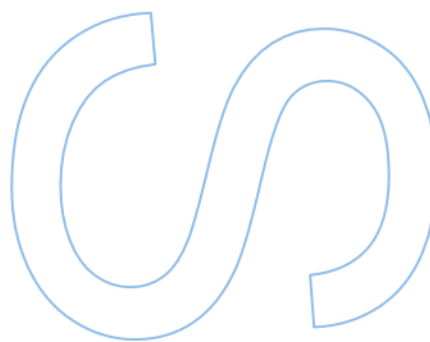
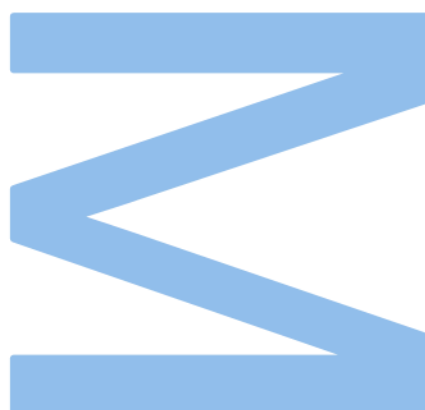


Study of the Development of a Camera's Distortion Parameters Calibration Algorithm



João Pedro da Silva Brito
Master in Engineering Physics
Faculty of Sciences of the University of Porto and
Faculty of Engineering of the University of Porto
2024/2025



Study of the Development of a Camera's Distortion Parameters Calibration Algorithm

João Pedro da Silva Brito

Internship report carried out as part of the Master in Engineering

Physics

Department of Physics and Astronomy

2024/2025

Supervisor

Prof. José Costa Pereira, Assistant professor, Faculty of Engineering
of the University of Porto

External Supervisor in the Reception Entity

Guilherme Franco, AI Developer, wTVision



Acknowledgments

I would like to express my deepest gratitude to my supervisor, Professor José Costa Pereira, for his constant guidance, patience, and invaluable support throughout this journey. His knowledge, encouragement, and thoughtful advice were essential not only for the development of this work but also for my growth during this process. I am truly grateful for the opportunity to learn under his supervision and for the trust and dedication he showed throughout this project.

I would also like to thank Guilherme Franco, External Supervisor at wTVision, for his availability, support, and for sharing his practical knowledge and experience. His guidance and insights into real-world production environments were fundamental to shaping this work and connecting it to practical applications.

A special thank you goes to my friends who accompanied me throughout this journey. The long hours, the challenges, and the moments of uncertainty were made lighter thanks to their presence and encouragement. In particular, I would like to thank Nicolau, Patrícia, and Luís for their friendship, support, and for always being there, both during the difficult moments and the moments worth celebrating.

Finally, I would like to thank my family for their unconditional support, understanding, and encouragement throughout this entire journey. Their belief in me made it possible to reach this milestone.

Abstract

wTVision, a company that specializes in real-time graphics and Augmented Reality (AR) solutions for the broadcast and entertainment industries, provided a corporate setting for this internship. wTVision's technology enables the production of immersive experiences that demand a high degree of visual accuracy and precision.

In AR and Virtual Reality (VR) applications, seamless alignment between virtual features and real-world visuals is necessary to deliver an immersive user experience. The goal is to make the virtual world look geometrically aligned with the real world. However, current calibration methods are extremely time-consuming and require meticulous manual adjustments at different zoom and focus levels. Manual calibration typically takes between 4 to 6 hours and is heavily reliant on human labour, increasing the likelihood of errors.

The main goal of this project is to create a calibration algorithm that optimizes and automates the procedure, reducing the calibration time to about 30 minutes without compromising accuracy. The suggested approach makes better use of cutting-edge methods like image analysis and Machine Learning (ML) to fix geometric distortion issues. This study encompasses a thorough examination of current calibration techniques and the creation of a calibration algorithm capable of determining a camera's distortion coefficients under different scenarios

The results indicate that the proposed system can significantly reduce the time required for calibration while maintaining a high degree of consistency with reference values. The integration of an efficient communication layer between Python scripts and the R^3 software enabled faster image capture and increased the practical feasibility of the method. It is concluded that automation is viable in the studied context, showing high agreement especially in the FoV and good overall alignment of the radial coefficients, which supports its use as an alternative (or support) to the manual workflow in a production environment.

Keywords: Camera calibration, geometric distortion, image analysis, ML, AR, VR, automation, visual accuracy.

Resumo

A wTVision, uma empresa especializada em soluções de gráficos em tempo real e de realidade aumentada para as indústrias de broadcast e entretenimento, proporcionou um ambiente corporativo para este estágio. Com a tecnologia da wTVision, são criadas experiências imersivas que exigem um elevado grau de precisão e rigor visual.

Em aplicações de realidade aumentada e virtual, é necessário um alinhamento perfeito entre os elementos virtuais e os visuais do mundo real para proporcionar uma experiência imersiva ao utilizador. O objetivo é fazer com que o mundo virtual pareça estar geometricamente alinhado com o mundo real. No entanto, os métodos de calibração atuais são extremamente morosos e exigem ajustes manuais meticulosos em diferentes níveis de zoom e focagem. A calibração manual demora, normalmente, entre 4 a 6 horas e depende fortemente de fatores humanos, o que aumenta a probabilidade de erros.

O objetivo deste projeto é criar um algoritmo de calibração que otimize e automatize o processo, reduzindo o tempo total de calibração para cerca de 30 minutos, mantendo a precisão. A abordagem proposta tira melhor partido de métodos avançados, como a análise de imagem e a aprendizagem computacional, para corrigir problemas de distorção geométrica. Este estudo engloba uma análise aprofundada das técnicas de calibração existentes e o desenvolvimento de um algoritmo de calibração capaz de determinar os coeficientes de distorção de uma câmara em diferentes cenários.

Os resultados indicam que o sistema proposto pode reduzir significativamente o tempo necessário para calibração, mantendo um elevado grau de consistência com valores de referência. A integração de uma camada de comunicação eficiente entre scripts Python e o software R^3 permitiu acelerar a captura de imagens e aumentar a viabilidade prática do método. Conclui-se que a automatização é viável no contexto estudado, apresentando elevada concordância sobretudo no FoV e um bom alinhamento global dos coeficientes radiais, o que suporta a sua utilização como alternativa (ou apoio) ao fluxo manual em ambiente de produção.

Palavras-chave: Calibração de câmaras, distorção geométrica, análise de imagem, aprendizagem computacional, realidade aumentada, realidade virtual, automatização, precisão visual.

Contents

Acknowledgements	i
Abstract	ii
Resumo	iii
Contents	vii
List of Tables	viii
List of Figures	xii
Acronyms	xiii
1 Introduction	1
1.1 wTVision	2
1.2 Problem Definition	2
1.3 Document Overview	4
2 State of the Art	6
2.1 Theoretical Background	6
2.1.1 Augmented Reality	6

2.1.2	Lens Distortion	7
2.1.3	R^3 software	9
2.2	Camera Calibration	12
2.2.1	Distortion Parameters	12
2.2.2	TrackMen Lens Calibration Tool	14
2.2.3	Deep Learning-Based Radial Distortion Estimation from a Single Image	15
2.2.3.1	The Brown–Conrady Radial Distortion Model	15
2.2.3.2	Single-image CNN for k_1 and k_2 Estimation	16
2.2.4	R^3 software Calibration	17
2.2.4.1	Detailed Calibration Process	17
2.2.4.2	Nodal Point Calibration	21
2.2.4.3	Challenges of Current Method	22
2.3	Summary	23
3	Camera Parameters Calibration Algorithm	25
3.1	Communication Between Python and R^3	26
3.2	Edge Detection Algorithm using OpenCV	27
3.2.1	Testing with Blender	29
3.2.2	Results and Discussion	31
3.3	Camera Calibration Algorithm	33
3.3.1	Center-shift calibration	34
3.3.2	FoV, k_1 and k_2 Calibration	36

3.3.3	Optimizing Calibration Efficiency	39
3.3.4	Results and Discussion	39
3.4	Deep Learning-Based Estimation of Camera Radial Distortion Parameters	43
3.4.1	Synthetic Dataset Generation	44
3.4.2	Network Architecture and Training Process	45
3.4.3	Results and Discussion	46
4	Conclusions and Future Work	48
A	HTTP Headers	50
B	Calibration Results	51
	Bibliography	58

