sensors

Article

Machine-Learning-Inspired Workflow for Camera Calibration

Alexey Pak [1], Steffen Reichel [2,*] and Jan Burke [1]

[1] Fraunhofer Institute of Optronics, System Technologies, and Image Exploitation IOSB, Fraunhoferstraße 1, 76131 Karlsruhe, Germany
[2] Hochschule Pforzheim, Tiefenbronner Straße 65, 75175 Pforzheim, Germany
* Correspondence: steffen.reichel@hs-pforzheim.de

Abstract: The performance of modern digital cameras approaches physical limits and enables high-precision measurements in optical metrology and in computer vision. All camera-assisted geometrical measurements are fundamentally limited by the quality of camera calibration. Unfortunately, this procedure is often effectively considered a nuisance: calibration data are collected in a non-systematic way and lack quality specifications; imaging models are selected in an ad hoc fashion without proper justification; and calibration results are evaluated, interpreted, and reported inconsistently. We outline an (arguably more) systematic and metrologically sound approach to calibrating cameras and characterizing the calibration outcomes that is inspired by typical machine learning workflows and practical requirements of camera-based measurements. Combining standard calibration tools and the technique of active targets with phase-shifted cosine patterns, we demonstrate that the imaging geometry of a typical industrial camera can be characterized with sub-mm uncertainty up to distances of a few meters even with simple parametric models, while the quality of data and resulting parameters can be known and controlled at all stages.

Keywords: camera calibration; machine learning; active targets; phase shifting; 3D localization

check for updates

Citation: Pak, A.; Reichel, S.; Burke, J. Machine-Learning-Inspired Workflow for Camera Calibration. *Sensors* **2022**, *22*, 6804. https://doi.org/10.3390/s22186804

Academic Editor: Alessandro Artusi

Received: 2 August 2022
Accepted: 31 August 2022
Published: 8 September 2022

1. Introduction

Digital cameras are relatively inexpensive, fast, and precise instruments capable of measuring 2D and 3D shapes and distances. Equipped with $\mathcal{O}(10^6 - 10^8)$ pixels, each sensing $\mathcal{O}(10^2 - 10^3)$ intensity levels in several color channels, and a perspective lens commanding a field of view of $\mathcal{O}(10^1 - 10^2)$ degrees, a typical digital camera may technically resolve features as small as 10^{-3} to 10^{-4} rad. Moreover, by exploiting the prior knowledge of the scene, one may transcend the physical sensor resolution limits and locate extended objects or prominent features to within $\mathcal{O}(10^{-1} - 10^{-2})$ pixels. This impressive sensitivity powers metrological methods, such as laser triangulation and deflectometry, as well as various shape-from-X techniques in the closely related field of computer vision. In some state-of-the-art camera-based measurements, the residual uncertainty reaches tens of nanometers, even with non-coherent illumination [1].

Depending on the employed optical scheme, translation of images to metric statements may be a non-trivial task. In order to take full advantage of data, one needs a sufficiently accurate mathematical model of the imaging process and its uncertainties as well as an adequate calibration procedure to fix all relevant parameters based on dedicated measurements (calibration data). In metrology, once an instrument is calibrated, one naturally should quantify the following properties of the model:

(a) **Consistency:** How well does the calibrated model agree with the calibration data?
(b) **Reproducibility:** Which outcomes should be expected if the calibration were repeated with data collected independently under similar conditions?
(c) **Reliability:** Which uncertainties should one expect from a measurement made in a given application-specific setup that uses the calibrated camera model?

In Bayesian terms, calibration aims to estimate the posterior probability distribution function (PDF) over the model parameters given evidence (calibration data) and prior knowledge (e.g., physical bounds on parameter values). In case such a posterior PDF can be formulated, it may be analyzed using entropy, Kullback–Leibler divergence, and other information-theoretical tools in order to optimally address the points (a–c) above. However, finding true closed-form PDFs for complex systems with multiple parametric inter-dependencies such as a camera may be difficult and/or impractical. Nevertheless, Bayesian methods may be used to analyze, e.g., Gaussian approximations to true PDFs.

A complementary approach adopted in the field of Machine Learning and advocated for in this paper treats camera models as trainable black or gray boxes whose adequacy to the actual devices can be estimated statistically using sufficiently large datasets, interpreted as samples drawn from implicit stationary PDFs. The above quality characteristics (a–c) then may be naturally related to empirical metrics such as loss function values on training/testing/validation datasets, dispersion of outcomes in cross-validation tests, various generalizability tests, etc. In what follows, we provide a simple example of how such techniques may deliver more useful descriptions than the commonly accepted practices while remaining relatively inexpensive to implement. The wider acceptance of similar methods may therefore lead to better utilization of hardware and benefit numerous existing applications. Ultimately, unambiguous and reproducible quality indicators for geometrical camera calibration may even facilitate the introduction of new industrial standards similar to those that exist for photometric calibration [2].

Throughout this paper, we assume the most typical use-case: a compact rigid imaging unit—a camera consisting of optics and a sensor—is placed at various positions in space and acquires data at rest with respect to the observed scene. Light propagates in a transparent homogeneous medium (air, vacuum, or non-turbulent fluid). For simplicity, we exclude "degenerate" imaging geometries with non-trivial caustics of view rays (e.g., telecentric or complex catadioptric systems): such devices usually serve specific purposes, and there exist dedicated methods and separate bodies of literature devoted to their calibration. We further only consider the constellation parameters consistent with geometrical optics, i.e., diffraction and interference effects are assumed negligible. The internal state of the camera is supposed to be fixed throughout all measurements. Finally, we assume that the calibration is based on data collected in a dedicated session with a controlled scene. The end user/application then is ultimately interested in the geometry of uncontrolled scenes recorded under similar conditions, e.g., sizes and shapes of objects and/or positions and orientations of the camera itself.

The purpose of this paper is to provide some background and motivate and experimentally illustrate a practical ML-inspired calibration approach intended for applications in precision optical metrology and advanced computer vision tasks. This procedure and the quality assurance tools it is based on represent our main contribution. In order to keep the focus on the workflow rather than technical details, we deliberately employ a very simple experimental setup (described in Appendix B) and use the standard calibration algorithm by Zhang [3] implemented in the popular OpenCV library [4].

The structure of this paper is as follows. Section 2 briefly mentions a few camera models and calibration techniques often used in practice and outlines typical quality characterization methods. Section 3 introduces the generic notation for camera models and Section 4—for calibration data. After that, Section 5 discusses the quantification of model-to-data discrepancies and schematically explains how calibration algorithms work. Section 6 provides some basic methods of the quantitative calibration quality assessment; Section 7 adjusts these recipes accounting for the specifics of camera calibration requirements. Our proposed workflow is then summarized in Section 8 and subsequently illustrated with an experiment in Section 9. It is thus Sections 7 and 8 that contain novel concepts and algorithms. We discuss the implications of our approach in Section 10 and conclude in Section 11.

2. Typical Calibration Methods, Data Types, and Quality Indicators

The basics of projective geometry may be found, e.g., in [5]. The common theory and methods of high-precision camera calibration are thoroughly discussed in [6], including the pinhole model that virtually all advanced techniques use as a starting point. In what follows, we many times mention the popular Zhang algorithm [3] that minimizes re-projection errors (RPEs) in image space and estimates the parameters of a pinhole model extended by a few lower-order polynomial distortion terms.

In its canonical version, the Zhang algorithm needs a sparse set of point-like features with known relative 3D positions that can also be reliably identified in the 2D images. In practice, one often uses cell corners in a flat checkerboard pattern printed on a rigid flat surface. More sophisticated calibration patterns may include fractals [7] or complex star-shaped features [8] in order to reduce biases and enable more robust feature detection. Advanced detection algorithms may be too complex to allow an easy estimation of the residual localization errors. In order to reduce the influence of these (unknown) errors, one may employ sophisticated procedures such as the minimization of discrepancies between the recorded and inversely rendered pattern images [9]. However, all these improvements leave the sparse nature of datasets intact.

2.1. Calibration with Dense Datasets Produced by Active Target Techniques

As a fundamentally better type of input data, calibration may employ active target techniques (ATTs) [10,11] that use high-resolution flat screens as targets. The hardware setup here is slightly more complex than that with static patterns: a flat screen displays a sequence of special coded patterns (10 s to 100 s of patterns depending on the model and the expected quality) while the camera synchronously records frames. From these images, one then independently "decodes" screen coordinates corresponding to each modulated camera pixel. Note that the decoding does not rely on pattern recognition and that the resulting datasets (3D to 2D point correspondences) are dense. Furthermore, from the same data, one may easily estimate decoding uncertainties in each pixel [12].

For generic non-central projection, Grossberg and Nayar [13] propose a general framework where a camera induces an abstract mapping between the image space and the space of 3D view rays; this is the base of many advanced camera models. As a rule, the calibration of such models needs significantly better quality data than the Zhang algorithm, and ATTs have for a long time been a method of choice, in particular, for the calibration of non-parametric models in optical metrology [10,14–17].

However, even for simpler models, ATTs are long known to outperform the static pattern-based techniques [11]. For the best accuracy, one may relatively easily include corrections for the refraction in the cover glass of the display [18] and other minor effects [17]. In our experiments, ATTs routinely deliver uncorrelated positional decoding uncertainties of order 0.1 mm (less than half of the typical screen pixel size), while the camera-to-screen distances and screen sizes are of order 1 m.

We claim that there are no reasons to not use ATTs as a method of choice also in computer vision (except, perhaps, some exotic cases where this is technically impossible). Even if an inherently sparse algorithm such as Zhang's cannot use full decoded datasets without sub-sampling, one may easily select more abundant and better quality feature points from dense maps at each pose than a static pattern would allow. In practice, sub-sampled ATT data considerably improve the stability of results and reduce biases.

2.2. Quality Assessment of Calibration Outcomes

Regardless of the nature of calibration data, in most papers, the eventual quality assessment utilizes the same datasets as the calibration. This approach addresses the consistency of the outcomes but not their reproducibility and reliability. Very rarely, the calibration is validated with dedicated scenes and objects of known geometry [19].

As an alternative example, consider a prominent recent work [8] from the field of computer vision. Its authors propose novel static calibration patterns along with a dedicated detection scheme and calibrate several multi-parametric models capable of fitting higher-frequency features. The calibration quality is evaluated (and the models are compared) based on dedicated testing datasets, i.e., data not used during the calibration. This is a serious improvement over the usual practice. However, following the established tradition, the authors report the results in terms of RPEs on the sensor measured in pixels. Without knowing (in this case, tens of thousands) the calibrated camera parameters, it is impossible to translate these maps and their aggregate values into actual metric statements about the 3D geometry of the camera view rays. As we demonstrate below, such translation may be quite helpful; the comparison of model-related to data-related uncertainties is a powerful instrument of quality control. Finally, in the absence of dense datasets, the authors of [8] have to interpolate between the available data points in order to visualize artifacts in dense error maps; this step may potentially introduce complex correlations and biases into integral (compounded) quality indicators.

Dense datasets generated by an ATT could have been useful to resolve these issues. Even better, estimates of uncertainties in decoded 3D point positions—a by-product in advanced ATTs—could potentially enable an even more metrologically sound characterization of residual model-to-data discrepancies.

2.3. Note on Deep-Learning-Based Camera Calibration Methods

"Traditional" calibration algorithms such as Zhang's rely on the explicit numerical optimization of some loss function (cf. Section 5.2). As an alternative, some recent works demonstrate that multi-layer neural networks can be trained to infer some camera parameters using, e.g., images of unknown scenes or otherwise inferior-quality data [20–22]. While we acknowledge these developments, we point out that so far they mostly address issues that are irrelevant in the context of this paper. Nevertheless, if at some point a "black-box" solution for ML-based high-quality calibration appears, it may also be trivially integrated into and benefit from our proposed workflow of Section 7.

3. Camera Models and Parameters

The approach presented in this paper is model-agnostic and can be easily applied to arbitrarily flexible discrete and smooth generic parameterizations [6,8,14,23–29]. However, in our experiments and illustrations, we employ the venerable Zhang model, and more specifically, its implementation in the open-source library OpenCV [4,30] (version 4.2.0 as of writing). We also adopt the OpenCV-style notation for the model parameters.

