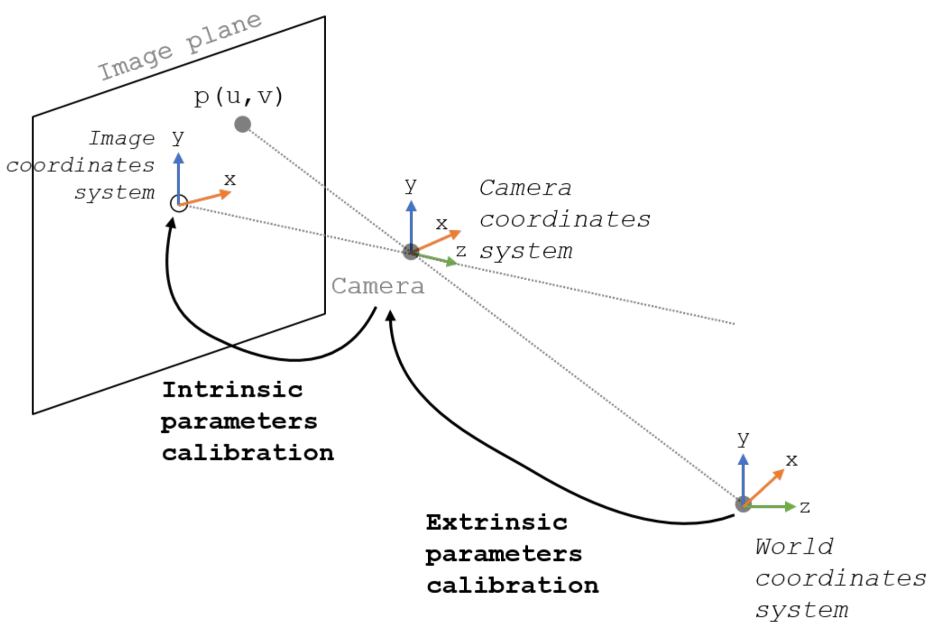

Image plane
p(u,v)
Image coordinates system
y
x
Camera coordinates system
z
Camera
Intrinsic parameters calibration
Extrinsic parameters calibration
World coordinates system

Figure 2: Intrinsic and extrinsic camera parameters [2]

Real lenses usually have some distortion, mostly radial distortion, and slight tangential distortion. So, the above model is extended as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x x'' + c_x \\ f_y y'' + c_y \end{bmatrix} \tag{2}$$

where

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} x' \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_1x'y' + p_2(r^2+2x'^2) + s_1r^2 + s_2r^4 \\ y' \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + p_1(r^2+2y'^2) + 2p_2x'y' + s_3r^2 + s_4r^4 \end{bmatrix} \tag{3}$$

with

$$r^2 = x'^2 + y'^2 \tag{4}$$

and

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix}, \tag{5}$$

if $Z_c \neq 0$. The distortion parameters are the radial coefficients k_1, k_2, k_3, k_4, k_5, and k_6 , p_1 and p_2 are the tangential distortion coefficients, and s_1, s_2, s_3, and s_4, are the thin prism distortion coefficients.

The next figures show two common types of radial distortion: barrel distortion ($1 + k_1r_2 + k_2r_4 + k_3r_6$ monotonically decreasing) and pincushion distortion ($1 + k_1r_2 + k_2r_4 + k_3r_6$ monotonically increasing).

More details you can find in OpenCV documentation - Camera Calibration and 3D Reconstruction [3].

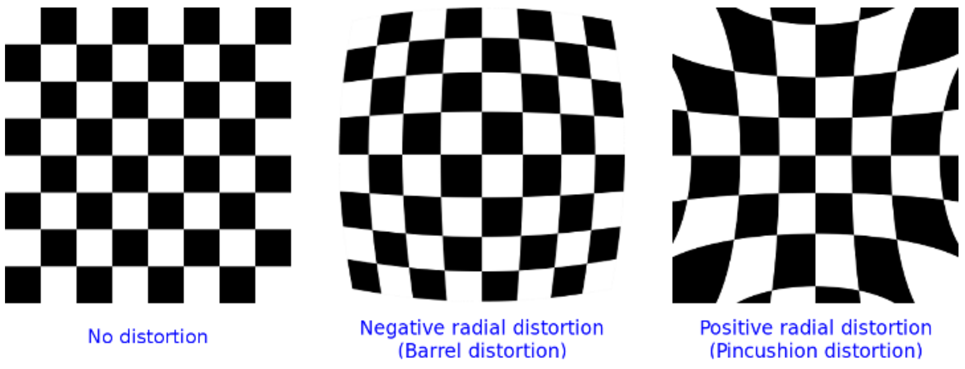

Figure 3: Distortion examples [3]