List of Tables

2.1 Calibration Times for Each Coefficient	22
A.1 HTTP Headers Used in the Request	50

List of Figures

2.1 Example of an AR scene and its corresponding overlay developed by wTVision.	7
2.2 Types of lens distortions	8
2.3 Complete Interface View	10
2.4 Tracking Console Interface	10
2.5 Visual impact of lens distortion on augmented reality rendering.	11
2.6 Parallax effect observed with human vision.	13
2.7 ResNet-18 architecture used for distortion estimation [1]	17
2.8 Center-shift calibration procedure	18
2.9 FoV calibration procedure	19
2.10 Radial distortion calibration for the k_1 coefficient.	20
2.11 Target positions used during calibration	21
3.1 Demonstration of Canny edge detection on a real-world image	28
3.2 Edge detection applied to a checkerboard calibration target.	29
3.3 Edge detection results on Blender-generated test images	32
3.4 Center-shift calibration procedure	35
3.5 Illustration of visual misalignment despite correct center-shift calibration	35
3.6 Calibration of FoV, k_1 , and k_2 using edge detection	36
3.7 Comparison between ground truth and estimated calibration parameters using the proposed algorithm with MockBoard integration	40

3.8 Correlation analysis of estimated calibration parameters	41
3.9 Sample calibration output for Zoom 1 and Focus 1	42
3.10 Comparison of the image before and after application of the Brown–Conrady distortion model	45
3.11 Training loss (MSE) per epoch.	45
3.12 Correlation between predicted and ground truth values of the distortion coefficients k_1 and k_2	46
B.1 Zoom 1 Focus 1 - (a) FoV, (b) K1, (c) K2.	52
B.2 Zoom 1 Focus 2 - (a) FoV, (b) K1, (c) K2.	52
B.3 Zoom 2 Focus 1 - (a) FoV, (b) K1, (c) K2.	52
B.4 Zoom 2 Focus 2 - (a) FoV, (b) K1, (c) K2.	53
B.5 Zoom 3 Focus 1 - (a) FoV, (b) K1, (c) K2.	53
B.6 Zoom 3 Focus 2 - (a) FoV, (b) K1, (c) K2.	54
B.7 Zoom 4 Focus 1 - (a) FoV, (b) K1, (c) K2.	54
B.8 Zoom 4 Focus 2 - (a) FoV, (b) K1, (c) K2.	55
B.9 Zoom 5 Focus 1 - (a) FoV, (b) K1, (c) K2.	55
B.10 Zoom 5 Focus 2 - (a) FoV, (b) K1, (c) K2.	56
B.11 Zoom 6 Focus 1 - (a) FoV, (b) K1, (c) K2.	56
B.12 Zoom 6 Focus 2 - (a) FoV, (b) K1, (c) K2.	57
B.13 Zoom 7 Focus 1 - (a) FoV, (b) K1, (c) K2.	57

Acronyms

AR	Augmented Reality	PTZ	Pan-Tilt-Zoom
VR	Virtual Reality	CCD	Charge-Coupled Device
FoV	Field of View	ML	Machine Learning
CGI	computer-generated imagery	IPC	Inter-Process Communication
CNN	Convolutional Neural Networks	MSE	Mean Square Error
RGB	Red Green Blue		

Chapter 1

Introduction

The broadcast and entertainment industries have experienced significant expansion due to the growing demand for immersive visual experiences enabled by AR and VR technologies. These innovations are transforming the way people consume digital content by providing them with interactive and captivating experiences. wTVision, a company that specializes in real-time graphics and AR solutions for many industries, is at the vanguard of this transformation.

The dynamic, aesthetically captivating settings that wTVision's technology helps create for live broadcasts, sporting events, and virtual productions are essential. Achieving the desired level of realism that viewers anticipate requires a smooth transition between computer-generated imagery ([CGI](#)) and real-world scenes. However, achieving this level of visual fidelity remains challenging, particularly in faithfully replicating real-world lens distortion within the virtual scene.

In AR and VR settings, distortion can cause errors that impair the user experience. This difficulty is increased in applications that use varying zoom and focus settings since each one needs to be precisely calibrated in order to guarantee proper alignment of the virtual and real elements. At the moment, the time-intensive process of calibrating a camera takes 4 to 6 hours and involves manual adjustments. The repetitive nature of the work and the presence of a human in the loop make the process greatly inefficient.

The limits of manual calibration approaches become more evident as the demand for AR and VR applications grows, especially in live broadcasts and virtual events. An automated solution that can speed up the calibration process as well as increase accuracy is needed. As such, the project's goal is to create a calibration algorithm that makes use of cutting-edge tools like image

analysis and machine learning. The aim is to achieve maximum precision while radically reducing the time needed for the calibration process through automation and optimization to solve the current approach's problems.

1.1 wTVision

wTVision is a company specialized in creating integrated broadcasting solutions, focusing on software development, graphics design and live operations. Founded in 2001, and part of the Mediapro Group, wTVision has become a significant player in the broadcasting industry. The company provides a wide range of services, including real-time graphics, playout automation, AR, virtual productions, and data distribution, serving live broadcasts, sports events, election coverage, entertainment shows, and news programs [3].

It became one of the main real-time on-air graphics and playout automation providers due to its flexible solutions that can be integrated with multiple major graphics engines on the market. From small one-time broadcasts to some of the most important sport events, wTVision takes part in more than 7,000 broadcasts annually and has experience in more than 120 countries. It has hundreds of specialists around the globe and offices in Portugal, Spain, Belgium, Brazil, Bolivia, United Arab Emirates, India and USA [4].

The sports industry is a primary focus for the company, and football is the dominant market in Portugal and many other countries. Artificial intelligence and computer vision present a wide range of new opportunities to be explored; these topics have been increasingly investigated by researchers over the past years [5].

1.2 Problem Definition

wTVision developed a tool called R^3 in order to create immersive virtual environments and seamlessly integrate virtual objects into real-world scenes, primarily for live television and broadcast productions. The software enables compelling AR experiences by overlaying virtual content onto footage with high precision. It includes advanced simulation of lens distortion, allowing virtual elements to match the geometry of real-world visuals. However, accurate integration requires manual calibration of distortion coefficients for each unique combination of a camera's zoom and focus settings. Without this calibration the virtual elements distortion will

not match the same distortion as the real ones.

This discrepancy breaks the visual coherence between the virtual and real components of the scene, potentially reducing the realism and immersion of the AR experience. The objective of this project is not to correct or remove lens distortion but to replicate it accurately within the virtual content. By applying the same distortion profile to virtual elements, the final composite image will maintain consistency, making the augmented visuals appear more naturally integrated into the distorted real-world background.

Despite its lens distortion simulation capabilities, the current workflow of the R^3 software presents several operational challenges. It relies heavily on operator expertise, involves a time-consuming manual calibration process and lacks automation for handling dynamic changes in camera zoom and focus. These limitations highlight the need for a more efficient, scalable, and user-friendly calibration approach.

The primary goal of this project is to improve the calibration process, making it faster, more reliable, and less dependent on the expertise of the operator. More specifically, this thesis addresses the following research question: can the calibration parameters required by the R^3 workflow (center-shift, FoV, and radial distortion coefficients) be estimated automatically from rendered imagery with accuracy comparable to the established manual procedure, while reducing operator effort and execution time?

In this work, the values obtained through the manual R^3 calibration procedure are treated as a practical reference (“ground truth”) for comparison. They should not be interpreted as an error-free physical measurement, since small uncertainties may remain (e.g., operator-dependent adjustments and parameter step granularity). The success of this solution will be evaluated based on:

1. **Accuracy of Distortion Coefficient Estimation.** Evaluating the accuracy of the proposed model presents certain challenges. In particular, relative error can be misleading when ground truth values are very small, as even minor deviations can result in disproportionately large percentage errors. Moreover, a larger relative error does not necessarily translate into poorer qualitative calibration results.

To address this, the model will be evaluated by simulating images with known distortion parameters using the R^3 software. Experimental results will be qualitatively compared to these ground truth distortion coefficients to assess the model’s performance. The objective

is to achieve a correlation coefficient of at least 0.95 between the estimated and ground truth values for the Field of View (FoV), the primary radial distortion coefficient (k_1), and the centershift parameters. For the secondary radial distortion coefficient (k_2), a slightly lower threshold of 0.85 is deemed acceptable.¹ These thresholds ensure very strong correlations overall, while placing greater emphasis on FoV and the primary radial distortion coefficient (k_1) to maximize efficiency. These limits were defined from a production-oriented perspective and proposed by the company.

