UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Corner localization and camera calibration from imaged lattices

Author:
Andrii STADNIK

Supervisor:
Dr. James PRITTS

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2023

Declaration of Authorship

I, Andrii STADNIK, declare that this thesis titled, "Corner localization and camera calibration from imaged lattices" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Corner localization and camera calibration from imaged lattices

by Andrii STADNIK

Abstract

This thesis proposes a model-based approach to improve the detection of calibration board fiducials from calibration imagery taken by wide-angle or fisheye lenses. From a single image, we estimate the camera model and project the calibration board into the image to guide the search for missed detections and reject spurious detections. In addition, we propose a classifier to label ambiguous detections that are geometrically plausible given the estimated camera model and imaged board. The proposed method addresses shortcomings of the state-of-the-art, which struggle to reliably detect board fiducials at the extents of the image, where the lens distortion is most observable. The proposed method recovers additional corners that can be used to place additional constraints on the non-convex camera calibration problem, which improves the likelihood of convergence to a global minimum.

The code for this paper is available on GitHub [1].

[1] https://github.com/anstadnik/camera_calibration

Acknowledgements

I wish to thank the Armed Forces of Ukraine for protecting us. Thanks to those brave people I was able to study and work on this thesis. I want to thank my supervisor, Dr. James Pritts, for his guidance and support throughout the whole process of writing this thesis. I also want to express my gratitude to all of the people who encouraged me to keep going and inspired me to do my best: my groupmates, my friends, and my girlfriend.

Contents

Chapter 1

Introduction and motivation

1.1 Outline of the problem

Better camera calibration improves the performance of various downstream tasks by providing a more accurate mapping between 3D world coordinates and 2D image plane coordinates. This improved mapping enables precise alignment, positioning, and scaling of objects within the scene. By determining the camera's intrinsic and extrinsic parameters, algorithms can correct for lens distortion, estimate depth information, and accurately overlay virtual content. Consequently, tasks such as 3D reconstruction, augmented reality, and object detection can achieve better results in terms of precision, spatial consistency, and overall visual quality.

Although manufacturers can estimate camera calibration parameters a priori, fully automatic calibration is often preferred, especially when camera metadata is unavailable. Currently, wide-angle lenses, particularly in mobile phones and GoPro-type cameras, dominate consumer photography. These cameras pose additional challenges due to their requirement for highly non-linear models with numerous parameters. The high distortion of the image plane also makes finding key points robustly challenging.

Typically, camera calibration is obtained by capturing an image of a known calibration pattern, which is then used to estimate the camera parameters. Alternatively, some methods do not use a calibration pattern but instead, infer geometric constraints directly from the scene. However, this approach is generally less accurate.

As reported by Duisterhof et al. (2022) on Oct. 5, 2022, the current state-of-the-art methods (Olson, 2011; Schöps et al., 2020; Krogius, Haggenmiller, and Olson, 2019) fail on images with high distortion. Duisterhof et al., 2022 suggested an iterative the approach of image undistortion and target reprojection, achieving the superior robustness to the noise than the state-of-the-art methods because the feature detection is performed on the undistorted image.

Instead of searching for the features on the undistorted image from scratch, it is possible to utilize the prior knowledge of the geometry of the calibration board, effectively predicting the possible positions of previously undetected features. It can be done by projecting the board onto the image using the intermediate camera calibration, and then filtering the possible positions in order to eliminate false positives.

These additional points will further constrain the camera calibration, improving the accuracy of the calibration parameters.

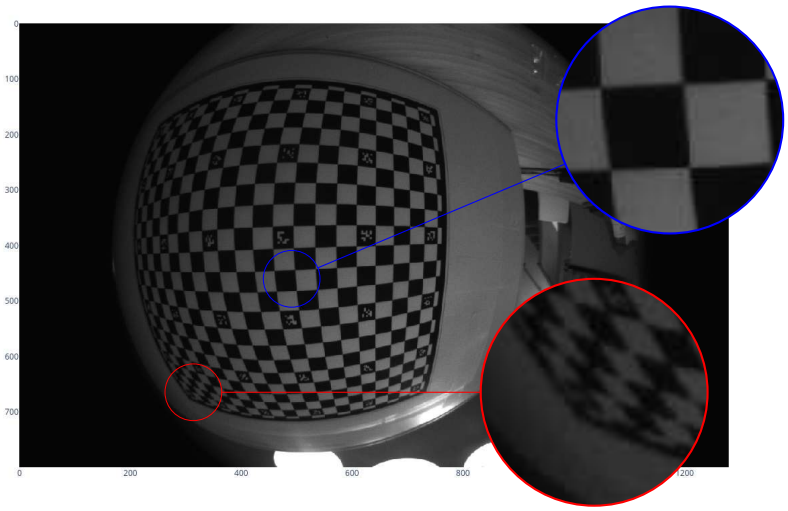


FIGURE 1.1: Example of corners near the center of the image and the edge

1.2 Research objective

The objective of this research is to add additional constrains to the camera calibration by finding previously undetected features on the calibration board. For that, we formulate the set of research questions:

- How to find additional features on the calibration board which were not detected by the feature detector?
- How to filter out falsely detected features?
- Is there a need for finding additional features on the calibration board? Are all of the points detected?

1.3 Thesis structure

This paper has the following structure: in chapter 2, we will describe subtopics of the camera calibration, and outline the current state-of-the-art solutions. In chapter 3, we will describe the key concepts of camera calibration. Also, there we provide additional details on the underlying math, for example, the derivation of the division model inversion. In chapter 4, we will describe the steps of the proposed method in detail, including feature detection, initial camera calibration, camera parameters estimation via minimization of the reprojection error, and the process of searching and filtering the previously undetected features. In chapter 5, we will describe the metrics we used, provide details about the datasets, show the results of each of the algorithm's steps, and evaluate it on multiple metrics. Finally, in chapter 6, we will summarize the results and outline future work.

Chapter 2

Related work

2.1 Camera calibration

Getting the correspondence between the spatial and the image coordinates requires camera calibration. Camera calibration consists of the geometric camera model and the parameters of this model. That information makes it possible to obtain the 2d image coordinates of any point in the 3d space.

Usually, the geometric camera model is obtained from the domain knowledge of the researcher or the camera manufacturer. Often, they choose the simplified model as a trade-off between accuracy and complexity. The model's parameters are usually obtained by solving the constrained optimization problem, given the set of points with known geometry.

2.2 Calibration boards

To achieve a robust calibration, images with repeating patterns are usually used. The camera calibration parameters can be found using prior knowledge of the properties of the pattern, such that the pattern invariants hold on the image. Initially, the chessboard (*OpenCV: Camera Calibration* 2023; V. Douskos, I. Kalisperakis, and G. Karras, 2007) patterns were used (fig. 2.1a).

Later, ArUco (Garrido-Jurado et al., 2014) (fig. 2.1b) and AprilTag (Olson, 2011) (fig. 2.1d) allowed detecting the orientation of the pattern, as well as uniquely identifying each located pattern even under occlusion. Later, based on ArUco pattern, ChArUcO (*OpenCV: Camera Calibration* 2023) (fig. 2.1c) was proposed as more robust.

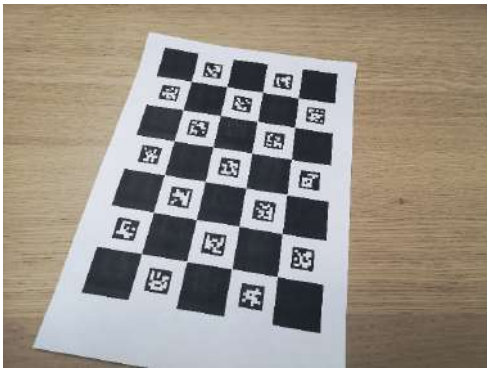
There are also other calibration patterns, some of which are not square (for example, Delitlle (Ha et al., 2017)).

2.3 Camera models

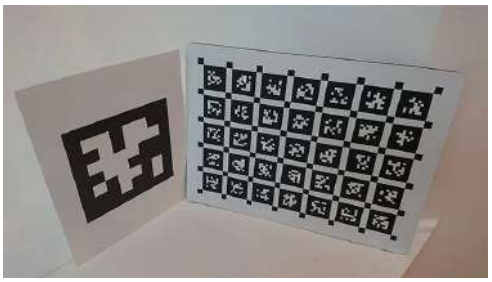
The choice of the camera model depends on the camera's physical properties and the accuracy required. Usually, the parametric models are simpler to use, as they have only a few parameters and deliver good accuracy. The most common are the Double Sphere model (Usenko, Demmel, and Cremers, 2018), the Kannala-Brandt model (Kannala and Brandt, 2006), and the Field-of-View model (Devernay and Faugeras, 2001). In the ill-posed problem of camera calibration, the common choice of the camera model is the division model (Fitzgibbon, 2001). This model is discussed in details in section 3.2.6. However, Schöps et al., 2020 shows that they tend to have significantly higher errors than the non-parametric (general) models. The

(A) Chessboard *OpenCV: Camera Calibration* 2023

(B) ArUco board *OpenCV: Detection of ArUco Markers* 2023



(C) Charuco board *OpenCV: Detection of ChArUco Boards* 2023



(D) AprilTag board Rosebrock, 2020 (right)

FIGURE 2.1: Calibration boards.