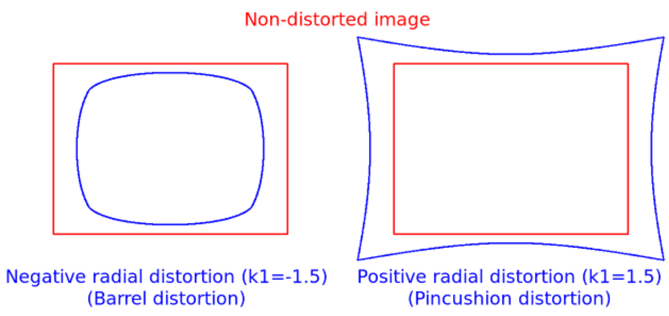
Non-distorted image
Negative radial distortion (k1=-1.5)
(Barrel distortion)
Positive radial distortion (k1=1.5)
(Pincushion distortion)

Figure 4: Distortion examples [3]

2 Tasks

2.1 Task 1

Write a program that allows the initial selection of photos for the purpose of calibration. As a result of the analysis, the identifiers of images that meet the assumed quality criteria should be returned. You need to extract 3 groups of images:

- calibration pattern visible in the left image
- calibration pattern visible in the right image
- calibration pattern visible in both images

2.2 Task 2

On the basis of photos selected during task 1, perform intrinsic camera calibration and designate distortions coefficients, compare the obtained results for the left and right camera. Save the calibration parameter (camera matrix and distortion parameters) to the file, use one of the popular formats - JSON, YAML, INI, or XML.

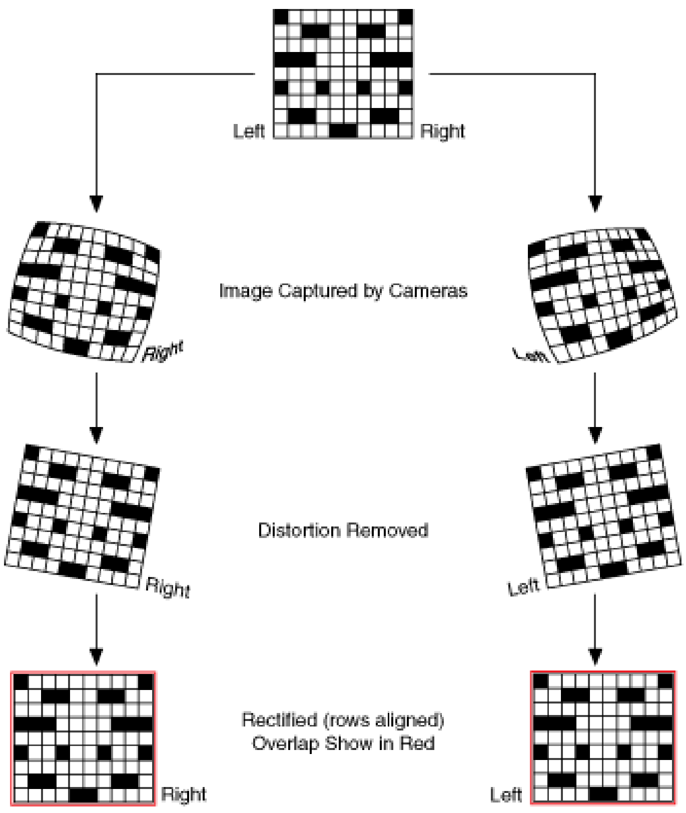

Figure 5: Rectification steps [1]