2. **Reduction in Calibration Time.** A key performance goal is to reduce the total calibration time to around 30 minutes, presenting a substantial reduction compared to the traditional manual process, which typically requires between 4 and 6 hours. This duration was deemed acceptable by the camera calibration operators.
3. **Consistency Across Multiple Zoom and Focus Levels.** The method will be validated across a range of zoom and focus settings. Consistency will be evaluated by comparing calibration results across these different configurations to ensure reliable performance throughout the lens's operational range.
4. **Automation of the Calibration Process.** The proposed system aims to minimize manual intervention by automating the calibration workflow as much as possible. This includes automatic feature detection, distortion model fitting, and coefficient estimation.

1.3 Document Overview

This thesis is organized into the following chapters:

- **Chapter 1 – Introduction:** Introduces the context and motivation behind this work, highlighting the challenges of manual camera calibration in AR systems and defining the objectives and scope of the project.
- **Chapter 2 – State of the Art:** Reviews the foundational concepts of AR, lens distortion, and the R^3 software. It presents common calibration techniques, analyzes the limitations of manual and OpenCV-based methods, and compares them with commercial and deep learning-based solutions.

¹The k_2 coefficient is generally considered less critical in most practical calibration scenarios. Field operators often omit k_2 during manual calibration under time constraints, as its inclusion typically yields only marginal qualitative improvements.

- **Chapter 3 – Camera Parameters Calibration Algorithm:** Describes the implementation of an automated algorithm for calibrating camera parameters in AR systems using the R^3 software. The chapter details each component of the algorithm—from edge detection to parameter optimization—and presents a quantitative evaluation across various zoom and focus settings. It also introduces performance improvements via MockBoard and concludes with a deep learning-based alternative for estimating distortion parameters in the absence of physical calibration infrastructure.
- **Chapter 4 – Conclusions and Future Work:** Summarizes the developed system and its results. Also, it highlights key contributions such as distortion estimation and MockBoard-based optimization. Outlines future directions including realtime calibration, enhanced deep learning methods, and full system automation.

Chapter 2

State of the Art

This chapter presents the fundamental concepts related to AR, lens distortion, and the R^3 software, a specialized tool developed by wTVision for creating AR scenes. It begins by introducing the necessary background for these topics and subsequently review the existing literature on camera parameter estimation for virtual environment construction.

2.1 Theoretical Background

Before proceeding, we review the key concepts that motivate camera parameter estimation and the adaptation of the virtual scene to those estimates in AR.

2.1.1 Augmented Reality

AR is a technology that overlays digital information—such as images, sounds, or text—onto the real world. Unlike VR, which creates a fully immersive digital environment, AR enhances reality by adding interactive, computer-generated elements. For example, in mobile applications or games like *Pokémon GO* [6], digital characters appear as if they are part of the physical world when viewed through the device’s camera. AR is used in various fields, including gaming, retail, education, and training, to enrich the user experience with additional layers of digital content.

wTVision implements AR on video streaming platforms. Figures [2.1a](#) and [2.1b](#) show a frame from a video of an AR scene developed by wTVision.



(a) **Example of AR graphics integrated into a live sports broadcast.** The virtual elements are rendered over the real-world view of the stadium. This overlay demonstrates the alignment and stability required in AR pipelines, where precise camera calibration ensures that virtual content maintains consistent perspective and spatial coherence relative to the physical scene.



(b) **Set of virtual elements used in the AR scene shown in Figure 2.1a.** The overlay includes team flags, a stylized emblem, and a sponsor bar containing brand logos. These assets are rendered as part of the AR composition and are aligned with the real-world camera view through precise calibration. Proper anchoring and occlusion behavior rely on accurate camera tracking and lens profiling.

Figure 2.1: Example of an AR scene and its corresponding overlay developed by wTVision

2.1.2 Lens Distortion

In ideal pinhole cameras [7], straight lines in the world project as straight lines in the image. However, real optical lenses introduce geometric aberrations that violate this principle. Among these, the most common types of lens distortion are radial distortion and tangential distortion.

Radial distortion arises due to the nonlinear refraction of light rays at increasing angles of incidence—a phenomenon governed by principles of geometric optics. This manifests in images as the curvature of lines that should be straight, particularly near the image periphery. It causes straight lines to appear curved—either outward (barrel distortion, see Figure 2.2b) or inward (pincushion distortion, see Figure 2.2c).

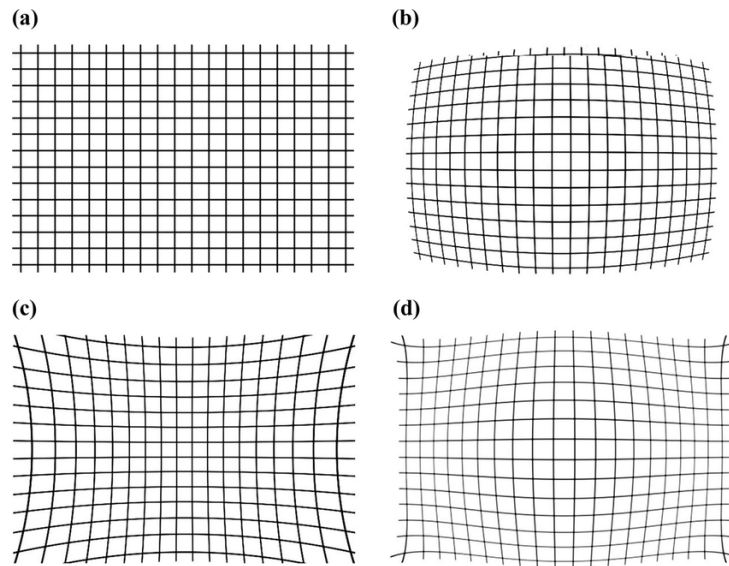


Figure 2.2: **Types of lens distortions.** (a) - No distortion, (b) - Barrel distortion, (c) - Pincushion distortion, (d) - Mixed types of distortion

Mathematically, the distortion is modeled as a function of the radial distance r from the optical center:

$$r = \sqrt{x^2 + y^2} \quad (2.1)$$

where x and y are the normalized undistorted coordinates in the camera plane.

The distorted coordinates are given by:

$$x_{\text{distorted}} = x \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \right) \quad (2.2)$$

$$y_{\text{distorted}} = y \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \right) \quad (2.3)$$

Here, k_1 , k_2 , and k_3 are the radial distortion coefficients. These terms represent higher-order corrections to model increasing nonlinearity at greater distances from the image center.

In general, lens distortion may also include tangential components; however, since the R3 workflow addressed in this project exposes only radial coefficients for calibration, the remainder of this thesis focuses on radial distortion.

2.1.3 R^3 software

In order to create, render and calibrate digital environments, wTVision created the R^3 software, comprised of an engine and a software platform. Within the platform, the R^3 *Spatial Designer* tool allows developers to build, layout, and arrange the virtual environment. While the engine handles rendering graphics, processing physics, and managing assets, R^3 *Space Designer* offers an interactive, visual interface for creating the look and feel of the virtual world.

The R^3 *Space Designer* environment comprises a set of integrated interface components specifically designed to facilitate the development of AR scenes, as illustrated in Figure 2.3. One of the fundamental elements is the *Scene Browser*, which presents the hierarchical structure of the 3D scene. This includes organizational elements such as layers, cameras, lights, and geometric primitives, thereby enabling structured scene management.

Another critical component is the *Render Surface*, which provides a real-time preview of the AR composition superimposed on a live video background. This module supports interactive control over navigation, coordinate alignment, and camera perspective, allowing users to evaluate the spatial coherence and integration of virtual elements within the real-world context.

Scene construction is further supported by the *Primitive Library*, which offers a predefined selection of commonly used components, including basic geometric shapes and lighting elements. These primitives can be readily incorporated into the scene to expedite the composition process.

Individual scene elements may be customized using the *Primitive Properties* panel, which exposes a range of rendering and visual configuration options, such as visibility toggles, blending modes, and background handling parameters. This level of control is essential for achieving precise visual integration between virtual and physical elements.