Lochman et al., 2021 suggested a framework for converting the parameters of a powerful back-projection Z. Zhang, 2000 model to recover different models' parameters.

2.4 Camera parameters estimation

Camera calibration using repeating patterns was an important subject for a long time, for example, Schaffalitzky and Zisserman, 1998 in 1998 and Z. Zhang, 2000.

Nevertheless, camera calibration is still an open problem; recently, multiple new approaches have arisen. Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart, 2006 proposed an automatic algorithm for camera calibration, which did not use any model of the specific omnidirectional sensor. Is is described in details in section 4.2.2. Lochman et al., 2021 suggest a universal approach to camera calibration, with a separate step of converting between camera models. Hu et al., 2019 used deep learning to detect ChArUcO boards. Recently, on Oct. 5, 2022, Duisterhof et al., 2022 introduced the iterative approach to camera calibration, which outperforms the previous state-of-the-art approaches for wide-angle cameras.

2.5 Boards' features detection

The detection of the calibration board's features is a crucial step in the camera calibration pipeline. The detection accuracy directly affects the calibration accuracy.

Besides feature detection, it's also important to associate the detected features with the board's corners.

There are various feature detection methods. Many of them require prior knowledge of the board's geometry, such as the number of rows and columns, and require the full visibility of the board. For example, the default OpenCV implementation of the chessboard detection (*OpenCV: Camera Calibration* 2023).

Because of that, other methods were proposed, such as Fuersattel et al., 2016 and Geiger et al., 2012 (section 4.4.1). Those methods can detect the board and associate the corners with the borders' corners under the partial occlusion and do not require prior knowledge of the board's geometry.

Chen and G. Zhang, 2005 (section 4.4.1) proposed using Hessian to find pixel positions of the corners.

Chapter 3

Background

3.1 Notation

Term	Description
$\mathbf{u} = (u, v, 1)^T$	A point in the board coordinate system
$\mathbf{x} = (x, y, z, 1)^T$	A point in the world coordinate system
R	A 3×3 rotation matrix
$\mathbf{t}$	A 3×1 translation vector
α_x	Scale factor in the x direction (pixels/mm)
α_y	Scale factor in the y direction (pixels/mm)
c_x, c_y	Coordinates of the principal point (image center)
θ	Angle between the x and y pixel axes
f	Distance from the camera center to the image plane (focal length)
f_x, f_y	Effective focal lengths in the x and y directions
K	Intrinsic matrix incorporating the scaling, introduced by the focal length
H	A 3×3 matrix viewing $z = 0$ (see section 3.2.5)
λ_n	Distortion coefficients

TABLE 3.1: Notation

3.2 Camera model

In this paper, scene and image points are represented using homogeneous coordinates. This approach allows to represent many geometric transformations as linear, which simplifies the mathematical representation of the camera model.

3.2.1 Perspective projection

The perspective projection is a mapping from a 3D point $(x, y, z)^T$ in the world coordinate to the 2D coordinate $(u, v)^T$ on the image plane which is distance f from the center of projection. It is given by the perspective projection equation:

$$(u, v)^T = \frac{f}{z} (x, y)^T .$$

This equation can be written using the homogeneous coordinates:

$$\alpha \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \tag{3.1}$$

where $\alpha = 1/z$ is a scale factor.

3.2.2 Scene to camera projection

A 3D scene point $(x, y, z)^T$ can be projected onto the image plane as $R\,(x, y, z)^T + \mathbf{t}$, where R is a 3×3 rotation matrix and $\mathbf{t}$ is a 3×1 translation vector. Using the homogeneous coordinates, this can be written as:

$$\begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{3.2}$$

3.2.3 Camera to image projection

To project a point from the camera coordinate system to the image plane, we need to apply a homography encoding the camera's intrinsic parameters. This is a 3×3 upper-triangular matrix:

$$\begin{bmatrix} \alpha_x & \alpha_x \cot\theta & c_x \\ 0 & \alpha_y \sin\theta & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.3}$$

where:

- α_x and α_y represent the scale factor of the camera in terms of $^{\text{pixels}}/_{\text{mm}}$ in the x and y directions respectively.
- c_x and c_y are the coordinates of the principal point, which is typically the image center.
- $\cot\theta$ and $\sin\theta$ are related to the skew coefficient, which measures the angle between the x and y pixel axes. The variable θ represents this angle.

For a typical camera, $\theta = \pi/2$ and $\alpha_x = \alpha_y$ Hartley and Zisserman, 2004.

Conventionally, the intrinsic matrix incorporates the scaling, introduced by the focal length:

$$K = \begin{bmatrix} \alpha_x & \alpha_x \cot\theta & c_x \\ 0 & \alpha_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_x & k & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.4}$$

By incorporating the assumptions, mentioned previously into the intrinsic matrix, we can simplify it to:

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.5}$$

3.2.4 Camera matrix

The composition of positioning and orienting the camera, projection, and imaging transformation can be represented by a 3×4 camera matrix (D. Scaramuzza, A. Martinelli, and R. Siegwart, 2006). This matrix can be expressed as:

$$K \begin{bmatrix} I_3 | \mathbf{0} \end{bmatrix} \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} = K \begin{bmatrix} R | \mathbf{t} \end{bmatrix}, \tag{3.6}$$

Hence, the transformation of a point in the scene by the camera $\mathrm{P}^{3\times4}$ can be formulated as:

$$\alpha (u, v, 1)^T = K \begin{bmatrix} R | \mathbf{t} \end{bmatrix} (x, y, z, 1)^T, \tag{3.7}$$

with α being $1/z$.

3.2.5 Projection of the points from the scene plane

When working with the coplanar scene points, we can simplify the projection by assuming that the scene plane is located at $z = 0$. In this case, the projection of the point becomes:

$$\alpha \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{r_3} & \mathbf{t} \end{bmatrix} \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix} = K \underbrace{\begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{t} \end{bmatrix}}_{H} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{3.8}$$

3.2.6 Distortion

The distortion of the image is caused by the lens not being perfectly planar. Typically, the small distortions caused by lens misalignment are ignored Hartley and Zisserman, 2004, allowing us to model the distortion as radially symmetric. Then, the function that maps a point $\mathbf{u} = (u, v, 1)^T$ from a retinal plane to the ray direction in the camera coordinate system is given by:

$$g(\mathbf{u}) = (u, v, \psi(r(\mathbf{u})))^T, \tag{3.9}$$

where $r(\mathbf{u}) = \sqrt{u^2 + v^2}$ is the radial distance from the principal point.

Back-projection using the Division Model

The division model has a good ability to model the distortion of the wide-angle lenses (Fitzgibbon, 2001), and is wildly used (Pritts et al., 2021; D. Scaramuzza, A. Martinelli, and R. Siegwart, 2006). The model is defined as:

$$\psi(r) = 1 + \sum_{n=1}^{N} \lambda_n r^{2n}, \tag{3.10}$$

where λ_n are the distortion coefficients.

The function $\psi(r)$ is not invertible in general. Let $\widehat{\mathbf{x}} = (x, y, z)^T = \alpha g(\mathbf{u})$ be a ray in the camera coordinate system.

Then,

$$\frac{\mathbf{x}}{z} = \left(\tfrac{x}{z}, \tfrac{y}{z}, 1\right)^T \tag{3.11}$$

$$= \left(\frac{\alpha u}{\alpha\psi(r(\mathbf{u}))}, \frac{\alpha v}{\alpha\psi(r(\mathbf{u}))}, 1\right)^T \tag{3.12}$$

$$= \left(\frac{u}{\psi(r(\mathbf{u}))}, \frac{v}{\psi(r(\mathbf{u}))}, 1\right)^T. \tag{3.13}$$

From 3.13 we see that

$$\begin{cases} \frac{x}{z} = \frac{u}{\psi(r(\mathbf{u}))} \\ \frac{y}{z} = \frac{v}{\psi(r(\mathbf{u}))} \end{cases} \implies \begin{cases} u = \frac{x\psi(r(\mathbf{u}))}{z} \\ v = \frac{y\psi(r(\mathbf{u}))}{z} \end{cases} . \tag{3.14}$$

Now, let $\widehat{r}$ be a root of $r(\mathbf{u}) = \sqrt{\frac{x*\psi(r)}{z}^2 + \frac{y*\psi(r)}{z}^2} = r$.
Then, $\mathbf{u} = f(\mathbf{x}) = \frac{\widehat{r}}{r(\mathbf{x})}\mathbf{x}$, where $f(\cdot)$ is the inverse of $g(\cdot)$.