The simplest part of a camera model is its *pose*, or embedding in the 3D world, which includes the 3D position and the orientation of the camera. At a given pose, the transformation between the world and the camera's coordinate systems is described by six *extrinsic* parameters: a 3D translation vector $\vec{t}$ and three rotation angles that we represent as a 3D vector $\vec{u}$. The relevant coordinate systems are sketched in Figure 1. According to this picture, a 3D point P with the world coordinates $\vec{p}^{\,W} = \left(x^W, y^W, z^W\right)^T$ has coordinates $\vec{p}^{\,C} = \left(x^C, y^C, z^C\right)^T$ in the camera's frame that are related by

$$\vec{p}^{\,C} = R(\vec{u})\,\vec{p}^{\,W} + \vec{t}. \tag{1}$$

The 3×3 rotation matrix $R(\vec{u})$ may be parameterized in many ways that may be more or less convenient at a given scenario; the recipe adopted in OpenCV is shown in Equation (A1).

The *direct* projection model $\vec{\pi} = \vec{\Pi}(\vec{p}^{\,C} \mid \vec{\theta})$ then describes the mapping of 3D point coordinates $\vec{p}^{\,C}$ onto 2D pixel coordinates $\vec{\pi} = \left(x^I, y^I\right)^T$ of the projection of point P on the sensor. The number of *intrinsic* parameters $\vec{\theta}$ may be as low as four in the pinhole model in Figure 1 or reach thousands in high-quality metrological models. The basic model $\vec{\Pi}^{(\mathrm{CV})}(\cdot|\cdot)$ implemented in OpenCV has 18 parameters as described in Equation (A2).

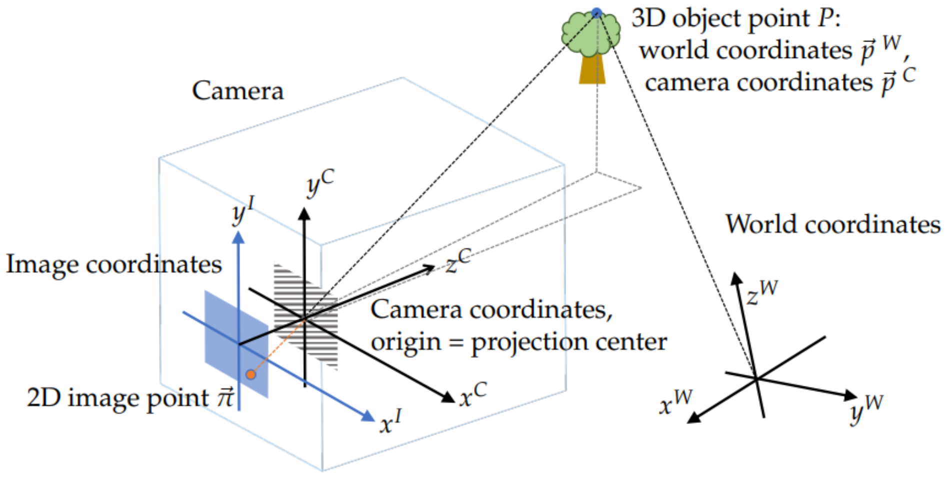
3D object point P:
world coordinates $\vec{p}^{\,W}$,
camera coordinates $\vec{p}^{\,C}$
Camera
World coordinates
Image coordinates
Camera coordinates,
origin = projection center
2D image point $\vec{\pi}$
y^I
x^I
y^C
z^C
x^C
z^W
x^W
y^W

Figure 1. World, camera, and image plane coordinates for a pinhole camera.

In addition to a direct mapping, it is often useful to also introduce an *inverse* projection model $\vec{r} = \vec{R}(\vec{\pi} \mid \vec{\theta})$ with the same parameterization that returns a direction vector $\vec{r}$ for a view ray that corresponds to the pixel $\vec{\pi}$. By definition, the two projection functions must satisfy $\vec{\pi} = \vec{\Pi}(\alpha\vec{R}(\vec{\pi} \mid \vec{\theta}) \mid \vec{\theta})$ for any pixel $\vec{\pi}$ and any scaling factor $\alpha > 0$. Although no closed-form inversion is known for the OpenCV model, Equation (A3) presents a practical way to define the respective inverse projection $\vec{R}^{(\mathrm{CV})}(\cdot|\cdot)$.

Note that the choice of the direct or the inverse projection function to define a camera model for non-degenerate optical schemes is arbitrary and is mostly a matter of convenience. For example, the OpenCV model is often employed for rendering where a succinct formulation of $\vec{\Pi}(\cdot|\cdot)$ is a plus, but many advanced metrological models are formulated in terms of the inverse mapping $\vec{R}(\cdot|\cdot)$. (For non-central projection schemes, the inverse model should also define view ray origins $\vec{o} = \vec{O}(\vec{\pi} \mid \vec{\theta})$ in addition to their directions $\vec{r}$ at each pixel, but this is a relatively straightforward extension [29].) Generally, powerful and highly parallel modern hardware in practice eliminates any real difference between the two formulations for any camera model.

Each component of $\vec{\theta}$ in the OpenCV model has a well-defined physical meaning (cf. the description in Appendix A): the model of Equation (A2) was clearly constructed to be "interpretable". A direct relation between intrinsic parameters and simple lens properties is obviously useful when we, e.g., design an optical scheme to solve a given inspection task. However, when our goal is to fit the imaging geometry of a real camera to an increasingly higher accuracy, we necessarily need more and more parameters to describe its "high-frequency features" [8]—minor local deviations from an ideal smooth scheme. At some point, the convenient behavior of the model in numerical optimization becomes more important than its superficial interpretability. Alternative—less transparent, or even "black-box"—models then may end up being more practical.

We may control the "flexibility" of the OpenCV model by fixing some intrinsic parameters to their default values. For illustration purposes, we devise two "toy models" as follows. The simpler **"model A"** uses only f_x, f_y, c_x, c_y, k_1, k_2, k_3, p_1, and p_2 (cf. Equation (A2)), while keeping the remaining nine parameters fixed to zeros. The full model with all 18 parameters enabled is referred to as **"model B"**. In our tests, the latter has proven to be less stable: in optimization, it often becomes caught in local minima and generally converges more slowly than the "model A", even with high-quality calibration data.

4. Calibration Data Acquisition and Pre-Processing

Cameras are typically calibrated based on a collection of points in 3D and their respective sensor images. Let us denote a pair $(\vec{p}^{\,W}, \vec{\pi})$—the world coordinates and the respective projection onto the sensor for some point-like object—a *record*, a collection

$A = \left\{ \left(\vec{p}_j^{\,W}, \vec{\pi}_j \right) \right\}_{j=1}^{M}$ (dense or sparse) of M records obtained at a fixed camera pose—a *data frame*, and a collection $Q = \{A_i\}_{i=1}^{N}$ of N data frames for various camera poses but identical intrinsic parameters—a *dataset*. Most calibration algorithms expect such a dataset as input; in particular, the central function `calibrateCamera()` in OpenCV [30] receives a dataset Q collected at $N > 3$ distinct poses and fits extrinsic parameters $(\vec{t}_i, \vec{u}_i)$ for each pose i and intrinsic parameters $\vec{\theta}$ that are assumed to be common to all poses.

ATTs collect data in a setup where a camera looks directly at a flat screen. The screen displays a sequence of modulated patterns while the camera captures respective frames. Finally, these images are decoded, and for each camera pixel, we obtain corresponding screen pixel coordinates. The procedure is outlined, e.g., in [29]; our experimental setup is described in Appendix B and is shown with Display 1 in operation in Figure 2.

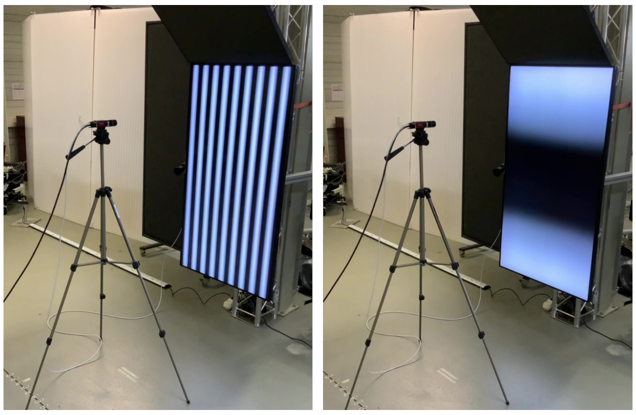

Figure 2. Our experimental set-up. The distance between the camera and the active target (a 55″ Philips monitor) is about 1000 mm. The two panels demonstrate data acquisition with phase-shifted cosine patterns modulated at different directions and different spatial frequencies.

In order to convert screen coordinates to 3D world vectors, we prescribe them a zero z-component, so that each world point is $\vec{p}^{\,W} = \left(x^S, y^S, 0\right)^T$, where x^S and y^S are the decoded screen coordinates. At each pose, we then obtain a dense data frame of M records, where M may be as large as the total number of camera pixels. As a by-product, at each pixel, the decoding algorithm may also estimate $\Delta\vec{p}^{\,W}$—the uncertainty in $\vec{p}^{\,W}$ originating from the observed noise level in the recorded camera pixel values.

Unfortunately, the `calibrateCamera()` function cannot use dense data nor take advantage of estimated uncertainties. We therefore have to extract a sparse subset of valid records from each data frame. To that end, we apply a Gaussian filter with the size of 3 pixels to the decoded coordinate maps and then pick valid pixels in the nodes of a uniform grid in the image space. We found that a 100 × 100 grid (providing at most 10,000 valid records per frame) ensures rather stable and robust convergence of calibration, while its runtimes remain under ten minutes on a modern CPU. Collecting equivalent-quality data with static (checkerboard) patterns would be very challenging.

A complete calibration dataset then includes several (normally 10–30) such sub-sampled sparse data frames recorded at different camera poses. Our experimental datasets span 3 to 4 different distances between the camera and screen and include typically 4 or more poses at each distance; for details, see Appendix B.

An example of a dense decoded data frame before filtering and sub-sampling is shown in Figure 3. Non-modulated pixels are identified during the decoding and are displayed as the white space in the maps. The panel (a) shows the decoded values x^S, the panel (c)—estimated uncertainties Δx^S. As should be expected, the high-frequency "jitter" in the coordinates that is visible, e.g., in the cutout 1D profile (b), has a typical amplitude that is close to the respective estimates in (d). In this particular case, the mean decoding uncertainty is about 0.09 mm, which is roughly the same as the screen pixel size of our

Display 2 (that was used to collect these data). Note that the decoding works even with a partial view of the target; by contrast, many simple checkerboard detectors require a full unobstructed view of the entire pattern. This may lead to systematic undersampling and reduce the reliability of calibrated models near the frame boundaries [7].

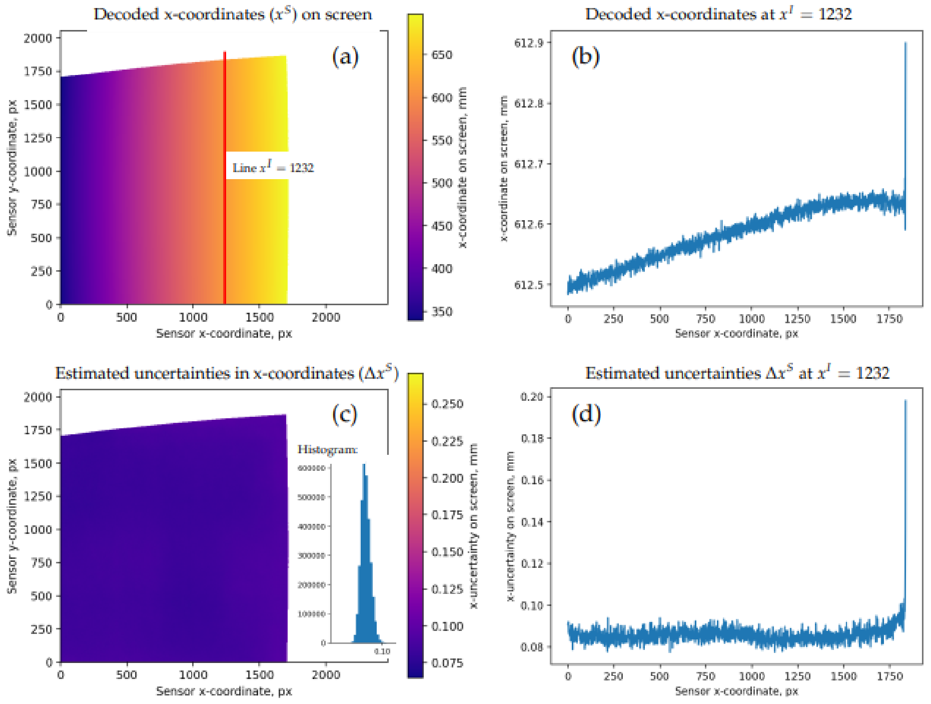

Figure 3. Sample data frame (dataset 4, pose 17). (**a**): map of the decoded x^S (x-coordinates on the screen); (**b**): profile of x^S along the line $x^I = 1232$ (indicated with a red line in (**a**); note that $x^I$and y^I refer to pixel coordinates in the recorded images); (**c**): estimated decoding uncertainties Δx^S and their histogram (number of pixels vs Δx^S); (**d**): profile of Δx^S along the line $x^I = 1232$. Similar maps are also available for y^S (decoded y-coordinates on the screen).