Another essential feature of the R^3 software is its capability to simulate and manipulate lens distortion effects with high accuracy. This functionality is accessible through the tracking

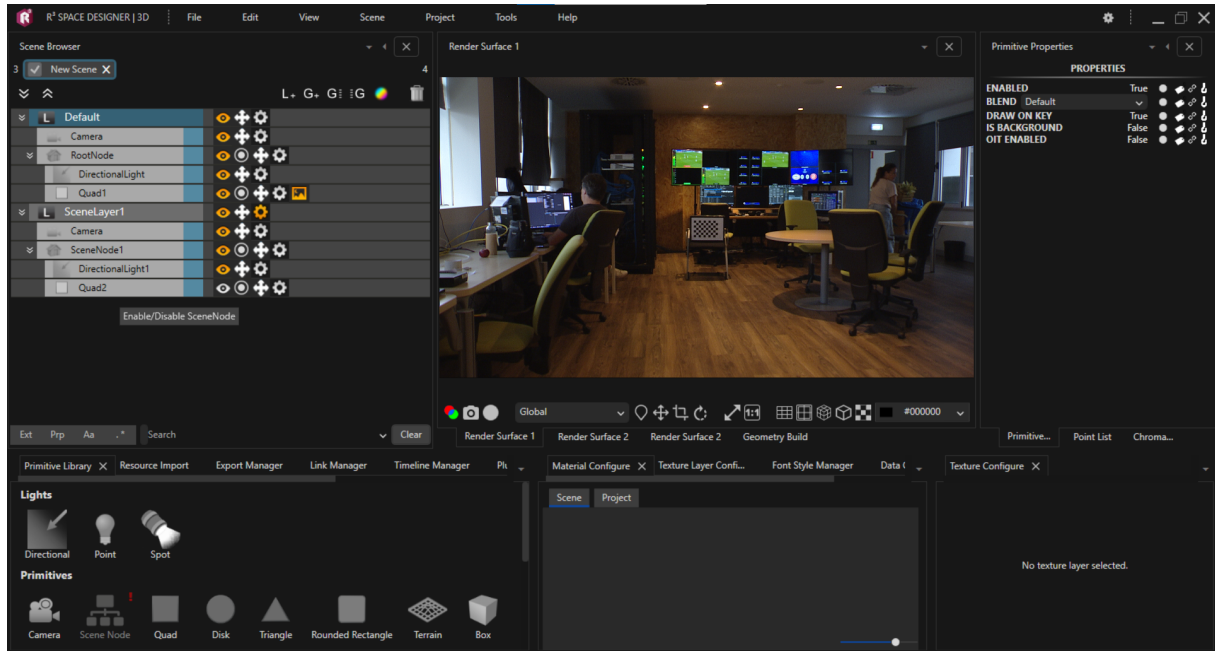


Figure 2.3: **Complete Interface View:** The full layout of the R^3 Space Designer environment, combining all UI panels—scene hierarchy, render preview, primitive library, and property editor—into a unified workspace for authoring AR scenes.

console, shown in Figure 2.4, which allows operators to manually adjust camera distortion parameters—including radial distortion coefficients, FoV, and optical center coordinates—in real time.

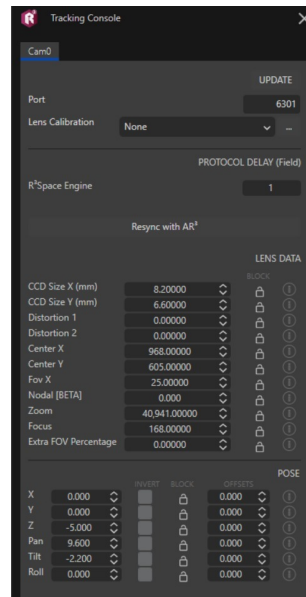


Figure 2.4: **Tracking Console Interface:** This interface enables fine-tuned manual calibration of the camera model by providing real-time control over Charge-Coupled Device (CCD) size, radial distortion coefficients (k_1 , k_2), optical center coordinates, FoV, and communication delay settings. These parameters are fundamental for achieving accurate alignment between the virtual camera and the physical optical system.

The significance of proper distortion calibration is demonstrated in Figure 2.5. When distortion parameters are not compensated, virtual objects appear spatially misaligned relative to the real world background, as shown in Figure 2.5c. In contrast, undistorted virtual content (Figure 2.5b) fails to reflect the actual optical path of the physical lens, resulting in perceptual inconsistencies. These observations underscore the critical importance of distortion correction for ensuring geometric coherence between virtual and real elements in augmented reality environments.



(a) **Background Image.** Static studio background used as a reference layer for virtual content placement.



(b) **Without Distortion Compensation.** Virtual objects rendered without modeling lens distortion appear geometrically inconsistent with the background.



(c) **With Distortion Applied.** Virtual elements rendered with simulated radial distortion, exhibiting more geometrical alignment between the virtual world and the real one.

Figure 2.5: Visual impact of lens distortion on augmented reality rendering. Accurate modeling and correction of lens distortion are essential for maintaining spatial consistency between virtual overlays and real-world backgrounds. Mismatches in distortion calibration can lead to noticeable visual artifacts, reducing immersion and realism.

This system significantly enhances the realism of augmented scenes by enabling real-time manual refinement of distortion parameters. It supports visual verification during calibration and facilitates more accurate camera model fitting.

2.2 Camera Calibration

In the context of virtual environment construction, camera calibration addresses the challenge of creating a seamless integration between real-world footage and computer-generated augmentations. A key issue in this process is lens distortion, optical imperfections introduced by the physical properties of the camera's lenses, which can disrupt the immersion of an AR experience.

To maintain visual coherence, the distortion present in the real scene must be replicated in the virtual elements. As stated previously, the objective is not to eliminate this distortion from the camera feed, but rather to apply the same distortion to the virtual objects so they appear naturally embedded within the scene. This alignment ensures that augmented elements blend seamlessly with the real world, preserving spatial and geometric consistency. [41]

2.2.1 Distortion Parameters

For accurate camera parameter estimation, several parameters must be estimated. These include the nodal point offset, the radial distortion coefficients k_1 and k_2 , the principal point offset (commonly referred to as the optical center or center-shift), and the Field of View (FoV). Some of these parameters can be mathematically derived using the focal length and the physical dimensions of the camera sensor. [7]

The FoV describes the angular extent of the scene captured by the camera. For a pinhole camera model, it can be computed from the focal length f and the sensor dimensions w (width) and h (height) as follows:

$$\text{FoV}_x = 2 \cdot \arctan\left(\frac{w}{2f_x}\right), \quad \text{FoV}_y = 2 \cdot \arctan\left(\frac{h}{2f_y}\right) \quad (2.4)$$

where f_x and f_y are the focal lengths in the horizontal and vertical directions, respectively, expressed in millimeters or pixels depending on the calibration context.

The center-shift (c_x, c_y) represents the optical center of the lens and is typically close to the image center. It accounts for any misalignment between the sensor and the optical axis. This offset is incorporated in the camera's intrinsic matrix¹. :

¹By *intrinsic parameters* we mean the pinhole-model terms that map 3D rays to pixels: focal lengths f_x, f_y , the principal point (c_x, c_y) , and optional skew s , grouped as shown in equation 2.5. Lens-distortion coefficients (e.g., k_1, k_2, p_1, p_2) are estimated separately and are not part of $\mathbf{K}$.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

The nodal point of a camera refers to the location along its optical axis where light rays ideally converge and where the camera should rotate to avoid inducing parallax. When the camera rotates around a point that does not coincide with its nodal point, nearby and distant objects appear to shift relative to one another—an effect known as parallax. Figure 2.6 illustrates this concept.



(a) **Scene as perceived with both eyes open.** This view represents the natural perception of depth and spatial arrangement enabled by binocular vision.



(b) **Scene as perceived with one eye closed.** The shift in relative position of the pen demonstrates the parallax effect caused by the change in viewpoint.

Figure 2.6: **Parallax effect observed with human vision.** Demonstration of the parallax effect in human vision. When one eye is closed, the loss of binocular cues results in a noticeable shift in perceived object positions.

To account for this, a nodal offset Δ_{nodal} may be introduced into the transformation as:

$$\mathbf{P}_{\text{camera}} = R \cdot \mathbf{P}_{\text{world}} + T + \Delta_{\text{nodal}}(z) \quad (2.6)$$

where $\mathbf{P}_{\text{world}}$ represents the 3D coordinates of a point in the world reference frame, $\mathbf{P}_{\text{camera}}$ represents the same point expressed in the camera coordinate system, R is the camera rotation matrix, T is the translation vector, and $\Delta_{\text{nodal}}(z)$ is a zoom-dependent offset that models the nodal point shift.

While some theoretical models allow for the derivation of focal length and related parameters from image geometry or metadata, in practical applications, most professional cameras do not provide real-time focal length or focus distance data. This limitation significantly impacts the calibration process. Accurate and dynamic parameter estimation—particularly for zoom and focus-dependent distortion—typically requires external hardware, such as lens encoders or dedicated interface modules. The necessity of such additional equipment not only increases system complexity and cost, but also restricts the applicability of automated calibration workflows to camera systems that support external hardware integration. As such, the reliance on external devices is a limitation in achieving hardware-independent calibration solutions.