3.2.7 Complete projection and backprojection

Now, the complete projection and back-projection can be formulated as follows:

$$\alpha\mathbf{u} = Kf(H\mathbf{x}) \qquad \text{(Projection)}$$

$$\alpha\mathbf{x} = H^{-1}g(K^{-1}\mathbf{u}). \qquad \text{(Back projection)}$$

Chapter 4

Approach

4.1 Feature detection

For feature detection, we used the approach, proposed by Geiger et al., 2012 (see sections 2.5 and 4.4).

4.2 Camera calibration

In this work, we obtained the camera calibration in two steps: first, we used the approach, proposed by Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart, 2006 to obtain the initial approximation of the camera parameters. Then, we used optimization to minimize the reprojection error between the board and the back-projected corners. A division model is very powerful and can express a wide range of distortions (Pritts et al., 2021). In this work, we used a model with two parameters to allow for better flexibility during the optimization.

4.2.1 Reprojection error

The reprojection error is the distance between the reprojected point and the measured one. It is used to evaluate the quality of the camera calibration.

We minimized the reprojection error between the board and back-projected corners, which were initially detected. The projection and back-projection are the inverse of each other, hence minimizing the error between the projection of the board and the detected corners and minimizing the error between the back-projection of the detected corners and the board are equivalent. We minimized the error between the board and the back-projected corners because back-projection does not require the root finding.

Let's define the following variables: $\boldsymbol{\theta} = \left(\theta_x, \theta_y, \theta_z\right)^T$ is the vector of Euler rotation angles, $\mathbf{t} = \left(t_x, t_y, t_z\right)^T$ is the translation vector, $\boldsymbol{\lambda} = \left(\lambda_1, \lambda_2\right)^T$ is the intrinsic parameters vector, is the focal length, and $\mathbf{s} = \left(s_x, s_y\right)^T$ is the sensor size. From the input image, we know the resolution $\mathbf{r} = \left(r_x, r_y\right)^T$. From the rotation vector, we can compute the rotation matrix $\mathbf{R}$ as:

From $\boldsymbol{\theta}$, the rotation matrix R can be calculated as follows:

$$R = R(\theta_x)R(\theta_y)R(\theta_z), \tag{4.1}$$

where

$$R(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \tag{4.2}$$

$$R(\theta_y) = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \tag{4.3}$$

$$R(\theta_z) = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.4}$$

Then, H is given by:

$$H = \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{t} \end{bmatrix}. \tag{4.5}$$

We can compute the intrinsic camera matrices as follows:

$$K = \begin{bmatrix} \frac{fr_x}{s_x} & 0 & \frac{r_x}{2} \\ 0 & \frac{fr_y}{s_y} & \frac{r_y}{2} \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.6}$$

Then, the back-projection of a 2D point $\mathbf{u} = (u, v, 1)$ into a scene point with $z = 0$ $\mathbf{x} = (x, y, 1)$ is given by:

$$\mathbf{x} = H g_{\lambda_1,\lambda_2}(K^{-1}\mathbf{u}). \tag{4.7}$$

4.2.2 Initial approximation

Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart, 2006 proposed an automatic method for camera calibration, which consisted of the following steps:

- Solving for the camera extrinsic parameters
- Solving for the camera intrinsic parameters
- Linear refinement of the intrinsic and extrinsic parameters
- Iterative center detection
- Non-linear refinement

In this work, we focused on a single image, while the original approach relied on multiple images. Therefore, we used only the first two steps of the algorithm.

Solving for the camera extrinsic parameters

To derive the solver for the camera extrinsic parameters, start from eq. (Projection):

$$\alpha \mathbf{u} = K f(H\mathbf{x}) \tag{4.8}$$

$$\alpha K^{-1}\mathbf{u} = f(H\mathbf{x}) \qquad \text{Move } K \text{ to the left side} \tag{4.9}$$

$$\alpha g(K^{-1}\mathbf{u}) = g(f(H\mathbf{x})) \qquad \text{Set } \widehat{\mathbf{u}} = K^{-1}\mathbf{u}\text{; apply } g(\cdot) \text{ to both sides} \tag{4.10}$$

$$\alpha \begin{pmatrix} \widehat{u}_x \\ \widehat{u}_y \\ \psi(r(\widehat{\mathbf{u}})) \end{pmatrix}^T = H\mathbf{x}. \tag{4.11}$$

For K we used the placeholder values. f was set to a constant value for the typical consumer camera, and c_x, c_y were set to the center of the image. The correct values will be computed during the optimization section 4.2.3.

To eliminate the dependency on the scale α, multiply both sides vectorially by $g(\widehat{\mathbf{u}})$:

$$\alpha g(\widehat{\mathbf{u}}) \times g(\widehat{\mathbf{u}}) = g(\widehat{\mathbf{u}}) \times H\mathbf{x} = 0 \implies \left(\widehat{u}, \widehat{v}, \psi(r(\widehat{\mathbf{u}}))\right)^T \times \begin{bmatrix}\mathbf{r_1} & \mathbf{r_2} & \mathbf{t}\end{bmatrix}^T \mathbf{x} = 0. \quad (4.12)$$

From eq. (4.12), we can see that a point contributes to three homogeneous equations:

$$\widehat{v}(r_{31}x + r_{32}y + t_3) - g(r(\widehat{\mathbf{u}}))(r_{21}x + r_{22}y + t_2) = 0 \quad (4.13)$$
$$g(r(\widehat{\mathbf{u}}))(r_{11}x + r_{12}y + t_1) - \widehat{u}(r_{31}x + r_{32}y + t_3) = 0 \quad (4.14)$$
$$\widehat{u}(r_{21}x + r_{22}y + t_2) - \widehat{v}(r_{11}x + r_{12}y + t_1) = 0. \quad (4.15)$$

Only eq. (4.15) is linear in the unknowns. Each point gives a single equation. Now, by rewriting the equation in the matrix form $M \cdot \mathbf{h} = 0$, where

$$\mathbf{h} = \left(r_{11}, r_{12}, r_{21}, r_{22}, t_1, t_2\right)^T,$$

we get:

$$M = \begin{bmatrix} -\widehat{v}_1 x_1 & -\widehat{v}_1 y_1 & -\widehat{u}_1 x_1 & -\widehat{u}_1 y_1 & -\widehat{v}_1 & -\widehat{u}_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -\widehat{v}_N x_N & -\widehat{v}_N y_N & -\widehat{u}_N x_N & -\widehat{u}_N y_N & -\widehat{v}_N & -\widehat{u}_N \end{bmatrix} \quad (4.16)$$

, where N is the number of points.

The linear estimate of $\mathbf{h}$ is found by minimizing $\|M \cdot \mathbf{h}\|^2$ using SVD. The solution is known up to a scale factor.

To find t_1 and t_2, note that $\mathbf{r_1}$ and $\mathbf{r_2}$ are orthonormal:

$$\begin{cases} \lambda^2 r_{11} r_{12} + \lambda^2 r_{21} r_{22} + \lambda^2 r_{31} r_{32} = 0 & (4.17a) \\ \lambda \sqrt{r_{11}^2 + r_{21}^2 + r_{31}^2} = 1 & (4.17b) \\ \lambda \sqrt{r_{12}^2 + r_{22}^2 + r_{32}^2} = 1, & (4.17c) \end{cases}$$

where λ is non-zero multiplier.

Now, to solve for r_{31} and r_{32}, first find possible values for r_{32}^2:

$$-\frac{r_{11}r_{12}+r_{21}r_{22}}{r_{32}} = r_{31} \quad \text{Solve eq. (4.17a) for } r_{31} \tag{4.18}$$

$$\sqrt{\lambda^2\left(\frac{(r_{11}r_{12}+r_{21}r_{22})^2}{r_{32}^2}+r_{11}^2+r_{21}^2\right)} = 1 \quad \text{Substitute into eq. (4.17b)} \tag{4.19}$$

$$\lambda^2\left(\frac{(r_{11}r_{12}+r_{21}r_{22})^2}{r_{32}^2}+r_{11}^2+r_{21}^2\right) = 1 \quad \text{Square both sides} \tag{4.20}$$

$$\lambda^2\left(\frac{(r_{11}r_{12}+r_{21}r_{22})^2}{r_{32}^2}+r_{11}^2+r_{21}^2-r_{12}^2-r_{22}^2-r_{32}^2\right) = 0 \quad \text{Subtract eq. (4.17c)} \tag{4.21}$$

$$\frac{(r_{11}r_{12}+r_{21}r_{22})^2}{r_{32}^2}+r_{11}^2+r_{21}^2-r_{12}^2-r_{22}^2-r_{32}^2 = 0 \quad \text{Divide both by } \lambda^2 \tag{4.22}$$

$$r_{32}^4-(r_{11}^2+r_{21}^2-r_{12}^2-r_{22}^2)r_{32}^2-(r_{11}r_{12}+r_{21}r_{22})^2 = 0 \quad \text{Multiply by } r_{32}^2. \tag{4.23}$$

Now, solve eq. (4.23) for r_{32}^2, and take a root to find possible values for r_{32}.