5. Data-to-Model Consistency Metrics

Consider a camera with intrinsic parameters $\vec{\theta}$ and extrinsic parameters $\vec{t}$, $\vec{u}$. The consistency of a model $\vec{\Pi}(\vec{p}^{\,C} \mid \vec{\theta})$ to a record $(\vec{p}^{\,W}, \vec{\pi})$ may be evaluated in the image space using point-wise vector- and scalar-valued *re-projection errors* (RPEs):

$$\begin{aligned} \vec{D}_{\text{RPE}}\left(\vec{p}^{\,W}, \vec{\pi} \mid \vec{\theta}, \vec{t}, \vec{u}\right) &= \vec{\Pi}\left(R(\vec{u})\,\vec{p}^{\,W} + \vec{t} \mid \vec{\theta}\right) - \vec{\pi}, \\ D_{\text{RPE}}\left(\vec{p}^{\,W}, \vec{\pi} \mid \vec{\theta}, \vec{t}, \vec{u}\right) &= \left\|\vec{D}_{\text{RPE}}\left(\vec{p}^{\,C}, \vec{\pi} \mid \vec{\theta}, \vec{t}, \vec{u}\right)\right\|. \end{aligned} \tag{2}$$

In order to aggregate point-wise RPEs over a data frame $A = \left\{\left(\vec{p}_j^{\,W}, \vec{\pi}_j\right)\right\}_{j=1}^{M}$ we may define the respective "root mean squared" (RMS) value:

$$D^2_{\text{DF RMS RPE}}\left(A \mid \vec{\theta}, \vec{t}, \vec{u}\right) = \frac{1}{M} \sum_{j=1}^{M} D^2_{\text{RPE}}\left(\vec{p}_j^{\,W}, \vec{\pi}_j \mid \vec{\theta}, \vec{t}, \vec{u}\right). \tag{3}$$

Similarly, for a dataset $Q = \{A_i\}_{i=1}^{N}$ we may define

$$D^2_{\text{DS RMS RPE}}\left(Q \mid \vec{\theta}, \{\vec{t}_i, \vec{u}_i\}_{i=1}^{N}\right) = \frac{1}{\sum_i M_i} \sum_{i=1}^{N} \sum_{j=1}^{M_i} D^2_{\text{RPE}}\left(\vec{p}_{ij}^{\,W}, \vec{\pi}_{ij} \mid \vec{\theta}, \vec{t}_i, \vec{u}_i\right), \tag{4}$$

where M_i is the number of records in the frame A_i, and the normalization is chosen in order to match the convention adopted in the `calibrateCamera()` function. Figure 4 shows residual RPEs for the pose 13 of the dataset 0 upon the calibration of our "model A".

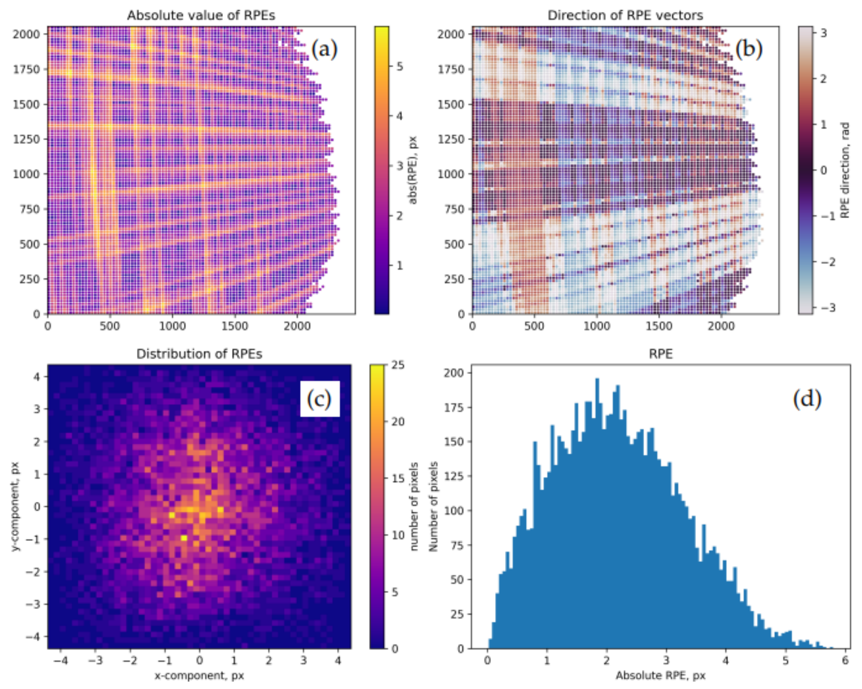

Figure 4. Residual point-wise RPEs for the model A calibrated over the dataset 0, pose 13. The values are shown at sub-sampled pixels. (**a**): D_{RPE} mapped over the sensor. The visible low-frequency Moiré structure results from the interference between the pixel grids of the camera and the screen. (**b**): direction angles $\tan^{-1}\left((\vec{D}_{\mathrm{RPE}})_y/(\vec{D}_{\mathrm{RPE}})_x\right)$. RPEs here demonstrate significant correlations (a systematic bias) which may be partially explained by the same Moiré effect. (**c**): 2D histogram of point-wise vectors $\vec{D}_{\mathrm{RPE}}$. (**d**): histogram of D_{RPE} values. The ragged contour of the valid region in this data frame is due to the oblique view angle of the camera; the modulation contrast of the observed patterns falls below the threshold for the remote parts of the screen.

5.1. Forward Projection Errors

However useful, RPEs have limitations. Most importantly, they are defined in terms of pixels. Effective pixel sizes can often be re-defined by software or camera settings and do not trivially correspond to any measurable quantities in the 3D world. Pixel-based discrepancies are justified when the uncertainties in data are also naturally represented in pixels—for example, when the detections are based on pattern recognition, as is the case with checkerboards and cell corners. Decoding errors in ATTs, however, are defined in length units (e.g., meters) in the 3D space and cannot be directly related to RPEs.

Therefore, it appears useful in addition to RPEs to also define *forward projection errors* (FPEs). Using the same notation as in Equation (2), point-wise FPEs can be defined as

$$\begin{aligned}
&\vec{D}_{\mathrm{FPE}}\left(\vec{p}^{\,W}, \vec{\pi} \mid \vec{\theta}, \vec{t}, \vec{u}\right) = \vec{p}^{\,E}(\vec{\pi} \mid \vec{\theta}, \vec{t}, \vec{u}) - \vec{p}^{\,W} \quad \text{and} \\
&D_{\mathrm{FPE}}\left(\vec{p}^{\,W}, \vec{\pi} \mid \vec{\theta}, \vec{t}, \vec{u}\right) = \left\|\vec{D}_{\mathrm{FPE}}\left(\vec{p}^{\,C}, \vec{\pi} \mid \vec{\theta}, \vec{t}, \vec{u}\right)\right\|, \quad \text{where} \\
&\vec{p}^{\,E}(\vec{\pi} \mid \vec{\theta}, \vec{t}, \vec{u}) = (x^E, y^E, z^E)^T = \vec{o}^{\,W} + \alpha\, \vec{r}^{\,W} \quad \text{is the expected hit point on the target,} \\
&\vec{o}^{\,W} = -R(\vec{u})^T\, \vec{t} \quad \text{is the projection center in world coordinates, and} \\
&\vec{r}^{\,W} = R(\vec{u})^T\, \vec{R}(\vec{\pi} \mid \vec{\theta}) \quad \text{is the view ray direction in world coordinates.}
\end{aligned} \tag{5}$$

The scaling factor α here is found as the solution of the linear equation $z^E = 0$, which encodes our assumption that the (flat) active target is located in the plane $z^W = 0$.

In plain words, Equation (5) may be interpreted as follows: We emit a view ray from $\vec{o}^{\,W}$ along the direction $\vec{r}^{\,W}$ that corresponds to the pixel $\vec{\pi}$ according to our inverse projective mapping $\vec{R}(\cdot|\cdot)$ and find its intersection $\vec{p}^{\,E}$ with the canonical screen plane $z^W = 0$. The discrepancy on the screen between $\vec{p}^{\,E}$ and the actual decoded point $\vec{p}^{\,W}$ then determines a 3D (effectively a 2D) vector $\vec{D}_{\text{FPE}}$. FPEs are defined in physical length units and may be directly compared with the estimated decoding uncertainties $\Delta\vec{p}^{\,W}$ when the latter are available. By analogy to Equations (3) and (4), we may also trivially define aggregate values $D_{\text{DF RMS FPE}}(\ldots)$ and $D_{\text{DS RMS FPE}}(\ldots)$ over data frames and datasets.

Figure 5 shows FPE maps for the same model and data frame as Figure 4. Qualitatively, FPE distributions in this case look similar to those of RPEs, but their values can now be interpreted without any reference to sensor parameters and model details.

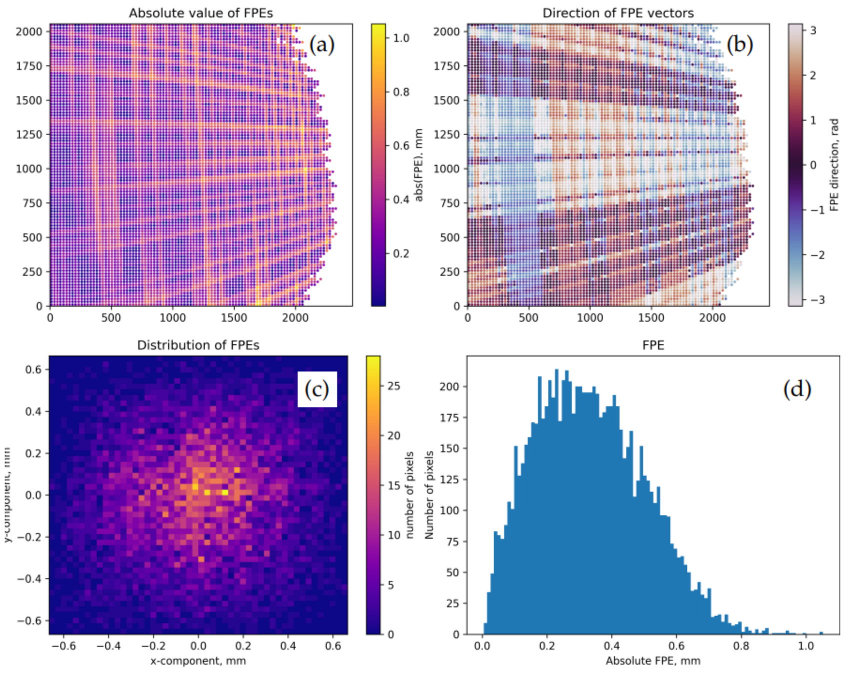

Figure 5. Residual FPEs for the same model and data frame as in Figure 4. (**a**): D_{FPE} mapped over the sensor. (**b**): direction angles $\tan^{-1}\left((\vec{D}_{\text{FPE}})_y/(\vec{D}_{\text{FPE}})_x\right)$. (**c**): 2D histogram of vectors $\vec{D}_{\text{FPE}}$. (**d**): histogram of D_{FPE} values.

One possible reason why FPEs have been less popular than RPEs in practice so far is their dependence on the inverse camera mapping $\vec{R}(\cdot|\cdot)$, whose evaluation may be more computationally expensive than the direct projection. However, as mentioned above, modern hardware increasingly tends to obsolesce this argument.