Nevertheless, a variety of approaches have been proposed in the field of camera calibration and lens distortion modeling to address this issue. These methodologies can be broadly classified into two main categories: interpolation-based methods, and machine learning-driven approaches.

2.2.2 TrackMen Lens Calibration Tool

A commercial example of an automated system is the TrackMen Lens Calibration Tool [8]. This solution is designed specifically for the broadcast industry and supports Pan-Tilt-Zoom (PTZ) cameras. During calibration, the camera automatically cycles through a predefined range of zoom and focus settings while recording images of a 3D calibration object. The software then computes a full lens profile for each zoom/focus pair, effectively mapping out the lens's optical behavior across its range. This process is highly automated and typically completes in under 30 minutes, making it a good reference point for real-time applications.

These systems illustrate that considerable improvements in efficiency and usability can be achieved through the reduction or elimination of manual processes, the incorporation of automation, and the application of advanced modeling techniques. Nonetheless, the TrackMen lens calibration tool relies on an external hardware interface to acquire analog or digital encoder signals from the lens, specifically to extract zoom and focus position data. This hardware dependency also imposes a significant limitation: in cases where camera systems or lenses do not provide access to such encoder outputs, the automated calibration functionality becomes inapplicable. Consequently, the solution cannot be universally adopted across all camera configurations.

In contrast, the R^3 software manual calibration system adopts a zoom/focus interpolation strategy. Rather than relying on real-time encoder data, this method involves calibrating

the camera at a discrete set of zoom and focus positions. The resulting parameters are then interpolated to approximate values across the full zoom and focus range. While this technique significantly reduces hardware dependency and remains sufficiently accurate for many production scenarios, it introduces a substantial time cost: a complete calibration session typically requires between 4 to 6 hours. To mitigate manual workload, calibration-on-demand techniques—such as those proposed by Fuh et al. [9]—introduce motorized control over zoom and focus, thereby streamlining the parameter acquisition process. These methods offer a flexible alternative for systems where encoder data is unavailable or where the integration of specialized hardware is impractical.

2.2.3 Deep Learning-Based Radial Distortion Estimation from a Single Image

Recent advancements in deep learning have enabled Convolutional Neural Networks (CNN) to directly estimate camera distortion parameters from image content by learning spatial patterns associated with lens-induced deformations. These models leverage visual cues—such as curved edges, radial stretching, and non-linear perspective effects—to regress distortion coefficients, including the commonly used radial terms k_1 and k_2 , even in unstructured scenes.

2.2.3.1 The Brown–Conrady Radial Distortion Model

A widely adopted model for describing lens distortion is the Brown–Conrady model [10], which decomposes the total distortion into radial and tangential components. For many practical applications, the tangential components are negligible, allowing the distortion to be modeled exclusively by the radial terms. Since in this project only the k_1 and k_2 will be addressed, the distorted image coordinates (x_d, y_d) are then expressed as:

$$x_d = x \left(1 + k_1 r^2 + k_2 r^4 \right), \quad y_d = y \left(1 + k_1 r^2 + k_2 r^4 \right) \quad (2.7)$$

where $r = \sqrt{x^2 + y^2}$ denotes the radial distance from the image center, and k_1 and k_2 represent the second- and fourth-order radial distortion coefficients, respectively. These parameters are dimensionless when coordinates are normalized, and they encapsulate higher-order transverse optical aberrations, which are dependent on lens geometry and the internal refractive index distribution.

Recent studies in computer vision have explored the feasibility of regressing distortion coefficients from a single image using deep learning techniques [11]. CNNs can infer such parameters by detecting consistent geometric distortions in image features, even in the absence of structured calibration patterns.

To enable supervised training, large synthetic datasets are generated by applying known distortion profiles to undistorted images using analytical distortion models. Each distorted image is then paired with the corresponding ground truth distortion parameters (k_1, k_2) , serving as labels during training.

This data-driven approach eliminates the need for physical calibration targets and facilitates distortion estimation in unconstrained environments. For instance, Lopez-Antequera et al. [12] generate synthetic distortions from panoramic image data to train their regression models for real-world application.

2.2.3.2 Single-image CNN for k_1 and k_2 Estimation

In this work, we also tested a CNN-based approach for estimating the radial distortion coefficients k_1 and k_2 directly from a single distorted Red Green Blue (RGB) image. The model architecture is built upon ResNet-18 [1], which is initially pre-trained on the ImageNet dataset and subsequently fine-tuned on a custom synthetic dataset specifically designed for the task of radial distortion estimation. ResNet-18 was selected because its residual architecture enables effective feature extraction while maintaining a relatively small number of parameters, making it suitable for training on moderately sized datasets. Figure 2.7 provides an overview of the ResNet-18 architecture employed in this study.

Training samples are synthetically generated by applying the Brown–Conrady distortion model to undistorted images. The distortion parameters are randomly sampled from the following uniform distributions:

$$k_1 \sim \mathcal{U}(-0.4, 0.4), \quad k_2 \sim \mathcal{U}(-0.2, 0.2) \quad (2.8)$$

The network is trained using the Mean Square Error (MSE) loss function, defined as:

$$\mathcal{L} = \frac{1}{2} \left[\left(k_1^{\text{pred}} - k_1^{\text{true}} \right)^2 + \left(k_2^{\text{pred}} - k_2^{\text{true}} \right)^2 \right] \quad (2.9)$$

This objective encourages the model to minimize the squared difference between the predicted and true distortion parameters. By learning a mapping from image features to distortion

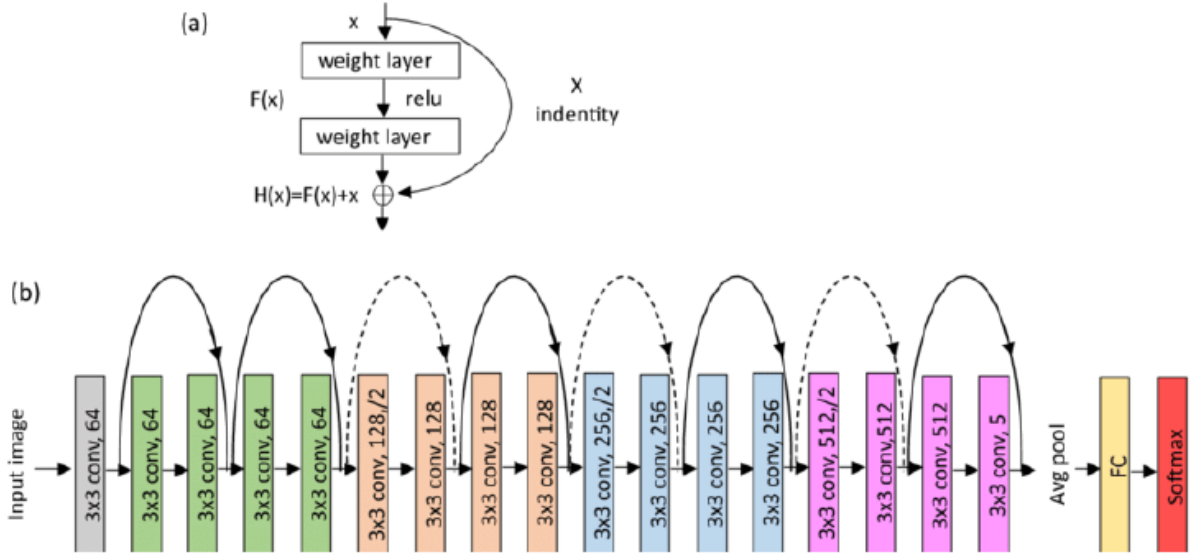


Figure 2.7: **ResNet-18 architecture used for distortion estimation:** [1] The model takes a single RGB image of size 256×256 as input and processes it through a series of convolutional layers and residual blocks. The final fully connected layer outputs a two-dimensional vector representing the predicted distortion coefficients k_1 and k_2 . This design enables the network to learn spatially-aware features that correlate with radial lens distortion.

coefficients, the network can generalize across a variety of distortion profiles without requiring prior geometric information.

Previous studies report that CNN-based distortion estimation methods may produce higher errors in textureless regions or under extreme distortion conditions due to the limited availability of reliable visual features [1], [12]. Despite these limitations, learning-based approaches remain a promising alternative when traditional calibration procedures are difficult to perform.