To find r_{31}, substitute the found values for r_{32} into eq. (4.18) or subtract eq. (4.17a) from eq. (4.17a) and solve for r_{31} depending on the value of r_{32}:

$$\begin{cases} r_{31} = -\dfrac{r_{11}r_{12}+r_{21}r_{22}}{r_{32}} & r_{32} \neq 0 \quad (4.24a) \\ r_{31} = r_{11}^2+r_{21}^2-r_{12}^2-r_{22}^2 & r_{32} = 0. \quad (4.24b) \end{cases}$$

Now, it's possible to find H for each of the pairs of r_{31} and r_{32}.

Lastly, to select the correct H, the author assumes that one of the boards' corners has the coordinates $(0,0)^T$. Then, the rotation wouldn't affect this point, and it would be projected to $(r_{31}, r_{32})^T$. Hence, the best matrix would be such that has the closest $(r_{31}, r_{32})^T$ to $(x,y)^T$ of the corner, which is associated with the board's corner with coordinates $(0,0)^T$.

However, often the algorithm finds who matrices H such that they have the same r_{31} and r_{32}, but different r_{11}, r_{21}, r_{12}. To find the best matrix, we found the intrinsic values for each of them and back-projected the board's corners using both matrices. Then, we used the one which gave the smallest reprojection error.

Solving for the camera intrinsic parameters

Now, to find the rest of the parameters, we substitute the values, found in the previous step into eq. (4.13) and eq. (4.14). We assumed the number of the division model's parameter to be equal to 2, and the scalar multiplier to be equal to 1 section 3.2.6:

$$\begin{bmatrix} A_1\rho_1^2 & A_1\rho_1^4 & -v_1 \\ C_1\rho_1^2 & C_1\rho_1^4 & -v_1 \\ \cdots & \cdots & \cdots \\ A_N\rho_N^2 & A_N\rho_N^4 & -v_N \\ C_N\rho_N^2 & C_N\rho_N^4 & -v_N \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} B_1 - A_1 \\ D_1 - C_1 \\ \cdots \\ B_N - A_N \\ D_N - C_N \end{bmatrix}, \tag{4.25}$$

where

$$A_i = r_{21}x_i + r_{22}y_i + t_2 \tag{4.26}$$
$$B_i = v_i(r_{31}x_i + r_{32}y_i) \tag{4.27}$$
$$C_i = r_{11}x_i + r_{12}y_i + t_1 \tag{4.28}$$
$$D_i = u_i(r_{31}x_i + r_{32}y_i). \tag{4.29}$$

The solution can be found using the least squares method.

4.2.3 Optimization

The loss function is the sum of the squared reprojection errors between the board and the back-projected corners, which were initially detected:

$$L = \sum_{i=1}^{N} \left\| H^{-1} g_{\lambda_1,\lambda_2}(K^{-1}\mathbf{u_i}) - \mathbf{x_i} \right\|^2. \tag{4.30}$$

For the initial guess, we used the randomly chosen constant small values.

The model converged to the same results compared to the initial parameters, set using the Scaramuzza solver, unless the initial guess was very degenerate (i.e. R was such that the board plane passed through the principal point and all back-projected points were projected onto the same line).

This issue also occurred with random small initial values, due to the best approximation for R which minimizes L when the distance from the back-projected board from measure one was high (i.e., the t was far from the true value) being the degenerate solution. To avoid this issue, we first optimized only the t, until the loss function converged, meaning that t is close to the true value.

However, another issue was that when the board was rotated close to the 180°, R once again converged to the degenerate solution. In order to avoid this issue, we found the solution with the initial θ_z set to value, close to 0° and 180°, and then used the solution which minimized the loss function.

4.3 Additional features detection

Often, not all of the board's corners were detected initially. Firstly, we assumed that the whole board was detected, and imputed the missing points in the board space (section 5.5). Then, we tried extending the board points.

We used the obtained camera parameters to then project the imputed board points into the image space.

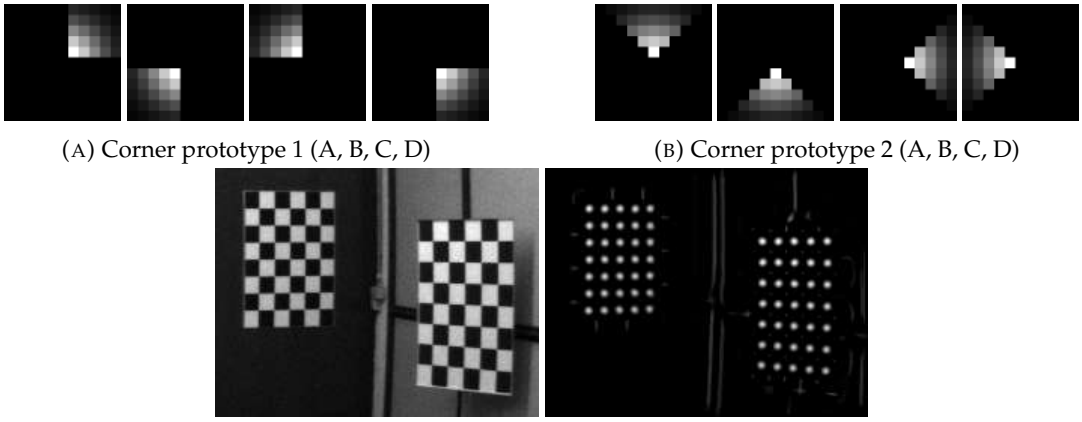
4.4 Classifier

In this work, one of the methods we used was proposed by Geiger et al., 2012 as it didn't require the whole board to be visible, automatically determines the board's number of rows and columns and worked quite well on highly-distorted images.

In this work, we directly used the following steps from the algorithm:

1. Corner detection
2. Sub-pixel corner and orientation refinement

4.4.1 Corner detection



(A) Corner prototype 1 (A, B, C, D) (B) Corner prototype 2 (A, B, C, D)

(C) Input image (D) Corner likelihood

FIGURE 4.1: Corner prototypes (Geiger et al., 2012)

According to Geiger et al., 2012, the following approach proved to be more robust to image clutter and blur than other common choices (Harris and Stephens, 1988; Shi and Tomasi, 2000).

To detect corners in a grayscale image I, the author used two $n \times n$ prototypes for axis-aligned and 45° rotated corners, respectively. Each prototype is constructed using four filter kernels $\{A, B, C, D\}$ (fig. 4.1). The corner likelihood c at each pixel is computed by:

$$\begin{aligned} c &= \max\left(s_1^1, s_2^1, s_1^2, s_2^2\right) \\ s_1^i &= \min\left(\min\left(f_A^i, f_B^i\right) - \mu, \mu - \min\left(f_C^i, f_D^i\right)\right) \\ s_2^i &= \min\left(\mu - \min\left(f_A^i, f_B^i\right), \min\left(f_C^i, f_D^i\right) - \mu\right) \\ \mu &= 0.25\left(f_A^i + f_B^i + f_C^i + f_D^i\right) \end{aligned} \tag{4.31}$$

Then, the authors apply the conservative non-maximum suppression to the response map, and additionally filter the corners by assuring that there are two modes of the gradient directions.

4.4.2 Classifier training

We tested the approach of Geiger et al., 2012, and, alternatively, the Hessian responses for the image, as proposed by Chen and G. Zhang, 2005.

To create a training dataset, we collected the true and false positives from the corners we already detected.

For each of the detected corners on all images, we collected the values of the response function at the previously detected corners, and around them. We didn't collect the values of all of the pixels, to reduce the number of very simple examples.

To determine the threshold for the classification, we used the ROC curve, and found the threshold which maximizes the geometric mean of the true positive rate and the false positive rate.

We used both approaches to train the classifier and compared the results in section 5.6.

Chapter 5

Experiments

5.1 Simulator

We created a simulator to generate distorted points as described in section 3.2.7. We used it to test the solver and test the correctness of projection and back-projection points. However, since the simulator used the same camera model as the solver, the initial camera calibration was perfect.