If the assumption of a flat target is inconvenient or inapplicable in a given setup, one may easily modify the definition Equation (5) as necessary. Furthermore, it is possible to combine FPEs and estimated decoding uncertainties into dimensionless "weighted" error metrics. The latter then optimally exploit the available information and represent the best minimization objective for metrological calibration algorithms [29].

5.2. Calibration Algorithms

The work principle of most calibration algorithms is to find model parameters that minimize some model-to-data consistency metric. In particular, assuming the definitions of Equations (2) and (4), the Zhang algorithm may be formulated as the following optimization problem. Given a dataset $Q = \{A_i\}_{i=1}^N$, it finds

$$\vec{\Theta}^* \equiv \left(\vec{\theta}^{\,*}, \{\vec{t}_i^{\,*}, \vec{u}_i^{\,*}\}_{i=1}^N\right) = \operatorname{argmin}_{\left(\vec{\theta}, \{\vec{t}_i, \vec{u}_i\}_{i=1}^N\right)} D^2_{\text{DS RMS RPE}}\left(Q \mid \vec{\theta}, \{\vec{t}_i, \vec{u}_i\}_{i=1}^N\right), \quad (6)$$

where the camera model $\vec{\Pi}(\cdot|\cdot) = \vec{\Pi}^{(\text{CV})}(\cdot|\cdot)$ in the definition of $D_{\text{DS RMS RPE}}$ is the OpenCV model Equation (A2) and $\vec{\theta}$ represents the selected subset of intrinsic parameters affected by the optimization (i.e., those that are not fixed to their default values).

Note that Equation (6) treats all the records in the dataset Q equally. This is equivalent to implicitly prescribing the same isotropic uncertainty $\Delta\vec{\pi}$ to the projected sensor coordinates at all calibration points. In terms of imaging geometry, this means that detections of similar quality (metric uncertainty) that originate at 3D points further away from the camera constrain the model more strongly than those that are located near the camera. This effect may introduce a non-trivial bias into calibrated models.

Alternatively, it is possible to use Equation (5) or similar definitions in order to design calibration algorithms that minimize FPEs instead of RPEs [14]. Such approach is in fact preferable for metrological applications, but we refrain from discussing it here.

6. Characterization of Calibration Quality

Let us imagine that a calibration algorithm such as Equation (6) returns not only the most likely parameters $\vec{\Theta}^*$ but also the complete posterior PDF $p(\vec{\Theta} \mid Q)$. For a sufficiently well-behaved model and high-quality data, such a (in general case, intractable) PDF may be expected to have a high "peak" at $\vec{\Theta}^*$, and therefore we may reasonably well approximate it with a Gaussian $\mathcal{N}(\vec{\Theta} \mid \vec{\mu}_\Theta, \Sigma_\Theta)$ that has some central point $\vec{\mu}_\Theta = \vec{\Theta}^*$ and a covariance matrix Σ_Θ. If we further assume that the true $\vec{\mu}_\Theta$ and Σ_Θ are known, the calibration quality aspects (a–c) formulated in Section 1 could be addressed as follows.

6.1. Consistency

It is a common practice to inspect residual RMS values and per-pose maps of RPEs and FPEs computed for $\vec{\Theta}^*$ such as in Figures 4 and 5. In the best case, typical FPEs should match the level of uncertainties $\Delta\vec{p}^{\,W}$ if these are available. RPEs, in turn, may be compared with the diffraction-related blurring scale for the optics measured in pixels. For example, in our experiments, the size of the diffractive spot on the sensor is about 4 μm, which corresponds to 1–2 pixels. A significantly higher level of RPEs/FPEs and the presence of prominent large-scale non-uniformities in per-pose error maps may indicate data collection issues (cf. Moiré structures in Figures 4 and 5), convergence problems in optimization, or an excessively "stiff" model (the situation known as "underfitting").

6.2. Reproducibility

Let us assume the following splitting of the calibration state vector $\vec{\Theta}$ and the parameters of the respective Gaussian posterior PDF:

$$\vec{\Theta} = \begin{pmatrix} \vec{\theta} \\ \vec{\gamma} \end{pmatrix}, \quad \vec{\mu}_\Theta = \begin{pmatrix} \vec{\mu}_\theta \\ \vec{\mu}_\gamma \end{pmatrix}, \quad \text{and } \Sigma_\Theta = \begin{pmatrix} \Sigma_\theta & \Sigma_{\theta\gamma} \\ \Sigma_{\theta\gamma}^T & \Sigma_\gamma \end{pmatrix}, \tag{7}$$

where $\vec{\theta}$ represents intrinsic camera parameters, $\vec{\gamma}$ collectively denotes all per-pose extrinsic parameters, and Σ_θ and Σ_γ are some symmetric positive-definite matrices. The off-diagonal block $\Sigma_{\theta\gamma}$ captures the correlations between $\vec{\theta}$ and $\vec{\gamma}$. As discussed below, these correlations are typically hard to estimate reliably, and we ignore them.

Only $\vec{\theta}$ may be compared between independent calibration attempts since poses are chosen each time anew. We propose to use a Gaussian distribution $\mathcal{N}(\vec{\theta} \mid \vec{\mu}_\theta, \Sigma_\theta)$ as the induced posterior PDF over the expected calibration outcomes. This form is equivalent to the full PDF marginalized over $\vec{\gamma}$ and is consistent with the absence of any additional information that could constrain the model parameters. In case such information does exist (in the form of, e.g., independent measurements of camera positions by an external sensor), one should modify this rule and, e.g., implement some form of conditioning.

The consistency of some set of intrinsic parameters $\vec{\theta}^{\,\prime}$ with the current calibration results then may be estimated with the help of Mahalanobis distance $D_M(\vec{\theta}^{\,\prime} \mid \vec{\mu}_\theta, \Sigma_\theta)$ defined in Equation (A4) and the respective plausibility level $\text{P}_M(\vec{\theta}^{\,\prime} \mid \vec{\mu}_\theta, \Sigma_\theta)$ of Equation (A6).

6.3. Reliability

If one intends to use the calibrated camera model in geometric measurements, its crucial physical characteristic is the expected 3D uncertainty of the view rays induced by the uncertainty in the model parameters. Let us assume that the inverse camera model $\vec{R}(\cdot|\cdot)$ returns a vector $\vec{r} = (r_x, r_y, 1)^T$ whose third component is fixed at unity similarly to our definition of the inverse Zhang model in Equation (A3). Then, the function

$$\rho(\vec{\pi} \mid \vec{\theta}, \Sigma_\theta) = \sqrt{\mathrm{Tr}\Big(\Sigma_r(\vec{\pi} \mid \vec{\theta})\Big)}, \text{ where} \tag{8}$$
$$\Sigma_r(\vec{\pi} \mid \vec{\theta}, \Sigma_\theta) = J(\vec{\pi} \mid \vec{\theta})\, \Sigma_\theta\, J(\vec{\pi} \mid \vec{\theta})^T \text{ and } J(\vec{\pi} \mid \vec{\theta}) = \frac{\partial \vec{R}(\vec{\pi} \mid \vec{\theta})}{\partial \vec{\theta}}$$

describes the scalar projected uncertainty of the view rays at their intersection with the plane $z^C = 1$. In other words, $\rho(\vec{\pi} \mid \vec{\theta}, \Sigma_\theta)$ defines a map of "expected FPE gains" (EFPEGs). These may be interpreted as expected FPEs (EFPEs) evaluated over a virtual screen orthogonal to the camera's axis and displaced by 1 m from the projection center.

Obviously, for any central model, the uncertainty of view ray positions is linearly proportional to the distance from the camera. That is, an EFPE for a view ray corresponding to some pixel $\vec{\pi}$ on the plane $z^C = z$ in the camera's frame is

$$D_{\mathrm{EFPE}}(\vec{\pi} \mid \vec{\theta}, \Sigma_\theta, z) = z\, \rho(\vec{\pi} \mid \vec{\theta}, \Sigma_\theta). \tag{9}$$

Thus, a single map of EFPEGs over the sensor is sufficient to derive calibration uncertainty-related errors for any scene of interest. In particular, we can derive EFPEs for the actual calibration targets and compare them with the respective residual FPEs. If EFPEs significantly exceed residual errors, the model may be "under-constrained", i.e., the calibration dataset is too small or the camera poses in it are insufficiently diverse.

7. Calibration Workflow and Data Management

The approach of Section 6 allows us, in principle, to fully characterize the quality of calibration. This section focuses on obtaining the best estimates of $\vec{\mu}_\theta$ and Σ_θ in practice.

Given a dataset Q with N data frames and a camera model, one may simply calibrate the latter using all N poses in order to obtain the best-fit parameters $\vec{\Theta}^{(ini)}$. We denote this step as *initial calibration*. In addition to $\vec{\Theta}^{(ini)}$, many calibration tools in fact also estimate their stability in some form. For example, the extended version of the `calibrateCamera()` function in OpenCV (overloaded under the same name in the C++ API, `calibrateCameraExtended()` in the Python API [30]) returns the estimated variances $\delta^2\vec{\Theta}^{(ini)}$ for the individual components of $\vec{\Theta}^{(ini)}$. From these, one may recover a covariance matrix $\Sigma_\Theta^{(ini)} = \mathrm{diag}(\delta^2\vec{\Theta}^{(ini)})$. This diagonal form is a conservative estimate of the true variability of outcomes that assumes no available knowledge about possible correlations between the errors in different components of $\vec{\Theta}^{(ini)}$. Covariances of parameters are hard to estimate reliably since they in practice need many more data frames than what is typically available and are not returned by OpenCV.

The values $\delta^2\vec{\Theta}^{(ini)}$ are found (at high added computational cost) from the approximate Hessian of the objective function in Equation (6) and the residual RPEs at the optimum, which is the standard approach for non-linear least square problems [31]. Unfortunately, in practice, this method may wildly under- or overestimate the variability of parameters depending on the lens properties, constellation geometries, and input data quality.

For example, Figure 6 shows EFPEs computed according to Equation (9) for the same calibration target as Figure 5 based on the output from `calibrateCamera()`. We see that this projection seriously underestimates the actual deviations, and $\Sigma_\Theta^{(ini)}$ is thus unreliable.

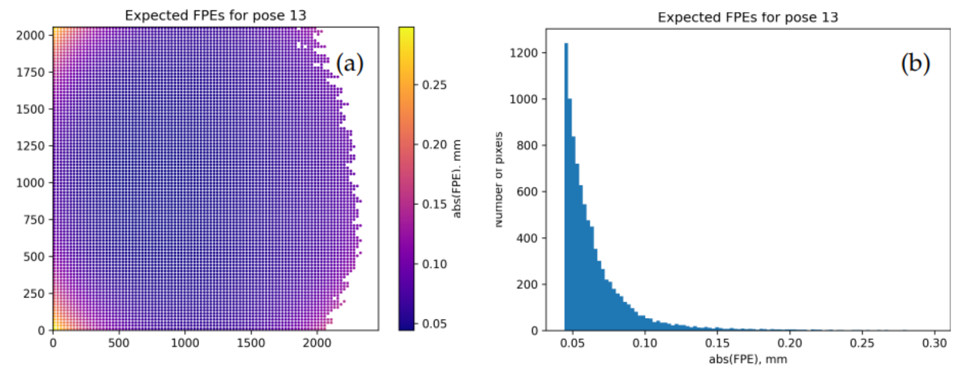

Figure 6. Expected FPEs (EFPEs) for the same model and camera pose as in Figure 5 based on $\Sigma_{\Theta}^{(ini)}$. (**a**): map of point-wise values D_{EFPE} over the sensor. (**b**): histogram of EFPE values.

7.1. Rejection of Outlier Data Frames

The data collection process is not trivial, and it is possible for some data records to have a significantly poorer quality than others. This may happen due to, e.g., a bad choice of calibration poses, illumination changes or vibration during the data acquisition, etc. In the context of ATTs, such quality degradation typically affects entire data frames—adjacent pixels in a frame are not likely to feature significantly different error levels.

If the decoded dataset contains per-record uncertainty estimates as in Figure 3, the latter will reflect any decoding problems, and an uncertainty-aware calibration tool may then automatically detect bad data and ignore them in the fits. Unfortunately, the Zhang method in Equation (6) cannot detect faulty frames and the latter, if present, may randomly affect the optimization outcomes. We therefore must detect and remove such "outlier frames" manually before we produce the final results and conclusions.