2.2.4 R^3 software Calibration

This section examines wTVision’s current approach to virtual environment calibration, which utilizes the R^3 software platform to align the geometry of the virtual environment with that of the real world. The method involves calibrating the camera at a set of discrete zoom and focus levels, followed by the interpolation of calibration parameters across intermediate values. This allows the system to maintain geometric consistency throughout a range of optical configurations.

2.2.4.1 Detailed Calibration Process

For the calibration process, a physical target in the form of a rectangle with known dimensions (width and height) and spatial coordinates is typically employed. By inputting this data into the

R^3 software, the operator can recreate a virtual counterpart that matches the real-world target in both size and scale within the virtual environment.

However, the proposed calibration method targets broadcast and industrial camera systems, where lens distortion is more pronounced than in typical consumer cameras. As access to such equipment was not available during this work, the real-world setup was simulated using a checkerboard pattern to replicate the calibration target.

Center-Shift Calibration

The calibration process begins with the use of a real-world reference point. In practical scenarios, this is typically generated by the flashlight of a mobile device, producing a visible white circle. A simulated representation of this setup is shown in Figure 2.8a. Subsequently, a virtual cone is introduced into the scene, as illustrated in Figure 2.8b. This cone does not require any specific properties, as it merely serves as a visual alignment aid. The center-shift parameters are then adjusted, as depicted in Figure 2.8c, until the tip of the cone is precisely aligned with the center of the white circle.

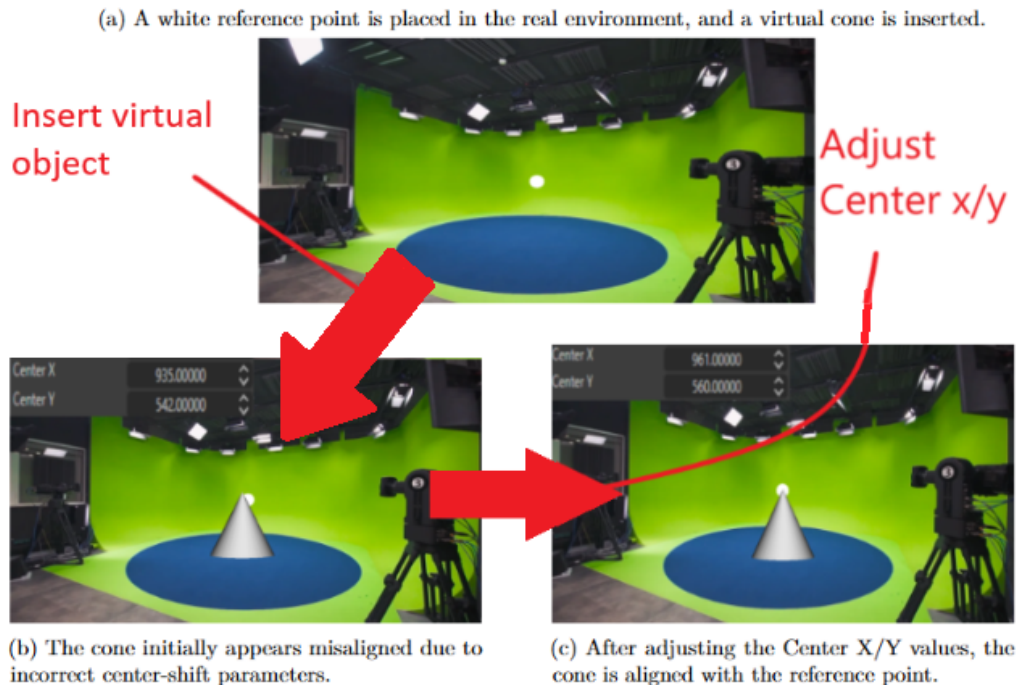


Figure 2.8: **Center-shift calibration procedure:** This sequence illustrates the process of aligning a virtual object's position with a real-world reference point using center-shift parameter adjustment.

Once alignment is achieved, it is important to verify that the cone’s tip remains centered even after zooming in. If a misalignment is observed at closer focal lengths, further refinement of the center-shift parameters is necessary to ensure consistent calibration.

Experimental validation of the manual calibration procedure showed that, when the camera’s tilt, pan, and roll are all set to zero, the center-shift values correspond precisely to the pixel coordinates of the white circle’s center. This conclusion was confirmed through manual alignment and further validated using an edge detection program to extract the center point’s coordinates.

Field of View Calibration

The estimation of the FoV begins by introducing a reference target—a red rectangle—into the scene. This target should be placed off-center, as illustrated in Figure 2.9a. A virtual object of identical dimensions and spatial coordinates is then inserted, as shown in Figure 2.9b. The virtual object is aligned with the target by iteratively adjusting the virtual camera’s FoV parameters until both objects match in position and apparent scale, as demonstrated in Figure 2.9c.

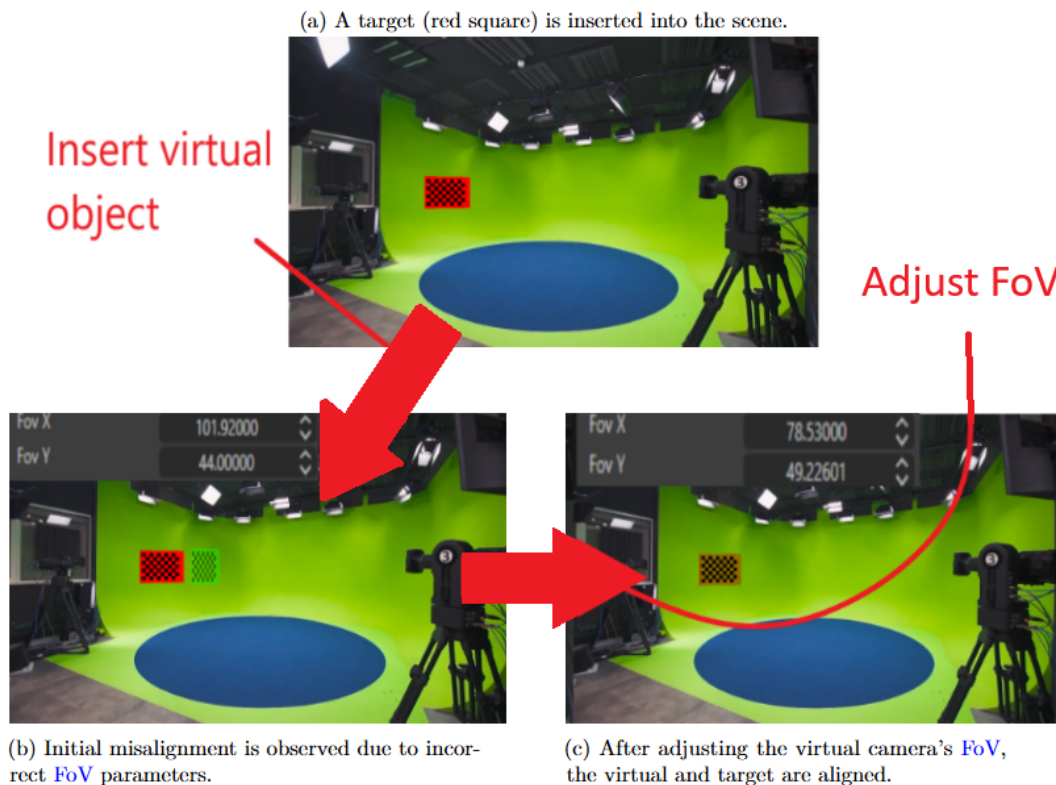


Figure 2.9: **FoV calibration procedure:** Calibration workflow used to align a virtual object with a reference by tuning the virtual camera’s field of view. Beginning with the insertion of a virtual rectangle adjacent to a real-world reference marker, the procedure involves iteratively adjusting the FoV parameters to correct spatial discrepancies.

To ensure robustness across focal lengths, this procedure is repeated at seven discrete zoom levels, thereby accounting for non-linear distortion effects. The overall calibration workflow is summarized in Figure 2.9.

k_1 and k_2 Calibration

The calibration of the radial distortion coefficients k_1 and k_2 involves aligning virtual objects with corresponding real-world reference markers under controlled positioning.

To calibrate k_1 , the target should be positioned along one of the horizontal edges of the image frame while remaining vertically centered, as shown in Figure 2.10a. A virtual object with matching dimensions and coordinates is then inserted into the scene (Figure 2.10b). The value of k_1 is iteratively adjusted until the virtual and real objects are visually aligned (Figure 2.10c).