5.2 Metrics

As noted by Duisterhof et al., 2022, the evaluation of the camera calibration is not straightforward. No ground truth exists for feature detection or camera calibration.

Typically, the reprojection error is used as a metric for the camera calibration, but it depends on multiple factors: type of the calibration pattern, the camera model, and the types of the lens.

As currently the algorithm only supports processing a single checkerboard, we cannot compare to Duisterhof et al., 2022, which uses AprilTags, nor to the Lochman et al., 2021, who provides the detected corners for all of the boards.

The paper's main contribution is finding additional calibration boards' features, which then can be used as an input to any other camera calibration algorithm. The measurements' error is assumed to be Gaussian with zero mean and a constant variance (Hartley and Zisserman, 2004). As calibration process minimizes the geometric distance between the projected and the measured points, it is equivalent to performing the maximum likelihood estimation (Hartley and Zisserman, 2004). As the number of the measurement grows, the estimated camera parameters converge to the true values, hence adding additional measurements improves the calibration (especially from the edge of the image, where the distortion is the highest).

However, the above statement assumes that the measurements are true positives. To evaluate that, we instead artificially removed some of the points from the detections, to ensure that we can recover them. We also created artificially occluded points by overlaying a separate image, as it poses problems for the feature detector.

Lastly, we evaluated the newly detected points on real-world datasets.

5.3 Dataset

For the project, we need highly distorted photos of calibration boards. It takes a lot of work to generate such a dataset, as cameras which produce such images are usually expensive. Therefore, it would be preferable to use an existing dataset.

For this project, we required a highly distorted dataset of chessboard images. We collected several datasets, but the feature detector we used supports detecting

only the boards with the constant tile size, therefore we cannot use AprilGrid nor CharuCO boards. The feature detector is the only limiting factor.

Lochman et al. (2021) collected a wide number of datasets, typically used in the field for the benchmarking of the camera calibration. They're provided in a Deltille *Deltille Detector* 2023 format, and are well documented.

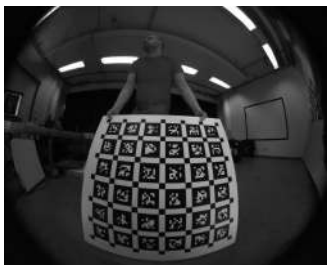
Kalibr (Maye, Furgale, and Roland Siegwart, 2013) contains several established datasets that are commonly used for testing the accuracy of camera calibration frameworks: Double Sphere Usenko, Demmel, and Cremers, 2018, EuRoC Burri et al., 2016, TUM VI Schubert et al., 2018, and ENTANIYA 1 *Calibration of a 250deg Fisheye Lens ů Issue #242 ů Ethz-Asl/Kalibr* 2023. The Kalibr calibration framework was used in the original publications cited above, hence the name of the dataset. As a calibration pattern, AprilGrid with $6\times_6$ tags of 88 mm size was used. In total, the datasets contain approximately 800 images.

OCamCalib (D. Scaramuzza, A. Martinelli, and R. Siegwart, 2006) is a dataset of approximately 300 images. As a calibration pattern, the checkerboard pattern of different sizes was used.

UZH (*Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset | Request PDF* 2023) is a dataset of approximately 800 images collected using the following cameras: As a calibration pattern, AprilGrid with $4\times_5$ tags of 75 mm size was used. The dataset contains approximately 800 images.

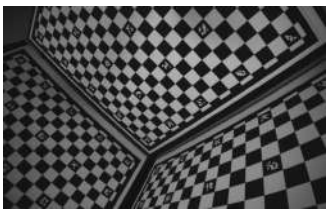
OV (Lochman et al., 2021) is a dataset of approximately 1400 images. It was collected using eight stereo cameras. As a calibration pattern, the checkerboard pattern with 9×6 tags of 22 mm size was used.

Duisterhof et al. (2022) also provide their dataset from the TartanCalib project. Currently, the dataset contains only AprilGrid patterns, as the toolchain doesn't support a chessboard.

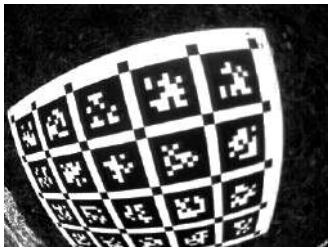


(A) Kalibr

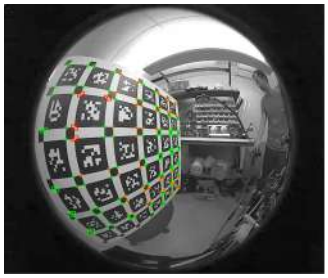
(B) **OCamCalib**



(C) **OV**



(D) UZH



(E) TartanCalib

FIGURE 5.1: Images from the datasets

5.4 Camera calibration

To get accurate predictions for the possibly missing points, we need very accurate camera calibration. We used the reprojection error as a metric for the calibration quality. To calculate the reprojection error, we have to perform the rootfinding section 3.2.6. We assume that all of the points lay within the image, or they get out of the borders a little bit (we use the constant 1.1 of the maximum radius). If the camera calibration is poor, the root-finding might fail. On the initial calibration the number of failed root-findings was much higher.

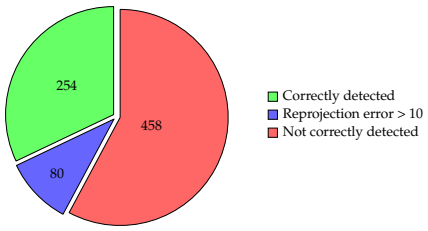


FIGURE 5.2: Initial calibration

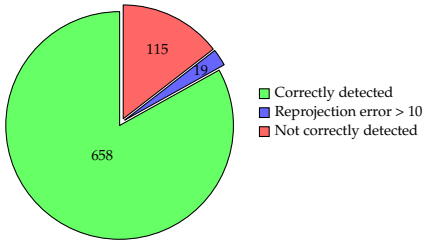


FIGURE 5.3: Final calibration

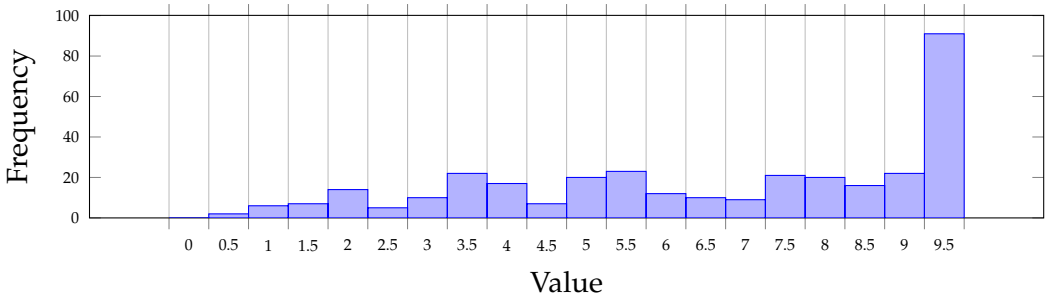


FIGURE 5.4: Initial calibration's reprojection error histogram

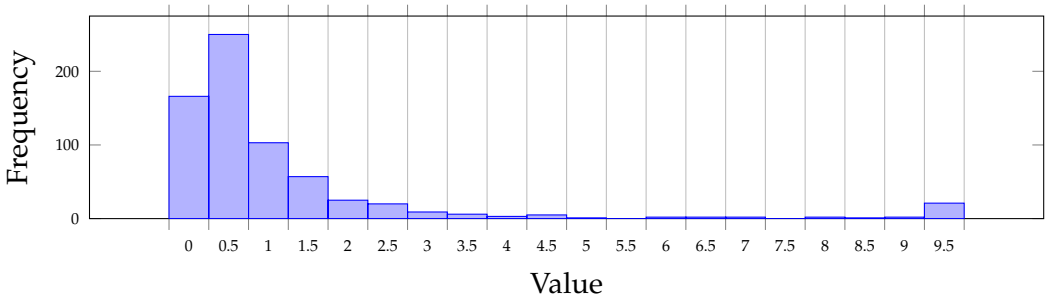
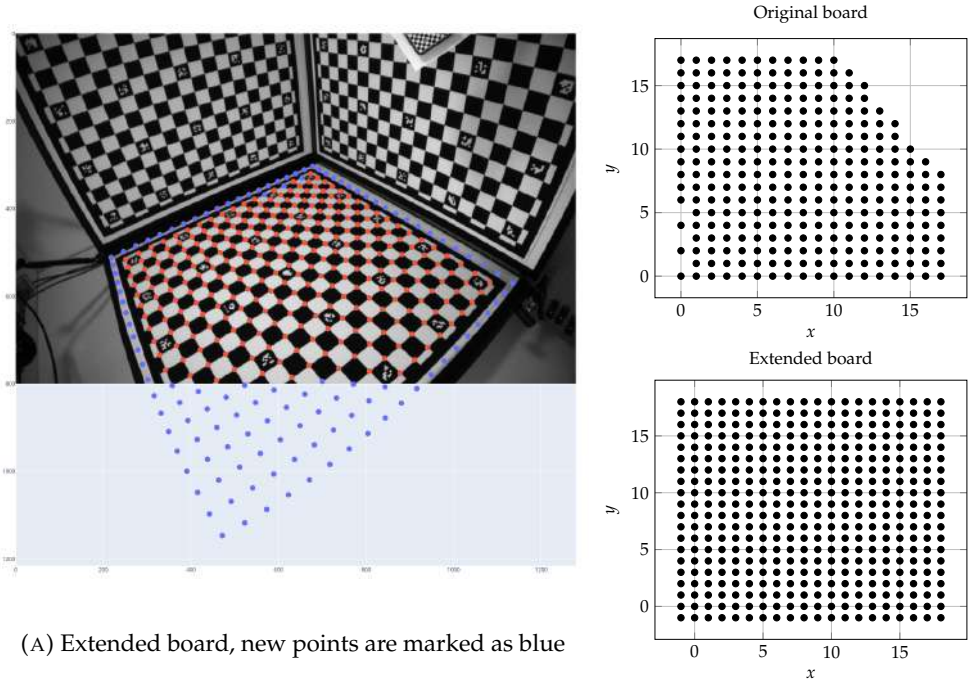