To that end, we analyze the residual per-pose RPEs after the initial calibration:

$$E_i^{(ini)} = D_{\text{DF RMS RPE}}(A_i \mid \vec{\theta}^{\,(ini)}, \vec{t}_i^{\,(ini)}, \vec{u}_i^{\,(ini)}) \tag{10}$$

for the data frames A_i and the respective camera parameters $\vec{\theta}^{(ini)}$, $\vec{t}_i^{\,(ini)}$, and $\vec{u}_i^{\,(ini)}$ ($i = 1, ..., N$) extracted from $\vec{\Theta}^{(ini)}$. We expect that the RPEs for the "good" frames will "cluster" together, while the "bad" frames will demonstrate significantly different residual errors. For example, Figure 7 shows residual per-pose RPEs for all 29 poses in our dataset 0. Indeed, the values appear to group together except for a few points. In this particular case, the "problematic" frames correspond to the camera being placed too close to the screen (at about 10 cm); the finite entrance aperture of the lens and the mismatching focus then lead to a high dispersion of the decoded point coordinates.

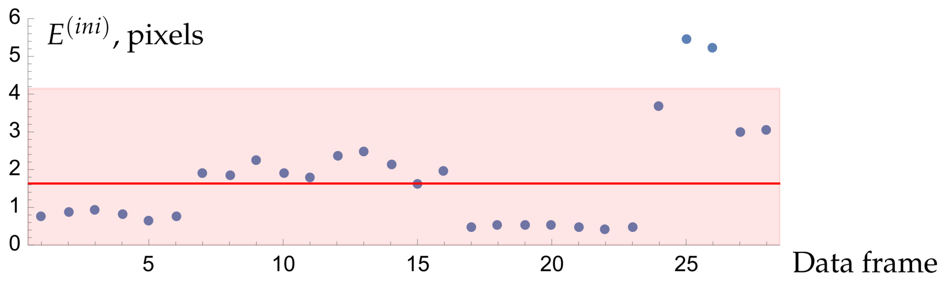

Figure 7. Residual per-pose values $E_i^{(ini)}$ of Equation (10) for the model A calibrated over all 29 frames in the dataset 0. The red line indicates the median of all values and the shaded region denotes the acceptance bounds according to the modified Z-score method with the threshold of 2.0.

The most well-known formal outlier detection techniques are based on Z-scores, modified Z-scores, and interquartile ranges. In our code, we use the modified Z-score method with the threshold of 2.0 since it is more stable for small samples and allows an easy interpretation. The shaded region in Figure 7 shows the acceptance bounds dictated by this method, according to which the frames 25 and 26 must be declared outliers. However, one is free to use any reasonable alternative methods and thresholds. In what follows, we assume that our dataset Q is free from such outliers.

7.2. Empirical Evaluation of Calibration Quality

As discussed in Section 6.1, residual per-dataset and per-pose errors as in Figure 7, as well as maps such as in Figure 4, characterize the consistency of the model with the data that it has "seen" during the calibration. However, they tell us nothing about its performance on "unseen" data. In machine learning (ML), this problem is solved by randomly splitting the available data into "training" and "testing" sets Q_{train} and Q_{test}. One then calibrates the model over Q_{train} and evaluates its performance on Q_{test}. Usually, one allocates 70% of data to Q_{train}; in our experiments, this proportion also seems to work well.

This approach addresses the so-called "generalizability" of the model. A well-trained model must demonstrate similar metrics on Q_{train} and Q_{test}. If, e.g., RPEs on Q_{test} significantly exceed the residual values in training, this may indicate "overfitting": an excessively flexible model that learns random fluctuations rather than physical features inherent in the data. Such a model cannot be considered reliable, nor its parameters reproducible. As a remedy, one could use more data or switch to a "stiffer" model.

Furthermore, most ML models have some "hyperparameters" such as discretization granularity, numbers of layers in a neural network, their types, etc. Generally, these affect the flexibility of the model as well as the number of trainable parameters. Sometimes, one may choose hyperparameters a priori based on physical considerations. Often, however, they may only be found from data along with the model's parameters. In this case, one uses the third separate "validation" dataset Q_{valid} [32]. Again, the basic rule here is that we should make the final evaluation on data that the model has not "seen" during its training nor during the fine-tuning of hyperparameters.

We can easily adapt this procedure to camera calibration. Given a dataset Q, we first split it into Q_{train}, Q_{valid}, and Q_{test}. Let us assume that we wish to calibrate either our "model A" or "model B". This choice is our hyperparameter: model B can fit more complex imaging geometries but is more prone to overfitting. We calibrate both models according to Equation (6) using Q_{train} and determine respective intrinsic parameters $\vec{\theta}^{\,(fin\ A)}$ and $\vec{\theta}^{\,(fin\ B)}$. After that, we evaluate $D_{\text{DF RMS RPE}}(Q_{\text{valid}} \mid \ldots)$ for both models. Depending on the outcomes, we pick the model that better fits the data and demonstrates neither over- nor underfitting. Finally, once the model is fixed, we find $E^{(fin)} = D_{\text{DS RMS RPE}}(Q_{\text{test}} \mid \ldots)$ and report it as a measure of the final calibration quality.

In what follows, we in fact assume a simpler procedure (we call it *final calibration*) which uses only Q_{train} and Q_{test}; we do not optimize hyperparameters and do not choose the model based on data. A similar strategy has been used in [8].

The recipe above warrants a few remarks. First, all records in a frame depend on the same camera pose. We thus can only split datasets at the level of frames. Second, in order to compute RPEs/FPEs on a new data frame for some fixed intrinsic parameters $\vec{\theta}$, we need to first find a best-fit camera pose via the so-called *bundle adjustment*:

$$(\vec{t}^{\,*}, \vec{u}^{\,*}) = \operatorname{argmin}_{(\vec{t},\vec{u})} D^2_{\text{DF RMS RPE}}\Big(A \mid \vec{\theta}, \vec{t}, \vec{u}\Big). \tag{11}$$

In OpenCV, this optimization is implemented in the `solvePnP()` function [30]. Note that formally the "final calibration" uses a smaller dataset and may thus produce a model inferior to the "initial calibration". This is the price of the added quality guarantees; we believe that the benefits are almost always worth it.

7.3. K-Fold Cross-Validation

The "final calibration" above provides us the final set of intrinsic parameters $\vec{\mu}_\theta = \vec{\theta}^{\,(fin)}$. In order to estimate their variability, we employ yet another ML-inspired empirical technique—the so-called *K-fold cross-validation* [32].

In essence, we repeat the same steps as in the "final calibration" K times (we use $K = 10$), each time making a new random splitting of data into Q_{train} and Q_{test}. The collection of the resulting intrinsic parameters $\vec{\theta}^{\,(k)}$ and the residual RMS RPE values $E^{(k)}_{\text{train}}$ and $E^{(k)}_{\text{test}}$ for $k = 1, \ldots, K$ is retained; the remaining calibrated parameters and evaluation results are discarded. From that, we derive the following two indicators:

$$\begin{aligned} \Sigma_\theta^{(KF\ full)} &= \operatorname{cov}\left(\left\{\vec{\theta}^{\,(k)}\right\}_{k=1}^{K}\right), \\ \delta^2 E^{(KF)} &= \operatorname{var}\left(\left\{E^{(k)}_{\text{train}}\right\}_{k=1}^{K}\right) + \operatorname{var}\left(\left\{E^{(k)}_{\text{test}}\right\}_{k=1}^{K}\right). \end{aligned} \tag{12}$$

The value $\delta E^{(KF)}$ estimates the stability of the residual RPEs and quantifies any claims of "significantly higher/lower" RPE levels when, e.g., detecting overfitting (Section 6.1). The matrix $\Sigma_\theta^{(KF\ full)}$ can in principle be used as an estimate for Σ_θ of Sections 6.2 and 6.3. However, in practice, K is usually significantly smaller than the number of independent components in $\Sigma_\theta^{(KF\ full)}$. The latter then ends up being rank-deficient, and even if it does have a full rank, its inverses are often unstable due to insufficient statistics. As a pragmatic fix to this problem, we define a "robustified" matrix $\Sigma_\theta^{(KF)}$ that has the same diagonal as $\Sigma_\theta^{(KF\ full)}$ and zero non-diagonal elements. As discussed above, such construction does not introduce new biases and improves stability. With $\Sigma_\theta = \Sigma_\theta^{(KF)}$, we then may complete the characterization of the model's quality.

Note that our recipe is different from the typical descriptions of K-fold cross-validation in the literature. In ML, one typically assumes relatively large datasets that may be arbitrarily sub-divided into parts; by contrast, in camera calibration, we usually deal with at most a few dozen camera poses, hence our pragmatical modification.

Figure 8 shows EFPEs obtained with $\vec{\theta} = \vec{\theta}^{(fin)}$ and $\Sigma_\theta = \Sigma_\theta^{(KF)}$ produced as discussed above. The typical values in Figure 8d are more consistent with Figure 5d than those in Figure 6; we thus believe that $\Sigma_\theta^{(KF)}$ in this case better characterizes the model than the internal estimation in the `calibrateCamera()` function.

One may argue whether EFPEG plots such as Figure 8a should use the units of mm/m or radians. Indeed, scalar EFPEs for central models directly correspond to angular uncertainties of the view rays emitted from the projection center. One reason to prefer our notation is that Equation (8) can easily accommodate anisotropic errors (separate estimates for Δx^W and Δy^W); respective angular quantities may be tricky to define. An even more complex picture arises for non-central camera models. In this case, the uncertainty profile for a view ray in 3D is described by a "Gaussian beam": a complicated object that induces a 2D Gaussian PDF over any intersecting plane. Depending on the calibration camera poses and the data quality, the "waist" of such beam will not necessarily be located at the camera's origin. A practical way to characterize the expected model errors in this case could include a series of plots such as Figure 8a but evaluated at different distances from the camera (selected as dictated by applications). (We thank the members of the audience at the 2022 Annual Meeting of the German Society for Applied Optics (DGaO) who have drawn our attention to this issue.)

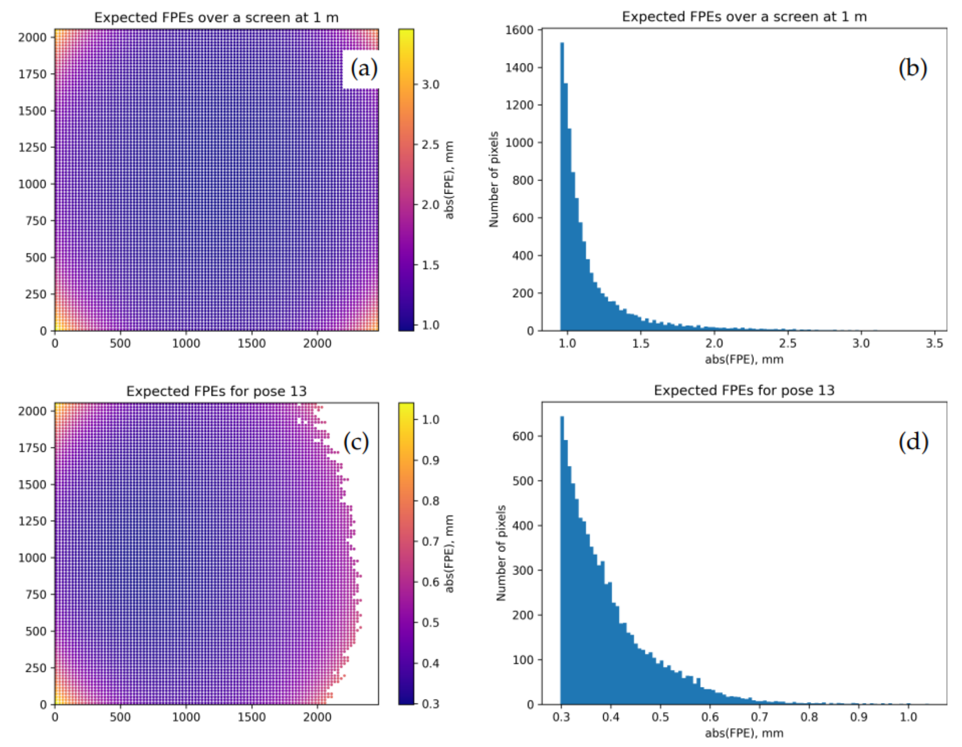

Figure 8. Uncertainty in view ray positions induced by the residual uncertainty in the intrinsic camera parameters as estimated by K-fold cross-validation. Model A is calibrated over the training subset Q_{train} of the dataset 0; x- and y-axes in (**a**,**c**) are camera pixel coordinates. (**a**,**b**): map and histogram of EFPEGs (expected FPEs over the plane 1 m away from the camera). (**c**,**d**): map and histogram of EFPEs for the same pose as in Figure 5. As could be expected, EFPEs in (**a**,**c**) are bounded from below by a positive value that is achieved near the center of the frame.