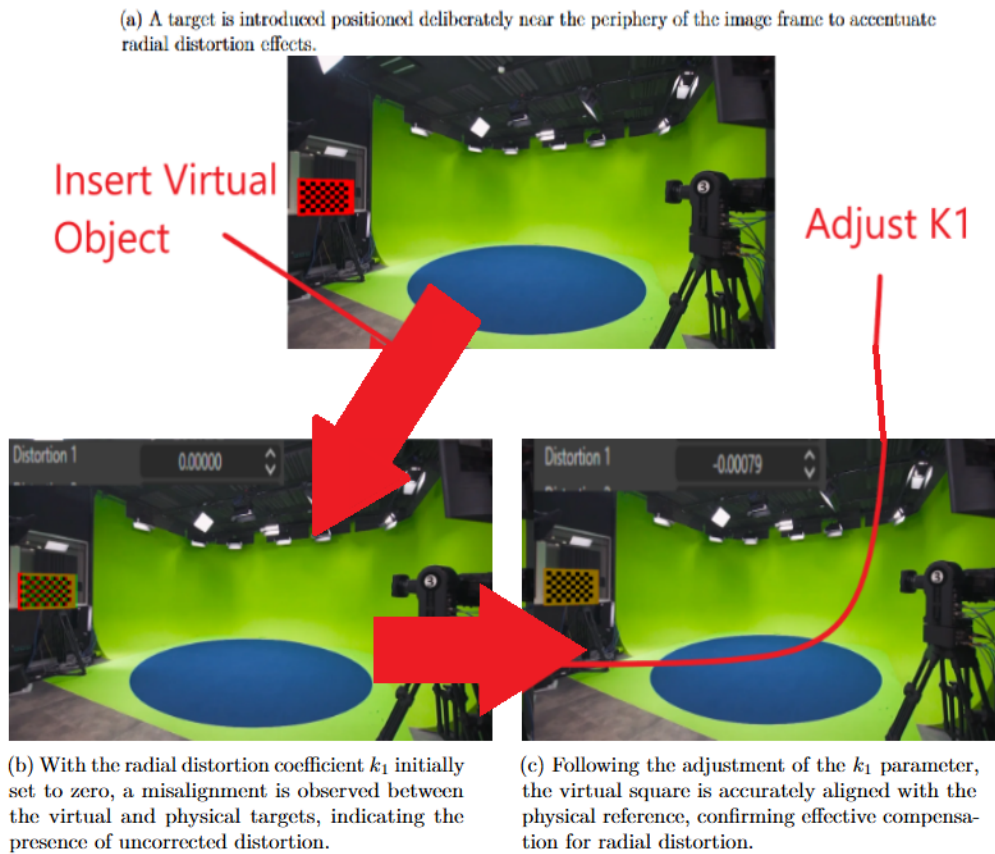


Figure 2.10: **Radial distortion calibration for the k_1 coefficient:** Procedure for correcting radial lens distortion by aligning a virtual object with a real-world reference placed near the edge of the image frame. The off-center positioning amplifies the effect of radial distortion, making discrepancies more perceptible. By iteratively adjusting the k_1 parameter, the distortion is minimized until the virtual and target objects are visually aligned, thereby achieving an accurate calibration.

This procedure should be repeated across all predefined zoom levels to account for varying distortion characteristics. The complete k_1 calibration workflow is depicted in Figure 2.10.

The calibration of k_2 follows a similar methodology. However, in this case, the real-world reference marker is placed near one of the corners of the image frame to emphasize the higherorder distortion effects that k_2 is intended to correct. Figure 2.11 illustrates the recommended placement of calibration targets for both k_1 and k_2 . The diagram was drawn based on guidelines provided by a wTVision operator.

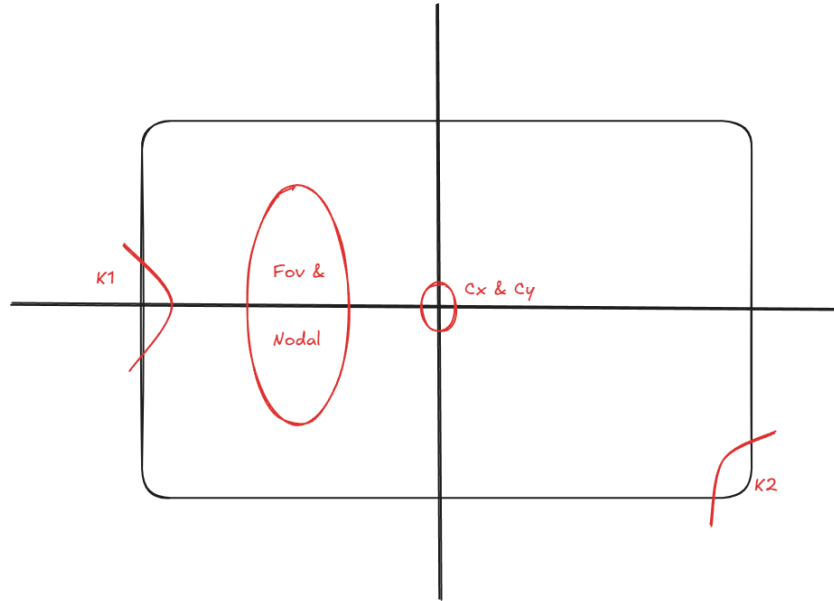


Figure 2.11: **Target positions used during calibration:** Placement of real targets in the image for each stage of the calibration process: center-shift, FoV, k_1 , k_2 , and nodal point. Each target position is chosen to emphasize the type of distortion or misalignment relevant to the parameter being adjusted.

2.2.4.2 Nodal Point Calibration

To achieve the no parallax effect², the camera must be aligned to rotate around its nodal point. This eliminates visual misalignments caused by parallax when the camera pans or tilts. The following step-by-step procedure is used to estimate the nodal point coefficient in the calibration software:

1. Place two identical reference objects (e.g., markers or printed patterns) in front of the camera, aligned along the same line of sight:

²The no parallax effect occurs when the camera rotates around its nodal point, preventing apparent shifts between foreground and background objects. This is crucial for seamless AR alignment, panoramic photography, and image stitching.

- **The front object** should be positioned approximately 1 meter from the camera.
 - **The rear object** should be placed directly behind it, at a distance of around 5 meters.
2. Adjust the camera position and orientation so that, when viewed through the lens, the rear object is completely hidden behind the front one.
 3. Slowly pan the camera horizontally (e.g., from -10° to $+10^\circ$), and observe the two targets:
 - **If** the rear object begins to shift into view (i.e., is no longer occluded), this indicates that parallax is present.
 4. Adjust the nodal point coefficient in the software until the rear target remains hidden behind the front target during the full range of panning motion.
 5. Repeat this test across several zoom levels to ensure that the nodal point setting holds consistently for different focal lengths.

This calibration ensures that virtual content behaves naturally when the camera moves, maintaining spatial consistency between real and virtual objects.

2.2.4.3 Challenges of Current Method

The current solution, described in this section, is effective but has a few limitations.

- Time-intensive calibration process.

Coefficient	Calibration Time (min)
Center-shift	5 – 10
FoV	60 – 90
k_1	60 – 90
k_2	60 – 90
Nodal	60 – 90

Table 2.1: Estimated calibration times for each coefficient based on information provided by a wTVision operator.

- Reliance on operator expertise for accurate results.
- Lack of automation for dynamic zoom and focus adjustments.

The manual calibration process implemented in wTVision’s R^3 software provides a high degree of flexibility and control, allowing expert operators to fine-tune the camera’s lens parameters. However, the approach suffers from significant drawbacks, particularly in terms of time consumption.

As summarized in Table 2.1, the full calibration procedure can take several hours and must be repeated across multiple zoom and focus levels. Furthermore, maintaining consistency across sessions or operators is challenging, and the absence of automation restricts scalability for large-scale or time-sensitive productions.

These challenges underscore the need for a more efficient, accurate, and automated approach to virtual environment calibration. The goal is to replicate the calibration process algorithmically and enhance it with computer vision techniques.

2.3 Summary

This chapter has presented a comprehensive overview of camera calibration and distortion estimation techniques, with a particular focus on their relevance to AR broadcasting applications. In such systems, as implemented by wTVision, accurate geometric alignment between the physical and virtual scenes is critical—especially under variable zoom and focus conditions. Robust camera calibration is thus a fundamental prerequisite for achieving spatial consistency in real-time augmentations.