FIGURE 5.5: Final calibration's reprojection error histogram

5.5 Additional features detection

As discussed in section 4.3, we filled the gaps in the originally detected board points, and then padded it for 1 additional element.



(A) Extended board, new points are marked as blue

FIGURE 5.6: Board extension

5.6 Classification

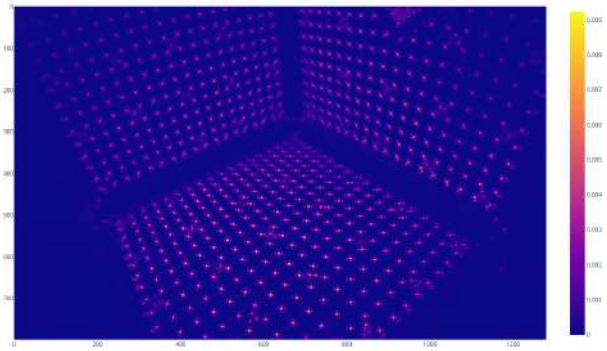


FIGURE 5.7: Hessian response

We compared both responses, as discussed in section 4.4. The Hessian proved to be more robust. The approach, proposed by Geiger et al., 2012 gave too many false positives, especially for the edges.

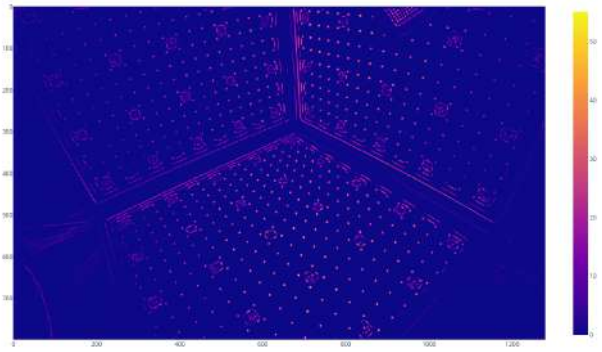
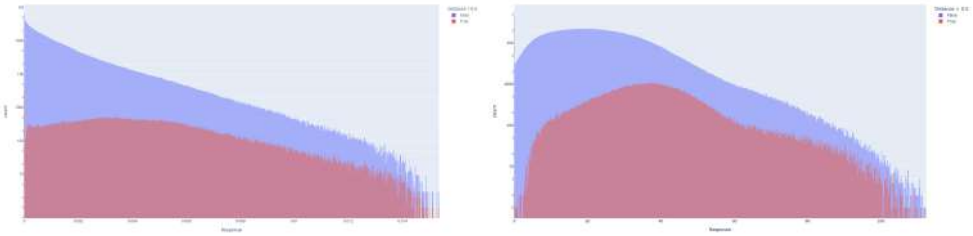


FIGURE 5.8: Response used by Geiger et al., 2012



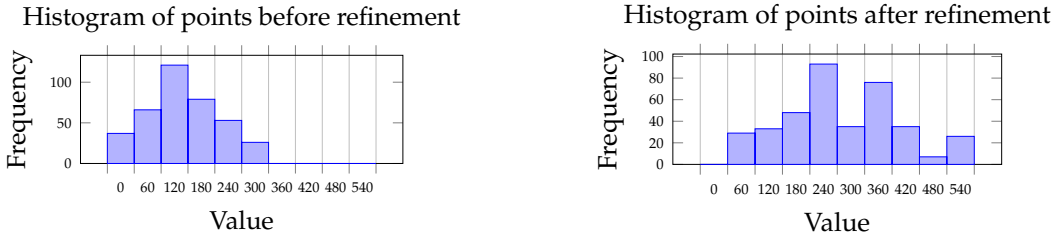
(A) Histogram of Hessian response (B) Histogram of response used by Geiger et al., 2012

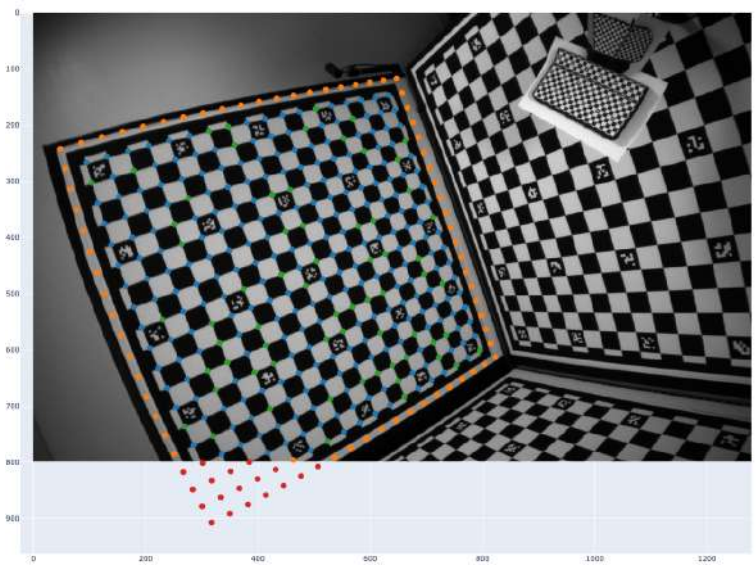
FIGURE 5.9: Distributions of the responses for the image on the section 5.5. Note the log y-scale! Most of the points have the response 0.

5.7 Evaluation

5.7.1 Recovery of artificially removed points

We removed 20% of the points from the original board and then tried to recover them.



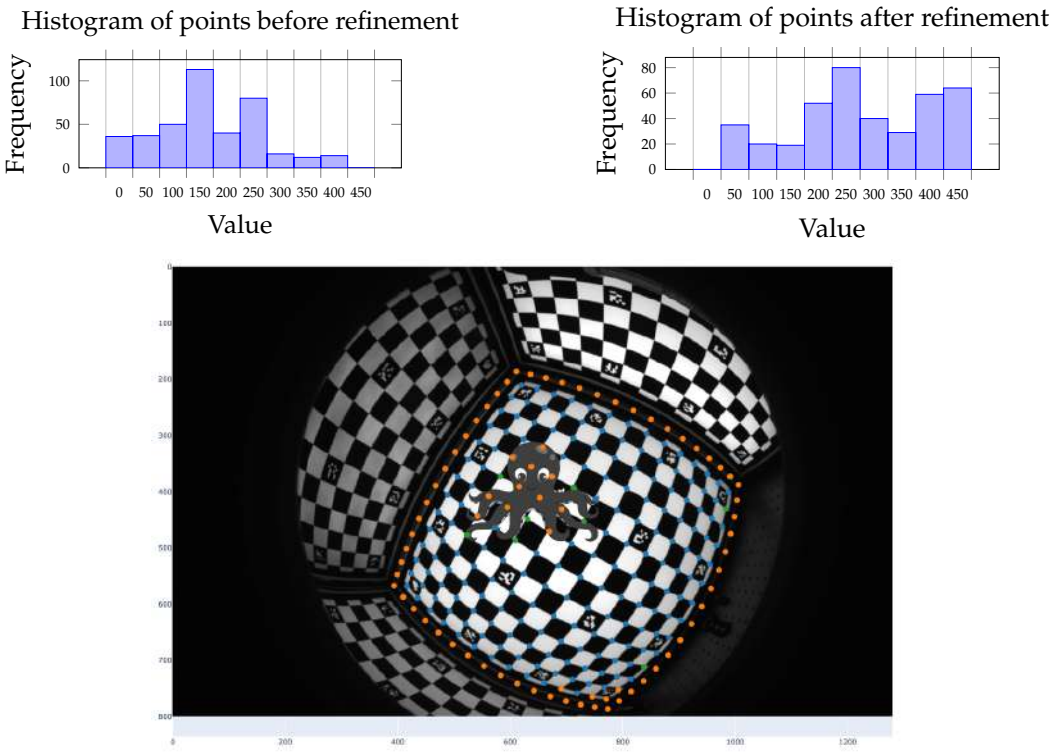


(A) Recovered pruned points (unchanged filtered out new corner out of image)

FIGURE 5.10: Feature refinement on the board with 80% of the points

5.7.2 Performance under occlusion

Occlusions pose additional complications for feature detection. We added an image on top of the board and tried to recover additional points. Often the initial feature detector failed to detect points near the occlusion.