8. Proposed Workflow and Reporting of Outcomes

Summarizing the discussion in the previous section, here, we present the list of essential steps needed to ensure a controllable calibration session. The method receives a dataset Q with N data frames as an input and produces intrinsic parameters $\vec{\theta}^*$ accompanied by sufficient quality specifications.

1. **Initial calibration.** Calibrate the model with all N data frames. Output N per-pose residual RPE values $E_i^{(ini)}$ (Section 7.1).
2. **Outlier rejection.** Based on values $E_i^{(ini)}$, remove "bad" data frames (Section 7.1).
3. **Final calibration.** Randomly sub-divide Q into Q_{train} and Q_{test}. Calibrate the model over Q_{train}. Output the resulting intrinsic parameters $\vec{\theta}^{(fin)}$ and the residual RPEs/FPEs as per-pose/per-dataset values and point-wise maps (Section 7.2).
4. **K-fold cross-validation.** Repeat K times the splitting of Q into training and test sets; calibrate sub-models. Output stability indicators $\delta E_{(KF)}$ and $\Sigma_\theta^{(KF)}$ (Section 7.3).
5. **Generalizability check.** Perform bundle adjustment for $\vec{\theta}^{(fin)}$, evaluate RPEs and FPEs over Q_{test}. Output respective per-pose, per-dataset values, and point-wise maps.
6. **Reliability metrics.** Map expected FPEs based on $\vec{\theta}^{(fin)}$ and $\Sigma_\theta^{(KF)}$. Output the map of EFPEGs and per-pose maps of EFPEs for the poses in Q (Section 6.3).

The outcomes of this procedure include the intrinsic parameters $\vec{\theta}^* \equiv \vec{\theta}^{(fin)}$, the covariance matrix $\Sigma_\theta^* \equiv \Sigma_\theta^{(KF)}$, and the map of EFPEGs over the sensor—as discussed above, these objects may be useful for the downstream applications of the calibrated model. In addition, one may report the residual RPEs and FPEs over Q_{train} and Q_{test} and the estimated variability $\delta E^{(KF)}$ of the residual RPEs. The user then may decide whether the model is sufficiently flexible and the size of the calibration dataset adequate.

9. Experimental Illustration

We conducted our experiments and collected five datasets as described in Appendix B. For each dataset, we calibrated three models: "model A", "model B", and the additional "model AB". The latter was introduced in order to overcome instabilities of the "model B". It uses the same parameterization, but the program first calibrates the "model A" and then uses it as the starting point for the subsequent calibration of the full "model B". Table 1 summarizes our results in terms of the following metrics:

Table 1. Outcomes of our experiments. The meaning of entries is described in the text.

Dataset/Model	**0/A**	**0/B**	**0/AB**	**1/A**	**1/B**	**1/AB**	**2/A**	**2/B**	**2/AB**	**3/A**	**3/B**	**3/AB**	**4/A**	**4/B**	**4/AB**
N_{poses}		29			18			17			17			19	
$E^{(ini)}$, px	2.69	2.69	2.69	2.30	2.27	2.30	2.10	2.09	2.09	0.47	0.40	0.47	0.51	0.40	0.51
N_{outliers}	2	2	2	1	0	1	1	1	1	1	1	2	2	4	2
$E_{\text{train}}^{(fin)}$, px	2.07	2.06	2.08	2.24	1.94	2.21	1.86	1.90	2.02	0.43	0.38	0.38	0.42	0.23	0.39
$E_{\text{test}}^{(fin)}$, px	2.29	2.35	2.28	1.81	2.96	1.43	2.09	1.93	1.35	0.43	0.33	0.41	0.33	0.25	0.41
$\delta E^{(KF)}$, px	0.31	0.42	0.31	0.70	0.42	0.65	0.68	0.45	0.66	0.10	0.11	0.08	0.07	0.05	0.08
$D_{\text{RMS EFPEG}}$, mm/m	1.37	51.3	1.70	3.30	37.2	3.87	3.50	3295.4	4.48	1.55	19.4	0.71	1.01	28.3	1.04
Sample pose		13			3			5			10			17	
$D_{\text{DF RMS RPE}}^{(fin)}$, px	2.41	2.36	2.40	1.26	1.30	1.27	0.46	0.49	0.45	0.23	0.25	0.22	0.38	0.26	0.38
$D_{\text{DF RMS FPE}}^{(fin)}$, mm	0.37	0.37	0.37	0.44	0.46	0.44	0.42	0.44	0.40	0.16	0.18	0.16	0.08	0.06	0.08
$D_{\text{DF RMS EFPE}}^{(KF)}$, mm	0.43	11.9	0.52	2.69	29.4	3.19	5.83	334.4	7.60	2.35	9.60	0.88	0.46	7.68	0.48

- N_{poses} is the number of data frames in the dataset before the rejection of outliers;
- $E^{(ini)}$ is the residual RMS RPE for the entire dataset after the "initial calibration";
- N_{outliers} is the number of detected (and rejected) "bad" data frames;
- $E_{\text{train}}^{(fin)}$ is the residual RMS RPE after the "final calibration" evaluated over Q_{train};
- $E_{\text{test}}^{(fin)}$ is the residual RMS RPE after the "final calibration" evaluated over Q_{test};
- $\delta E^{(KF)}$ is the empirical variability scale of the residual per-dataset RMS RPEs;
- $D_{\text{RMS EFPEG}}^{(KF)}$ is the RMS "expected FPE gain" (EFPEG) estimated from $\Sigma_{\theta}^{(KF)}$;
- "Sample pose" is used as an example to illustrate the subsequent per-pose values;
- $D_{\text{DF RMS RPE}}^{(fin)}$ is a sample per-pose RMS RPE upon the "final calibration";
- $D_{\text{DF RMS FPE}}^{(fin)}$ is a sample per-pose RMS FPE upon the "final calibration";
- $D_{\text{DF RMS EFPE}}^{(KF)}$ is a sample per-pose RMS EFPE based on K-fold cross-validation.

Figure 9 illustrates the calibration outcomes of the more nuanced "model AB" over the dataset 3. The EFPEG map and the EFPEs for pose 10 promise here sub-mm uncertainties.

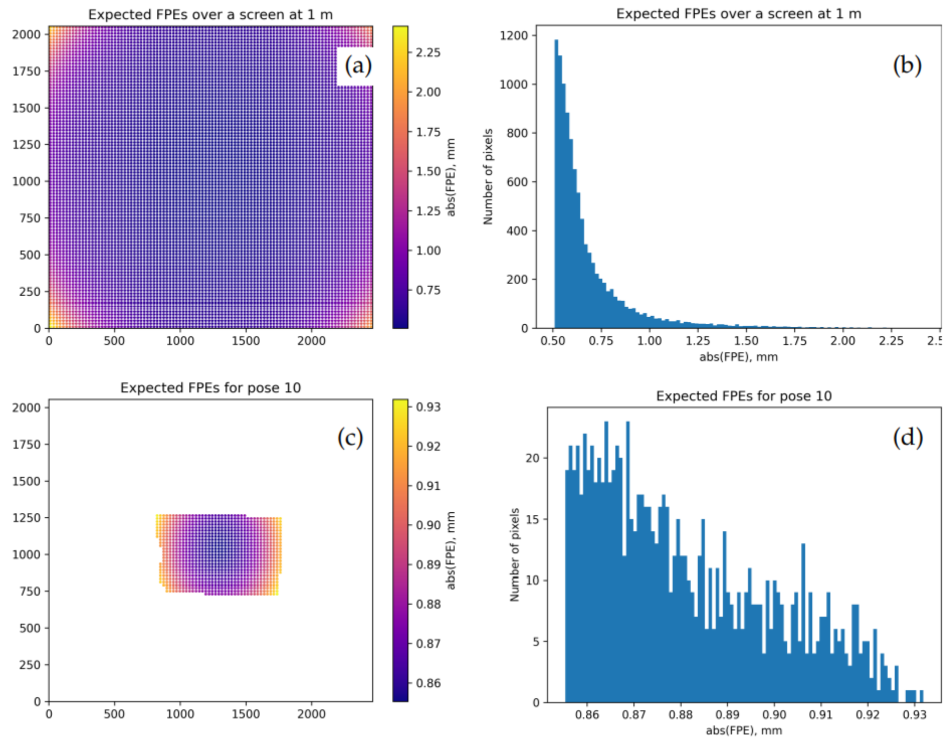

Figure 9. (**a**,**b**): map and histogram of the final EFPEGs for the model AB calibrated over the dataset 3. (**c**,**d**): map and histogram of the expected FPEs (EFPEs) for the pose 10. As in Figure 8, the x- and y-axes in (**a**,**c**) correspond to camera pixel coordinates.

Discussion of Experimental Results

By inspecting Table 1, we may derive a number of non-trivial conclusions:

1. Data collected with an 8K monitor (datasets 3, 4) enable a significantly better calibration that those obtained with an HD screen (datasets 0, 1, 2) as can be seen, e.g., in the residual error levels $E^{(fin)}_{\text{test}}$. Apart from larger screen pixels, this may be due to the aforementioned Moiré effect (more pronounced with an HD screen).
2. The commonly used aggregate error indicators are very weakly sensitive to the detailed calibration quality aspects defined in Section 1: models with wildly different reliability and reproducibility metrics demonstrate very similar values of $E^{(ini)}$.
3. The full OpenCV camera model ("model B") is much less stable than the reduced "model A", as follows from the values $D_{\text{RMS EFPEG}}$. The two-step optimization ("model AB") significantly improves the situation; one possible explanation is that the "model B" too easily becomes trapped in some shallow local minima, and the informed initialization helps it select a better local minimum. Therefore, in practice, one should always prefer the "model AB" to the "model B".
4. Our larger dataset 0 has relatively high decoding errors, and the performance metrics of the "model A" and the "model AB" are almost identical. Therefore, in this case, one should prefer the simpler "model A". As a benefit, the latter promises slightly lower EFPEGs ($D_{\text{RMS EFPEG}}$) due to its higher "stiffness". (For a better justification one should use the "validation" logic as discussed in Section 7.2.)
5. The datasets 1 and 2 are apparently too small to constrain our models, as follows from the increased $\delta E^{(KF)}$ and $D_{\text{RMS EFPEG}}$ values compared with the larger datasets. This may be also a reason for the test metrics $E^{(fin)}_{\text{test}}$ to be lower that the training accuracy $E^{(fin)}_{\text{train}}$ (although the difference is still of order $\delta E^{(KF)}$). The higher flexibility of the "model AB" then translates to higher expected errors $D_{\text{RMS EFPEG}}$.

6. With the better quality datasets 3 and 4, the "model AB" finally demonstrates an advantage and promises slightly better metric uncertainties $D_{\text{RMS EFPEG}}$. With the dataset 3, we see sub-mm RMS uncertainty at the distance of 1 m; in the center of the frame, the errors are even lower (Figure 9). Note that the residual FPEs $D^{(fin)}_{\text{DF RMS FPE}}$ for the dataset 4/pose 17 nicely agree with the estimated decoding errors in Figure 3.

10. General Discussion

Our results suggest that it is possible to relatively easily calibrate typical industrial cameras to the accuracy of order 1 mm when camera-assisted measurements happen at the distances of order 1 m and at the same time produce some "certificates" of calibration quality. The main enabling factor here is the superior quality of calibration data delivered by ATTs. We further observe that modern higher-resolution screens (4K and 8K) offer a significant advantage due to smaller pixels and suppressed Moiré effects. As such screens become widely available, we see no reason to calibrate cameras with static patterns, even for less demanding applications in computer vision.

We also observe that the decoding errors for the affordable 1920×1080 screens are relatively high and do not justify the use of advanced models such as our "model AB". A simple pinhole model with low-order distortions is sufficient here to obtain quite consistent, reliable, and reproducible results. Note, however, that this effect may be also due to the relatively high quality of lenses used in our experiments: simpler or wider-angle optics may cause higher distortions and necessitate a more flexible model.