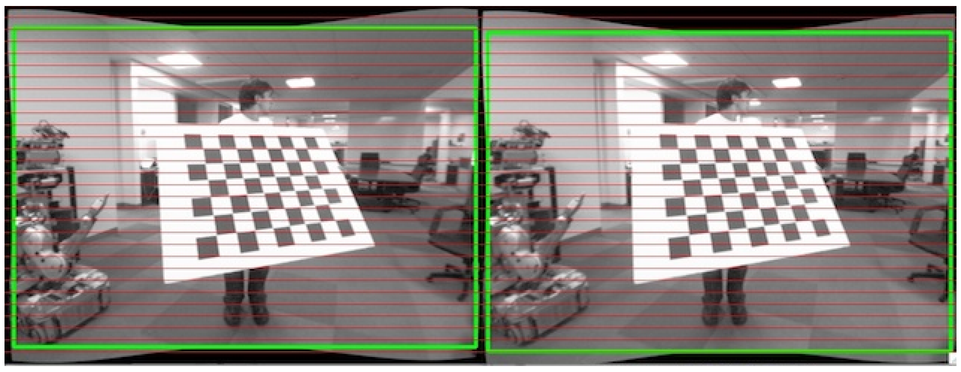

Figure 6: Sample rectification visualization [3]

2.3 Task 3

Using the camera matrix and distortion coefficients from task 2, prepare the application responsible for distortion removal. The input and output images should be displayed side by side. You can use one of the OpenCV methods (`cv::undistort` or `cv::initUndistortRectifyMap`), that transforms an image to compensate for lens distortion.

3 Appendix A

Example code in Python responsible for calibration pattern detection from images using the OpenCV library (`cv::findChessboardCorn` method).

```
import numpy as np
import cv2 as cv
import glob
# termination criteria
criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)
# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((6*7,3), np.float32)
objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)
```

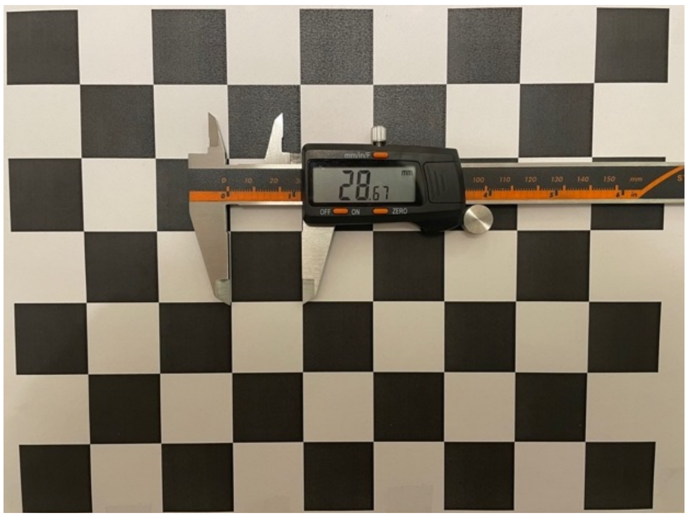

Figure 7: Checkerboard dimensions

```
# Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.
images = glob.glob('*.jpg')
for fname in images:
    img = cv.imread(fname)
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    # Find the chess board corners
    ret, corners = cv.findChessboardCorners(gray, (7,6), None)
    # If found, add object points, image points (after refining them)
    if ret == True:
        objpoints.append(objp)
        corners2 = cv.cornerSubPix(gray,corners, (11,11), (-1,-1), criteria)
        imgpoints.append(corners)
        # Draw and display the corners
        cv.drawChessboardCorners(img, (7,6), corners2, ret)
        cv.imshow('img', img)
        cv.waitKey(500)
cv.destroyAllWindows()
```

3 References

[1] National Instruments. *Stereo Image Rectification*. URL: https://zone.ni.com/reference/en-XX/help/372916P-01/nivisionconceptsdita/guid-c9c3535b-faf7-4ade-9166-513a49d1b90a/. (accessed: 24.05.2021).

[2] University College London. *Intrinsic camera parameters calibration*. URL: https://mphy0026.readthedocs.io/en/latest/calibration/camera_calibration.html. (accessed: 24.05.2021).

[3] OpenCV. *Camera Calibration and 3D Reconstruction*. URL: https://docs.opencv.org/master/d9/d0c/group__calib3d.html. (accessed: 24.05.2021).