(A) Recovered pruned points (unchanged filtered out new corner out of image)

FIGURE 5.11: Feature refinement on the board with partial board occlusion

5.7.3 Recovery of previously undetected points

Lastly, we recovered the points that were not detected by the initial feature detector. From a visual inspection, most of the points were recovered correctly fig. 5.12. On the top image, for example, there are green points close to the edge. Those corners weren't detected initially due to high distortion. On the bottom image, the algorithm found another row of corners, which weren't detected previously.

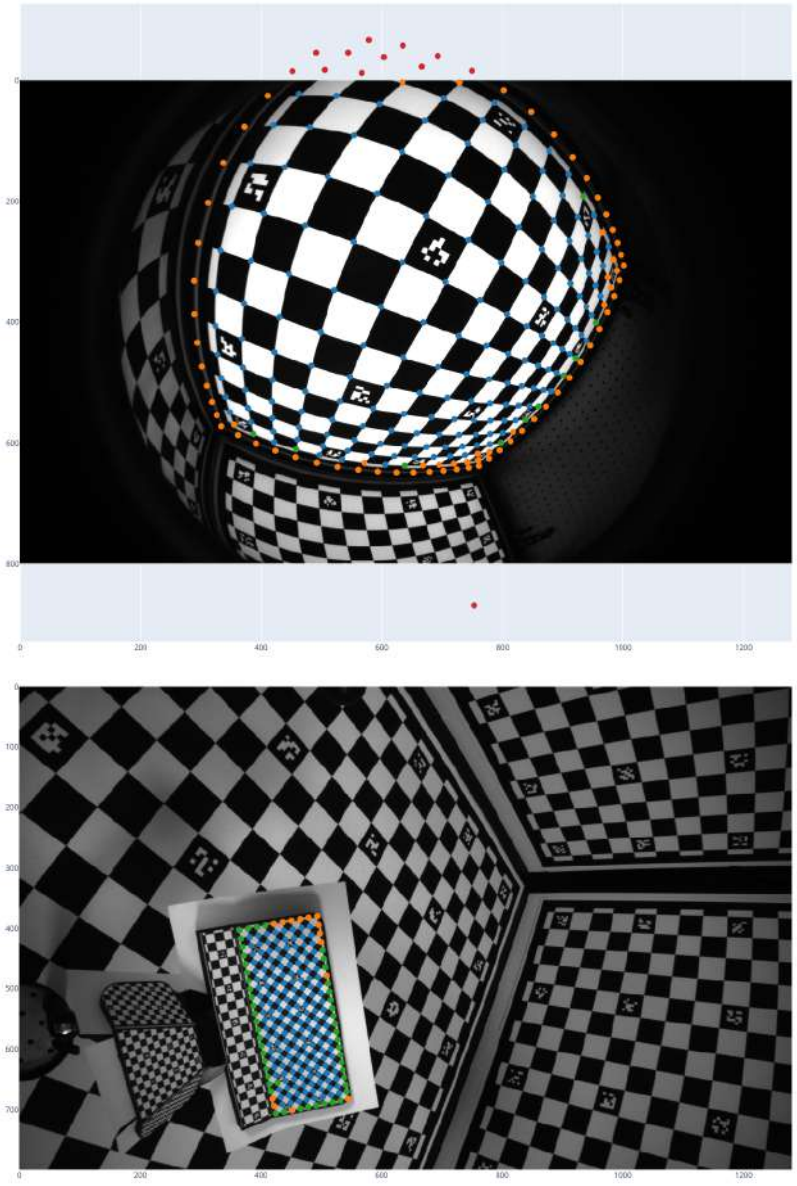
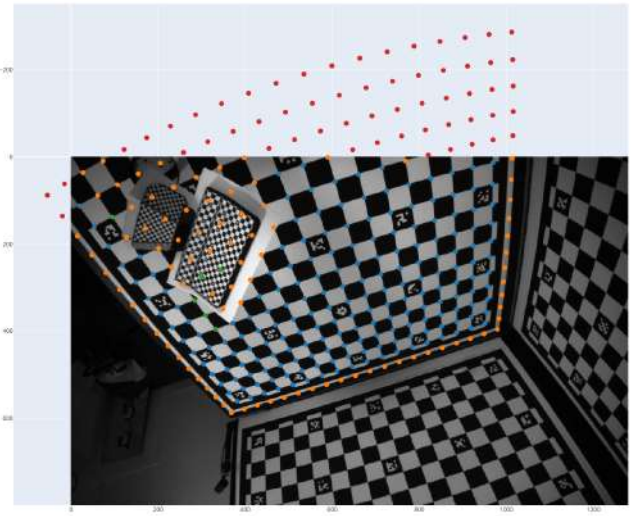
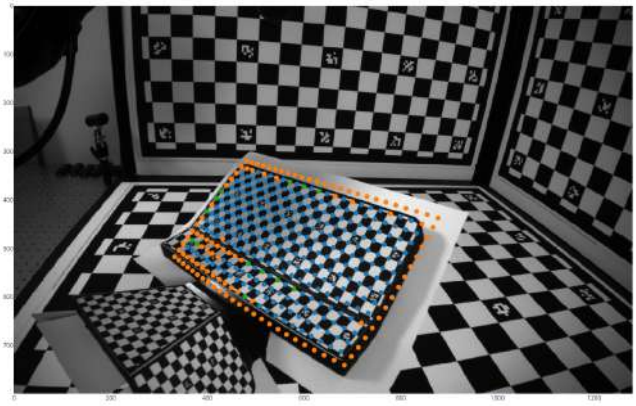


FIGURE 5.12: Recovered points (unchanged filtered out new corner out of image)

However, because of the occlusions, or similar to the board patterns in the background, there were false positives (fig. 5.13). Many images didn't have undetected corners, hence it's hard to make a general statement about the performance of the feature detector solely based on this experiment (fig. 5.14.)



(A) Occlusion on the board with distinct features



(B) Similar pattern to the board in the background

FIGURE 5.13: Examples of false positives

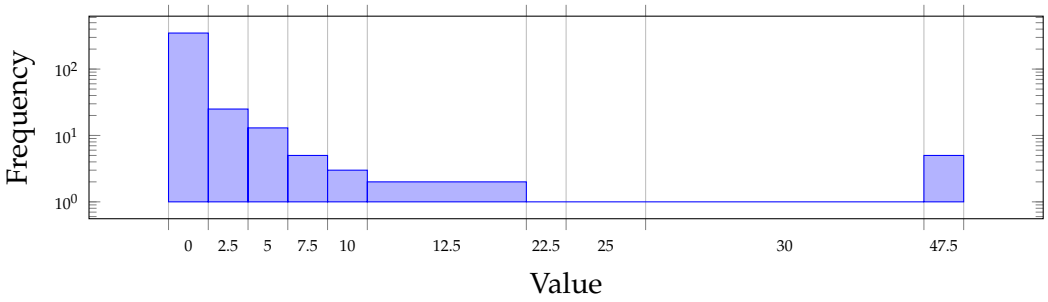


FIGURE 5.14: Histogram of the newly recovered features

Chapter 6

Conclusions

In this thesis, we have presented a new approach for feature detection improvement. We used the feature detector, proposed by Geiger et al., 2012, to detect the initial features. Then, we found the camera calibration for the camera model, proposed by Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart, 2006, and then improved it by minimizing the reprojection error between the back-projection of the detected features and the board. Lastly, we used the improved camera calibration to project the imputed and extended board back into the image and used a binary classification to find the previously undetected features. We tested the method on a real-world dataset and showed that it can improve the feature detection.