The ML-inspired procedure outlined in Section 7 appears to consistently deliver adequate variability estimates for the parameters and the performance metrics of camera models. By contrast, the internal estimates built into the OpenCV functions may be significantly off, sometimes by a factor of 10 or more. (For example, typical error scales in Figure 6 and in Figure 8c,d differ by about a factor of 6; we have witnessed significantly more extreme examples in our complete experimental database.) The numerical behavior of the complete OpenCV model also appears to be unstable, and one may need to apply some ad hoc fixes such as our "model AB" in order to obtain useful calibration results.

Further Improvements and Future Work

If order to further push the boundaries in terms of data quality and the resulting model uncertainties, one has to account for various minor effects that are present in any setup similar to ours. One such issue is the refraction in the screen's cover glass that introduces systematic deviations into decoded coordinates x^S and y^S of order 0.1 mm when the view angles are sufficiently far away from the normal incidence [18]. Respective optical properties of the screen as well as its deformations due to gravity or thermal expansion [17] then must be modeled and calibrated separately.

Furthermore, one may switch to high-dynamic-range screens and separately calibrate the pixel response functions (gamma curves). Respective corrections may improve the decoding accuracy or, alternatively, reduce the necessary data acquisition times.

As shown above, Moiré structures may significantly impact data quality. To the best of our knowledge, these effects have not been studied systematically. In practice, one usually adjusts calibration poses and screen parameters until such structures become less visible. For ultimate-quality measurements, one may require more rigorous procedures.

Another less-studied effect is blurred (non-sharp) projection. There is a general assumption that blurring does not affect coordinates decoded with phase-shifted cosine patterns [33]. However, this statement is true only for Gaussian blurring kernels whose scale is significantly smaller than the spatial modulation wavelengths of patterns. The decoding is also noticeably biased near screen boundaries if they appear in the frame.

Note that already our achieved uncertainty levels may approach the breaking point for the assumption of the ideal central projection. Depending on the quality of optics, at some point one then has to switch to non-central models. There is no simple and clear rule to determine this point; also, the calibration of such models is significantly more challenging in practice, and their theory still has some open questions [29].

At the system level, it may be interesting to extend the above approach and address the online calibration problem. In particular, once a camera is studied in a laboratory, one could use simpler procedures and less "expensive" indicators (that ideally do not interfere with its main task) in order to detect and quantify parametric drifts due to environmental factors along with variations in the quality metrics (such as EFPEGs).

All comparisons and quality evaluations in this paper are based on high-quality datasets collected with ATTs. Each such dataset may contain millions of detected 3D points, and therefore provides a very stringent test for a calibrated camera model. Nevertheless, it may be quite instructive to demonstrate the advantages of the proposed workflow in the context of demanding practical applications such as fringe projection-based or deflectometric measurements. We leave such illustrations to future work.

11. Conclusions

In this paper, we discuss the quality of camera calibration and provide practical recommendations for obtaining consistent, reproducible, and reliable outcomes. We introduce several quality indicators that, if accepted in common practice, may potentially lead to the better characterization of calibration outcomes in multiple applications and use-cases both in camera-assisted optical metrology and computer vision fields.

Our approach is empirical in nature and is inspired by the techniques used in Machine Learning. As an illustration of the proposed method, we conduct a series of experiments with typical industrial cameras and commercially available displays. Our procedure to characterize the quality of outputs is shown to be superior to the previously known methods based on fragile assumptions and approximations. The metric uncertainty of the calibrated models in our experiments corresponds to forward projection errors of order 0.1–1.0 mm at the distances of order 1 m and is consistent with the observed levels of residual projection errors.

We hope that the workflows and tools demonstrated in this paper may appear attractive to other practitioners of camera-based measurements. In order to lower the threshold for the adoption of these methods, we published the datasets and the Python code that were used to obtain our results; respective links are provided below.

Author Contributions: Conceptualization, all authors; methodology, A.P. and S.R.; software, A.P. and S.R.; experiments, all authors; resources, J.B.; data curation, A.P.; writing—original draft preparation, A.P. and S.R.; writing—review and editing, all authors. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The Python code and the original datasets for this paper are available upon request from the authors.

Acknowledgments: Most of this work was performed during S.R.'s research sabbatical in the winter 2020/21 at or with Fraunhofer IOSB in Karlsruhe. The authors would like therefore to acknowledge Fraunhofer IOSB in Karlsruhe and Pforzheim University for supporting this work. We are also thankful to the anonymous reviewers whose comments and suggestions have contributed to a better presentation of this work.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Camera Model Implemented in the OpenCV Library

The camera embedding into the 3D world is given by Equation (1) according to the extrinsic parameters $\vec{t}$, $\vec{u}$. The parameterization of rotation matrices $R(\vec{u})$ in OpenCV is as follows:

$$R(\vec{u}) = I\ \cos\theta + (1-\cos\theta)\ \vec{n}\ \vec{n}^T + N(\vec{u})\sin\theta,\ \text{where } I \text{ is a } 3\times 3 \text{ unit matrix,} \tag{A1}$$
$$N(\vec{u}) = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix},\ \theta = |\vec{u}|,\ \text{and } \vec{n} = (n_x, n_y, n_z)^T = \frac{\vec{u}}{\theta}.$$

The intrinsic camera model adopted in OpenCV is formulated in terms of the direct projection relation for a 3D point onto the sensor plane. Given the coordinates $\vec{p}^{\,C} = \left(x^C, y^C, z^C\right)^T$ of some 3D point P in the camera's intrinsic frame, the respective coordinates $\vec{\pi} = \left(x^I, y^I\right)^T = \vec{\Pi}^{(\mathrm{CV})}(\vec{p}^{\,C} \mid \vec{\theta})$ of its projection on the sensor are

$$\vec{\Pi}^{(\mathrm{CV})}(\vec{p}^{\,C} \mid \vec{\theta}) = \left(f_x x''' + c_x,\ f_y y''' + c_y\right)^T,\ \text{where} \tag{A2}$$
$$x''' = \frac{X'''}{Z'''},\ y''' = \frac{Y'''}{X'''},\ \begin{pmatrix} X''' \\ Y''' \\ Z''' \end{pmatrix} = T_1 T_2 \begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix},$$
$$T_1 = \begin{pmatrix} \cos\tau_y\cos\tau_x & 0 & \sin\tau_y\cos\tau_x \\ 0 & \cos\tau_y\cos\tau_x & -\sin\tau_x \\ 0 & 0 & 1 \end{pmatrix},$$
$$T_2 = \begin{pmatrix} \cos\tau_y & \sin\tau_y\sin\tau_x & -\sin\tau_y\cos\tau_x \\ 0 & \cos\tau_x & \sin\tau_x \\ \sin\tau_y & -\cos\tau_y\sin\tau_x & \cos\tau_y\cos\tau_x \end{pmatrix},$$
$$x'' = fx' + 2p_1x'y' + p_2(r^2 + 2x'^2) + s_1r^2 + s_2r^4,$$
$$y'' = fy' + p_1(r^2 + 2y'^2) + 2p_2x'y' + s_3r^2 + s_4r^4,$$
$$f = \frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6},\ r^2 = x'^2 + y'^2,\ x' = \frac{x^C}{z^C},\ \text{and } y' = \frac{y^C}{z^C}.$$

The 18 intrinsic parameters $\vec{\theta} = (f_x, f_y, c_x, c_y, k_1, k_2, k_3, p_1, p_2, k_4, k_5, k_6, s_1, s_2, s_3, s_4, \tau_x, \tau_y)$ can be interpreted as follows. The values f_x, f_y, c_x, c_y define the sensor's size, resolution, and placement in the image plane according to the basic pinhole camera model (skewed projection is not allowed). The remaining terms describe various types of distortions: k_1, k_2, k_3, k_4, k_5, k_6—radial, p_1, p_2—tangential, and s_1, s_2, s_3, s_4—thin-prism. Finally, τ_x and τ_y correspond to the possible tilting of the image plane with respect to the lens (related via Scheimpflug principle to the tilt of the focus plane). When the parameters $k_1, \ldots, \tau_y$ vanish, Equation (A2) reduces to the undistorted pinhole model.

In order to derive an inverse projection relation for the model Equation (A2), one would need to invert these formulas—a task that is intractable in closed form. Instead, in practice, we find the direction $\vec{r} = \vec{R}^{(\mathrm{CV})}(\vec{\pi} \mid \vec{\theta})$ of a view ray corresponding to a pixel $\vec{\pi}$ numerically as a solution of a small optimization problem:

$$\vec{R}^{(\mathrm{CV})}(\vec{\pi} \mid \vec{\theta}) = (r_x^*, r_y^*, 1)^T,\ \text{where} \tag{A3}$$
$$(r_x^*, r_y^*) = \operatorname{argmin}_{(r_x, r_y)} \left\| \vec{\pi} - \vec{\Pi}^{(\mathrm{CV})}\left((r_x, r_y, 1)^T \mid \vec{\theta}\right) \right\|^2.$$

In non-pathological cases, Equation (A3) can be solved by some Newton's scheme or another iterative method. In our code used in the experiments, we employ the Levenberg–Marquardt method [31] implemented in the SciPy library [34] and observe solid convergence.

Finally, the analysis of the calibrated model requires the derivatives $\partial\vec{\Pi}/\partial\vec{p}^{\,C}$, $\partial\vec{\Pi}/\partial\vec{\theta}$, $\partial\vec{R}/\partial\vec{\pi}$, and $\partial\vec{R}/\partial\vec{\theta}$ (cf. Section 6.3 and Equation (8)). The differentiation of Equation (A2) and the conversion of formulas to code is trivial but tedious. Instead, we apply an automated differentiation tool implemented in the AutoGrad package [35] to the code for $\vec{\Pi}^{(\mathrm{CV})}(\cdot|\cdot)$. The derivatives of $\vec{R}^{(\mathrm{CV})}(\cdot|\cdot)$ are then obtained by trivial matrix manipulations.

Appendix B. Hardware Setup and Collected Datasets

Our experimental setup uses two identical industrial cameras (**Camera 1** and **Camera 2**) by Allied Vision, model Mako G-507 B. Their 2/3 sensor (11.1 mm in diagonal) has 2464 × 2056 monochromatic pixels (horizontal × vertical) working at 12-bit resolution and arranged at the pitch of 3.45 µm × 3.45 µm. Each camera is coupled with a Fujinon HF8XA-5M lens chosen to provide adequate optical resolution. In all experiments, the focus distance was fixed at about 900 mm and the F-number was fixed at 4.0 when collecting all datasets, except for the dataset 3, where it was set to 2.0. The lens has a focal length of 8.0 mm, the field of view with a 2/3 sensor is about 56° × 48°.

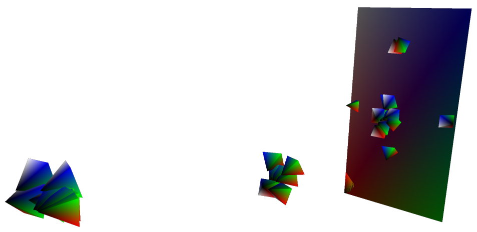
Figure A1. Visualization of camera poses for the dataset 0. The rectangle indicates the coding screen and each pyramid denotes a calibration camera pose. The projection center corresponds to the tip of each pyramid.

As active targets, we have used two different computer displays. We assume them to be sufficiently flat and make no modification to ensure planarity other than placing them vertically without tilting. The first one (**Display 1**) is a 55″ Philips monitor (model 55BDL5057P) with 1920 × 1080 pixels and the active region of 1209.6 mm × 680.4 mm (the pixel pitch is 0.630 mm × 0.630 mm). The second (**Display 2**) is a 32″ 8K Dell monitor (UP3218K) with 7680 × 4320 pixels in the active region of 697.0 mm × 392.0 mm (pitch 0.091 mm × 0.091 mm).

Each recorded calibration dataset includes data frames captured at different camera poses. In order to efficiently exploit the screen and the sensor resolution, and avoid Moiré effects as well as excessive blurring, the camera positions are chosen at about 100 mm, 250 mm, 1000 mm, and 2500 mm from the 55″ Philips monitor and at about 300 mm, 800 mm, and 1700 mm from the 8K Dell monitor. At each distance, we rotated the camera so as to cover most of the frame area with overlapping observations. Figure 2 shows the experimental setup where the camera is placed at a distance of about 1000 mm from the Philips monitor and Figure A1 visualizes all camera positions for the dataset 0.

The spatial frequencies and the numbers of phase shifts of the coding patterns were selected to enable robust decoding both at the shortest and at the furthest camera poses; the number of displayed patterns/camera images at each pose was 16 for the datasets collected with the Phillips monitor and 45 with the Dell monitor.

The collected datasets have the following characteristics:

Dataset	Camera	F-Number	Display	Number of Poses	Uncertainties
0	1	4	1	29	no
1	2	4	1	18	no
2	1	4	1	17	no
3	2	2	2	17	yes
4	2	4	2	19	yes

Appendix C. Mahalanobis Distance and the Plausibility of Outcomes

Given an n-dimensional Gaussian PDF $P = \mathcal{N}(\vec{\mu}; \Sigma)$ with a central value $\vec{\mu}$ and covariance matrix Σ, how can we evaluate the consistency of an arbitrary vector $\vec{\theta}$ with P? In other words, how plausible is the assumption that $\vec{\theta}$ has been sampled from P?

In the one-dimensional case ($n = 1$), we could find the difference between $\vec{\theta}$ and $\vec{\mu}$ in the units of standard deviation ("sigma") and apply the well-known "empirical rule": the probability of encountering a deviation of one "sigma" or less is 68.2%, a deviation of two "sigmas" or less—95.4%, etc. The generalization of this rule to higher dimensions $n > 1$ uses the so-called *Mahalanobis distance*:

$$D_M(\vec{\theta} \mid \vec{\mu}, \Sigma) = \sqrt{(\vec{\theta} - \vec{\mu})^T \Sigma^{-1} (\vec{\theta} - \vec{\mu})}. \tag{A4}$$

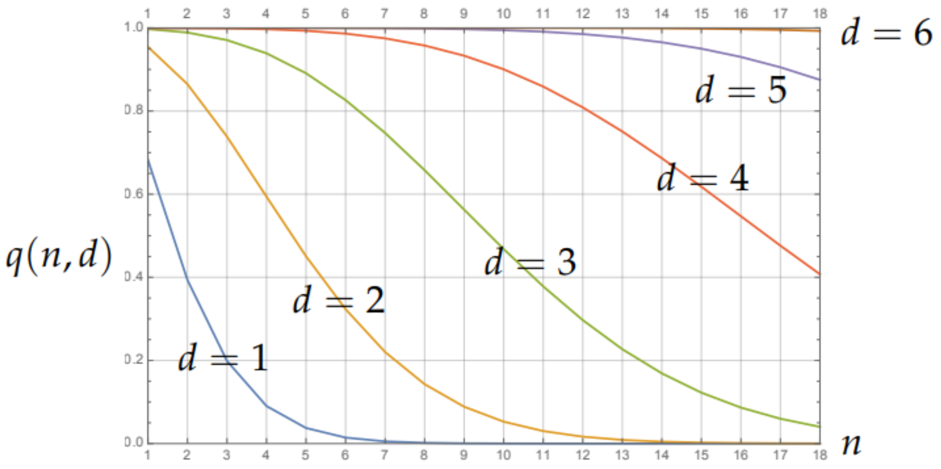

Figure A2. Profiles of the accumulated Gaussian probability mass $q(n,d)$ of Equation (A5) for several values of space dimensionality n and Mahalanobis distance d.

In order to interpret the values of D_M, one needs a generalized "empirical rule" that represents the probability for a random sample drawn from $\mathcal{N}(\vec{\mu};\Sigma)$ to land within a given Mahalanobis distance d from $\vec{\mu}$. This probability can be evaluated in closed form:

$$\text{Prob}\Big[D_M(\vec{\theta} \mid \vec{\mu}, \Sigma) < d \text{ for } \vec{\theta} \sim \mathcal{N}(\vec{\mu};\Sigma)\Big] = q(n,d) = 1 - \frac{\Gamma(n/2, d^2/2)}{\Gamma(n/2)}, \tag{A5}$$

where $\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$ and $\Gamma(a,z) = \int_z^\infty x^{a-1} e^{-x} dx$ are the regular and the incomplete gamma-functions (cf. χ^2-distribution and its cumulative distribution function).

Some values of the function $q(n,d)$ are shown in Figure A2. In particular, the $n = 1$ case reproduces the above-mentioned one-dimensional "empirical rule". In higher dimensions, the bulk of probability mass moves further away from the center. For instance, in $n = 9$ dimensions, only 0.056% of the accumulated probability remains inside the "one-sigma" ($d = 1$) hyperellipsoid, and there is only a 56.27% chance for a random sample from a Gaussian distribution to land within three "sigmas" away from its central point. These definitions allow us to define a convenient "plausibility" metric:

$$\text{P}_M(\vec{\theta} \mid \vec{\mu}, \Sigma) = q(n, D_M(\vec{\theta} \mid \vec{\mu}, \Sigma)). \tag{A6}$$

Using Equation (A6) and $\vec{\theta}^*$, Σ_θ—the outputs from the workflow of Section 8—one can, e.g., compare the outcomes of several calibrations and decide if they are compatible.

References

1. Betzig, E.; Patterson, G.H.; Sougrat, R.; Lindwasser, O.W.; Olenych, S.; Bonifacino, J.S.; Davidson, M.W.; Lippincott-Schwartz, J.; Hess, H.F. Imaging Intracellular Fluorescent Proteins at Nanometer Resolution. *Science* **2006**, *313*, 1642–1645. [CrossRef] [PubMed]
2. EMVA. *Standard for Characterization and Presentation of Specification Data for Image Sensors and Cameras*; EMVA—European Machine Vision Association: Barcelona, Spain, 2021.

3. Zhang, Z. A Flexible New Technique for Camera Calibration. *Pattern Anal. Mach. Intell. IEEE Trans.* **2000**, *22*, 1330–1334. [CrossRef]
4. Bradski, G. The OpenCV Library. 2000. Available online: https://www.drdobbs.com/open-source/the-opencv-library/184404319 (accessed on 1 August 2022).
5. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*, 2nd ed.; Cambrige University Press: Cambrige, UK, 2004.
6. Hanning, T. *High Precision Camera Calibration: Habilitation Thesis*; Springer Fachmedien, Vieweg+Teubner Verlag: Wiesbaden, Germany, 2011.
7. Schilling, H.; Diebold, M.; Gutsche, M.; Jähne, B. On the design of a fractal calibration pattern for improved camera calibration. *Tm-Tech. Mess.* **2017**, *84*, 440–451. [CrossRef]
8. Schöps, T.; Larsson, V.; Pollefeys, M.; Sattler, T. Why Having 10,000 Parameters in Your Camera Model is Better Than Twelve. Proc. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 2535–2544.
9. Hannemose, M.; Wilm, J.; Frisvad, J.R. Superaccurate camera calibration via inverse rendering. In Proceedings of the SPIE 11057, Modeling Aspects in Optical Metrology VII, Munich, Germany, 21 June 2019; p. 1105717. [CrossRef]
10. Huang, L.; Zhang, Q.; Asundi, A. Camera calibration with active phase target: Improvement on feature detection and optimization. *Opt. Lett.* **2013**, *38*, 1446–1448. [CrossRef]
11. Schmalz, C.; Forster, F.; Angelopoulou, E. Camera calibration: Active versus passive targets. *Opt. Eng.* **2011**, *50*, 113601.
12. Fischer, M.; Petz, M.; Tutsch, R. Vorhersage des Phasenrauschens in optischen Messsystemen mit strukturierter Beleuchtung. *Tm-Tech. Mess.* **2012**, *79*, 451–458. [CrossRef]
13. Grossberg, M.D.; Nayar, S.K. A general imaging model and a method for finding its parameters. In Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV), Vancouver, BC, Canada, 7–14 July 2001; pp. 108–115.
14. Ramalingam, S.; Sturm, P.F.; Lodha, S.K. Towards Complete Generic Camera Calibration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, 20–25 June 2005; IEEE Computer Society: Washington, DC, USA, 2005; pp. 1093–1098. [CrossRef]
15. Bothe, T.; Gesierich, A.; Li, W.; Schulte, M. Verfahren und Vorrichtung zur Kalibrierung einer optischen Einrichtung. German Patent DE102005061931A1, 28 June 2007.
16. Bothe, T.; Li, W.; Schulte, M.; von Kopylow, C.; Bergmann, R.B.; Jueptner, W.P.O. Vision ray calibration for the quantitative geometric description of general imaging and projection optics in metrology. *Appl. Opt.* **2010**, *49*, 5851–5860. [CrossRef]
17. Reh, T.; Li, W.; Burke, J.; Bergmann, R.B. Improving the generic camera calibration technique by an extended model of calibration display. *J. Europ. Opt. Soc. Rap. Public.* **2014**, *9*, 14044. [CrossRef]
18. Dierke, H.; Petz, M.; Tutsch, R. Photogrammetric determination of the refractive properties of liquid crystal displays. In *Forum Bildverarbeitung 2018*; Längle, T., Puente León, F., Heizmann, M., Eds.; KIT Scientific Publishing: Karlsruhe, Germany, 2018; pp. 13–24.
19. Salvi, J.; Armangue, X.; Batlle, J. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognit.* **2002**, *35*, 1617–1635. [CrossRef]
20. Bogdan, O.; Eckstein, V.; Rameau, F.; Bazin, J.C. DeepCalib: A deep learning approach for automatic intrinsic calibration of wide field-of-view cameras. In Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production (CVMP'18), London, UK, 13–14 December 2018; pp. 1–10. [CrossRef]
21. Brunken, H.; Gühmann, C. Deep learning self-calibration from planes. In Proceedings of the 12th International Conference on Machine Vision (ICMV'19), Amsterdam, The Netherlands, 31 January 2020; Volume 11433, p. 114333L. [CrossRef]
22. Zhang, J.; Luo, B.; Xiang, Z.; Zhang, Q.; Wang, Y.; Su, X.; Liu, J.; Li, L.; Wang, W. Deep-learning-based adaptive camera calibration for various defocusing degrees. *Opt. Lett.* **2021**, *46*, 5537–5540. [CrossRef]
23. Kannala, J.; Brandt, S. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 1335–1340. [CrossRef] [PubMed]
24. Popescu, V.; Dauble, J.; Mei, C.; Sacks, E. An Efficient Error-Bounded General Camera Model. In Proceedings of the 3DPVT'06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06), Chapel Hill, NC, USA, 14–16 June 2006; IEEE Computer Society: Washington, DC, USA, 2006; pp. 121–128. [CrossRef]
25. Dunne, A.K.; Mallon, J.; Whelan, P.F. Efficient generic calibration method for general cameras with single centre of projection. *Comput. Vis. Image Underst.* **2010**, *114*, 220–233. [CrossRef]
26. Sun, Q.; Hou, Y.; Tan, Q. A new method of camera calibration based on the segmentation model. *Optik* **2013**, *124*, 6991–6995. [CrossRef]
27. Miraldo, P.; Araujo, H. Calibration of smooth camera models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 2091–2103. [CrossRef]
28. Xiang, Z.; Dai, X.; Gong, X. Noncentral catadioptric camera calibration using a generalized unified model. *Opt. Lett.* **2013**, *38*, 1367–1369. [CrossRef] [PubMed]
29. Pak, A. The concept of smooth generic camera calibration for optical metrology. *Tm-Tech. Mess.* **2015**, *83*, 25–35. [CrossRef]
30. OpenCV Documentation: Camera Calibration and 3D Reconstruction Section. 2021. Available online: https://docs.opencv.org/4.5.2/d9/d0c/group__calib3d.html (accessed on 1 August 2022).
31. Nocedal, J.; Wright, S.J. *Numerical Optimization*; Springer Series in Operations Research; Springer: New York, NY, USA, 1999.

32. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning—Data Mining, Inference, and Prediction*, 2nd ed.; Springer: New York, NY, USA, 2017.
33. Werling, S.B.; Mai, M.; Heizmann, M.; Beyerer, J. Inspection of specular and partially specular surfaces. *Metrol. Meas. Syst.* **2009**, *16*, 415–431.
34. Jones, E.; Oliphant, T.; Peterson, P. SciPy: Open Source Scientific Tools for Python. 2022. Available online: http://www.scipy.org (accessed on 1 August 2022).
35. Maclaurin, D.; Duvenaud, D.; Johnson, M.; Townsend, J. AutoGrad Package for Python. 2022. Available online: https://github.com/HIPS/autograd (accessed on 1 August 2022).