6.1 Future work

The method, presented in this thesis can be improved in several ways. First, the feature detection algorithm can be tweaked to use alternating step sizes while constructing the board. It would allow to also match other types of the boars, such as ArUco or AprilTag. Second, the method can be extended to multiple boards on the same image, and multiple images. It would improve the robustness of the camera calibration, and reduce the number of detected points on the wrong board when there is overlap. Lastly, additional improvements can be made to the corner classification algorithm, specifically to the response function. One thing that could work is resizing the image, to find the corners of different sizes.

Bibliography

Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset | Request PDF (2023). URL: https://www.researchgate.net/publication/335144528_Are_We_Ready_for_Autonomous_Drone_Racing_The_UZH-FPV_Drone_Racing_Dataset (visited on 01/30/2023).

Burri, Michael et al. (Jan. 25, 2016). "The EuRoC Micro Aerial Vehicle Datasets". In: *The International Journal of Robotics Research* 35. DOI: 10.1177/0278364915620033.

Calibration of a 250deg Fisheye Lens ů Issue #242 ů Ethz-Asl/Kalibr (2023). GitHub. URL: https://github.com/ethz-asl/kalibr/issues/242 (visited on 01/30/2023).

Chen, Dazhi and Guangjun Zhang (Jan. 1, 2005). "A New Sub-Pixel Detector for X-Corners in Camera Calibration Targets." In: pp. 97–100.

Deltille Detector (Jan. 28, 2023). Meta Archive. URL: https://github.com/facebookarchive/deltille (visited on 01/30/2023).

Devernay, Frédéric and Olivier Faugeras (Aug. 1, 2001). "Straight Lines Have to Be Straight". In: *Machine Vision and Applications* 13.1, pp. 14–24. ISSN: 1432-1769. DOI: 10.1007/PL00013269. URL: https://doi.org/10.1007/PL00013269 (visited on 01/15/2023).

Duisterhof, Bardienus P. et al. (Oct. 5, 2022). *TartanCalib: Iterative Wide-Angle Lens Calibration Using Adaptive SubPixel Refinement of AprilTags*. DOI: 10.48550/arXiv.2210.02511. arXiv: 2210.02511 [cs]. URL: http://arxiv.org/abs/2210.02511 (visited on 01/05/2023). preprint.

Fitzgibbon, A.W. (Dec. 2001). "Simultaneous Linear Estimation of Multiple View Geometry and Lens Distortion". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. Vol. 1, pp. I–I. DOI: 10.1109/CVPR.2001.990465.

Fuersattel, Peter et al. (Mar. 1, 2016). "OCPAD Occluded Checkerboard Pattern Detector". In: pp. 1–9. DOI: 10.1109/WACV.2016.7477565.

Garrido-Jurado, S. et al. (June 1, 2014). "Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion". In: *Pattern Recognition* 47.6, pp. 2280–2292. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.01.005. URL: https://www.sciencedirect.com/science/article/pii/S0031320314000235 (visited on 01/05/2023).

Geiger, Andreas et al. (May 2012). "Automatic Camera and Range Sensor Calibration Using a Single Shot". In: *2012 IEEE International Conference on Robotics and Automation*. 2012 IEEE International Conference on Robotics and Automation, pp. 3936–3943. DOI: 10.1109/ICRA.2012.6224570.

Ha, Hyowon et al. (Oct. 1, 2017). "Deltille Grids for Geometric Camera Calibration". In: pp. 5354–5362. DOI: 10.1109/ICCV.2017.571.

Harris, C. and M. Stephens (1988). "A Combined Corner and Edge Detector". In: *Procedings of the Alvey Vision Conference 1988*. Alvey Vision Conference 1988. Manchester: Alvey Vision Club, pp. 23.1–23.6. DOI: 10.5244/C.2.23. URL: http://www.bmva.org/bmvc/1988/avc-88-023.html (visited on 05/27/2023).

Hartley, Richard and Andrew Zisserman (2004). *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge: Cambridge University Press. ISBN: 978-0-521-54051-3. DOI: 10.1017/CBO9780511811685. URL: https://www.cambridge.org/core/books/multiple-view-geometry-in-computer-vision/0B6F289C78B2B23F596CAA76D3D43F7A (visited on 05/28/2023).

Hu, Danying et al. (July 1, 2019). *Deep ChArUco: Dark ChArUco Marker Pose Estimation*. DOI: 10.48550/arXiv.1812.03247. arXiv: 1812.03247 [cs]. URL: http://arxiv.org/abs/1812.03247 (visited on 01/05/2023). preprint.

Kannala, Juho and Sami Brandt (Sept. 1, 2006). "A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses". In: *IEEE transactions on pattern analysis and machine intelligence* 28, pp. 1335–40. DOI: 10.1109/TPAMI.2006.153.

Krogius, Maximilian, Acshi Haggenmiller, and Edwin Olson (Nov. 2019). "Flexible Layouts for Fiducial Tags". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1898–1903. DOI: 10.1109/IROS40897.2019.8967787.

Lochman, Yaroslava et al. (Oct. 28, 2021). *BabelCalib: A Universal Approach to Calibrating Central Cameras*. DOI: 10.48550/arXiv.2109.09704. arXiv: 2109.09704 [cs]. URL: http://arxiv.org/abs/2109.09704 (visited on 01/05/2023). preprint.

Maye, Jérôme, Paul Furgale, and Roland Siegwart (June 2013). "Self-Supervised Calibration for Robotic Systems". In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013 IEEE Intelligent Vehicles Symposium (IV), pp. 473–480. DOI: 10.1109/IVS.2013.6629513.

Olson, Edwin (May 2011). "AprilTag: A Robust and Flexible Visual Fiducial System". In: *2011 IEEE International Conference on Robotics and Automation*. 2011 IEEE International Conference on Robotics and Automation, pp. 3400–3407. DOI: 10.1109/ICRA.2011.5979561.

OpenCV: Camera Calibration (2023). URL: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (visited on 01/15/2023).

OpenCV: Detection of ArUco Markers (2023). URL: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html (visited on 01/15/2023).

OpenCV: Detection of ChArUco Boards (2023). URL: https://docs.opencv.org/3.4/df/d4a/tutorial_charuco_detection.html (visited on 01/30/2023).

Pritts, James et al. (Nov. 1, 2021). "Minimal Solvers for Rectifying from Radially-Distorted Conjugate Translations". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11, pp. 3931–3948. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2020.2992261. arXiv: 1911.01507 [cs]. URL: http://arxiv.org/abs/1911.01507 (visited on 01/05/2023).

Rosebrock, Adrian (Nov. 2, 2020). *AprilTag with Python*. PyImageSearch. URL: https://pyimagesearch.com/2020/11/02/apriltag-with-python/ (visited on 01/30/2023).

Scaramuzza, D., A. Martinelli, and R. Siegwart (2006). "A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion". In: *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pp. 45–45. DOI: 10.1109/ICVS.2006.3. URL: http://ieeexplore.ieee.org/document/1578733/ (visited on 01/30/2023).

Scaramuzza, Davide, Agostino Martinelli, and Roland Siegwart (Oct. 1, 2006). "A Toolbox for Easily Calibrating Omnidirectional Cameras". In: IEEE International Conference on Intelligent Robots and Systems. DOI: 10.1109/IROS.2006.282372.

Schaffalitzky, Frederik and Andrew Zisserman (Sept. 18, 1998). "Geometric Grouping of Repeated Elements within Images". In: *In Shape, Contour and Grouping in Computer Vision LNCS* 1681. ISSN: 978-3-540-66722-3. DOI: 10.5244/C.12.2.

Schöps, Thomas et al. (June 23, 2020). *Why Having 10,000 Parameters in Your Camera Model Is Better Than Twelve*. DOI: 10.48550/arXiv.1912.02908. arXiv: 1912.02908 [cs]. URL: http://arxiv.org/abs/1912.02908 (visited on 01/05/2023). preprint.

Schubert, David et al. (Oct. 1, 2018). *The TUM VI Benchmark for Evaluating Visual-Inertial Odometry*, p. 1687. 1680 pp. DOI: 10.1109/IROS.2018.8593419.

Shi, Jianbo and Carlo Tomasi (Mar. 3, 2000). "Good Features to Track". In: *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 600. DOI: 10.1109/CVPR.1994.323794.

Usenko, Vladyslav, Nikolaus Demmel, and Daniel Cremers (July 24, 2018). *The Double Sphere Camera Model*.

V. Douskos, I. Kalisperakis, and G. Karras (2007). "Automatic Calibration of Digital Cameras Using Planar Chessboard Patterns". In.

Zhang, Z. (Nov. 2000). "A Flexible New Technique for Camera Calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11, pp. 1330–1334. ISSN: 1939-3539. DOI: 10.1109/34.888718.