Traditional methods are well-established and deliver reliable results in controlled environments. However, these techniques are constrained by manual procedures, dependence on calibration targets, and limited adaptability in live production scenarios. Commercial tools such as TrackMen [8] mitigate these constraints through automation but introduce a dependency on specialized hardware, which limits flexibility and portability across different setups.

The primary objective of this work is to develop an automated calibration framework for the wTVision R^3 platform that minimizes human intervention and operates independently of external hardware or deep learning components. The proposed method seeks to replicate the existing manual calibration process using classical computer vision algorithms and optimization techniques.

In parallel, we have conducted an exploratory study into learning-based alternatives for radial

distortion estimation. This investigation assessed the feasibility of integrating deep learning into the calibration pipeline, particularly for real-time or low-infrastructure scenarios.

The main contributions of this work are:

- The design and implementation of an automated calibration system that replicates and streamlines the manual calibration workflow used in the R^3 platform, reducing operator dependency while maintaining geometric accuracy.
- An evaluation of deep learning–based distortion estimation methods for single images, assessing their potential integration into traditional calibration pipelines as a complementary or hybrid approach.

By combining classical computer vision principles with insights from data-driven methods, the proposed solution aims to deliver a robust and adaptable calibration pipeline tailored to the demands of modern AR broadcasting environments.

Chapter 3

Camera Parameters Calibration Algorithm

The main objective of this internship was the development of an automated calibration algorithm for AR camera systems. The goal was to replicate, through software, the manual calibration procedure traditionally performed by an operator using the R^3 software interface.

To accomplish this, four primary tasks were defined. First, it was necessary to establish communication with the R^3 software to enable automated transmission of configuration commands. Second, a computer vision algorithm was developed to detect the pixel coordinates of the four corners of a quadrilateral calibration target, as well as its centroid. Third, the centroid coordinates were used to calibrate the center-shift parameters of the virtual camera. Finally, the coordinates of the target's corners were used to iteratively calibrate the FoV and the radial distortion coefficients k_1 and k_2 .

The algorithm calibrates each coefficient by detecting the coordinates of the target rectangle's corners and incrementally adjusting the parameter value in small steps. After each adjustment, the error is measured as the pixel distance between corresponding corners of the virtual and real ¹ targets. The parameter is considered calibrated once this error falls below a predefined threshold or if the algorithm detects overshooting. At each iteration, the best-performing value is retained based on the minimum determined corners distance.

Since the R^3 software does not provide direct access to the virtual target's corners coordinates,

¹Since a real camera was not used in this project, every scene component was simulated. When referring to a real target in this thesis, we are referring to a simulated red checkerboard, while the virtual target is an identical green checkerboard.

the same edge detection procedure used for the real target must be applied to extract them from rendered images. Once the real target’s coordinates have been recorded, the scene’s background is disabled, leaving only the virtual target visible for the iterative calibration.

Nodal point calibration was not addressed in this project. This stage of the calibration process requires a physical camera connected to the R^3 system, as the nodal adjustment depends on the real camera’s optical and mechanical properties and its interaction with the tracking system. Since access to such hardware was not available during this work, nodal calibration could not be performed and remains outside the scope of the current implementation.

The following sections describe each stage of the algorithm in detail, including the adopted methodology, encountered challenges, and the limitations and advantages of the proposed solution.

3.1 Communication Between Python and R^3

To enable automation of the calibration process, a communication interface was established between Python and the R^3 software using socket-based Inter-Process Communication (IPC). In this architecture, Python acts as the controlling environment, issuing calibration commands and receiving feedback from R^3 . The R^3 software already provides a socket-based interface, which was used to allow Python scripts to control camera parameters such as FoV, center-shift, and radial distortion coefficients.

The connection is established locally via the loopback interface, ensuring low-latency data transfer while avoiding external network dependencies. This setup allows Python scripts to remotely control R^3 parameters, such as the FoV, center-shift, and radial distortion coefficients, without requiring any direct modification to the R^3 software itself.

In practice, the communication interface enables Python to coordinate image acquisition and camera parameter updates in real time. Computer vision routines implemented in Python extract relevant visual features—such as the position of a calibration target—and use this data to iteratively adjust virtual camera parameters within R^3 . Because the two systems operate independently, the architecture remains robust to changes in either component, supporting future extension and potential integration with alternative calibration engines or computer vision frameworks.

Overall, this communication layer serves as the foundation of the automated calibration system, bridging the computational capabilities of Python with the proprietary rendering and configuration features of R^3 .

3.2 Edge Detection Algorithm using OpenCV

OpenCV is a widely used, open-source library for real-time computer vision and image processing. Written in C++ with bindings for Python, Java, and others, it provides optimized implementations of core algorithms such as filtering, feature detection and matching, segmentation, object tracking, optical flow, and camera calibration. Because of its breadth, efficiency, and active community, OpenCV is a standard toolkit in research and industry applications including robotics, augmented reality, medical imaging, surveillance, and autonomous systems.[13]

Even though this library does not directly compute the coefficients required for parameter estimation, it remains highly valuable due to its robust image processing capabilities. Among these, the Canny edge detection algorithm [2] plays a central role in this work by enabling the robust identification of calibration target boundaries under varying conditions.

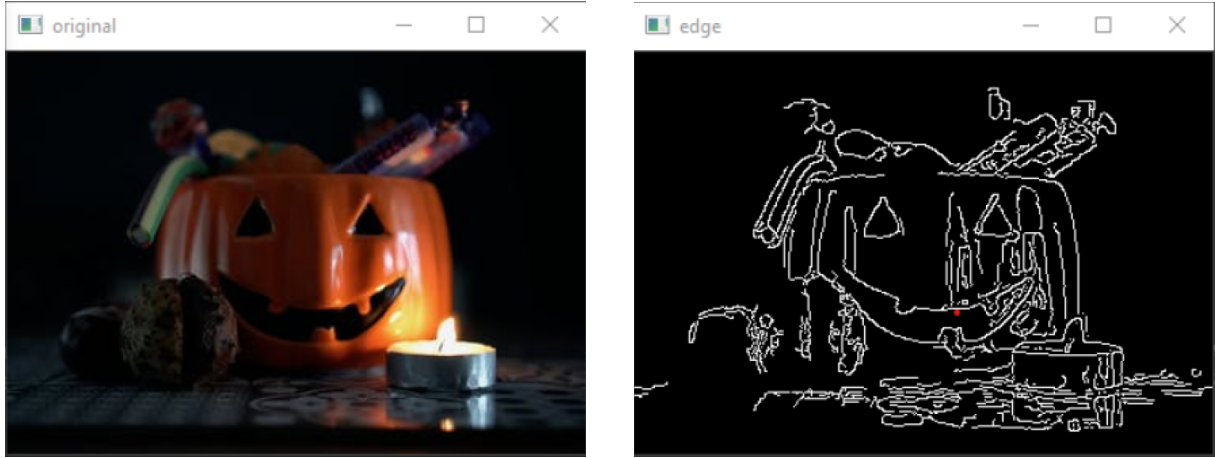
Canny edge detection is a well-established, multi-stage algorithm that aims to identify edge pixels corresponding to significant intensity changes in an image. The process begins with Gaussian blurring to reduce image noise and enhance stability. The gradient is then computed at each pixel, yielding two components (in the horizontal x and vertical y directions) that represent local changes in image intensity. From these components, the *gradient magnitude* (strength of the edge) and the *gradient direction* (orientation of the edge) are derived, which are then used to determine candidate edge pixels. These candidates are refined via non-maximum suppression, which thins the edges by retaining only local maxima of the gradient. Finally, a double-thresholding mechanism is applied, followed by edge tracking through hysteresis. This last step classifies edges as strong, weak, or irrelevant, and connects weak edges to strong ones if they form a continuous boundary [2].

An example illustrating the result of applying Canny edge detection is shown in Figure 3.1, where object contours are effectively extracted from a noisy input image.

In the context of this project, edge detection is used to identify the external contour of a rectangle-shaped calibration target. Once edges are detected using the Canny algorithm¹, OpenCV's contour approximation functions are employed to extract polygonal curves and identify the vertices and centroid of the target object [13]. These geometric features are essential inputs for computing camera parameters such as center-shift, FoV, and radial distortion coefficients.

The reliability of Canny-based detection across a variety of lighting conditions and distortion

¹Object detection approaches such as YOLO were considered; however, bounding-box detectors are less suitable when the calibration target appears distorted due to lens effects. Edge-based methods allow more precise localization of the target contours.



(a) **Original image.** Input image used for edge detection, depicting a pumpkin and surrounding objects.

(b) **Edge detection result.** Output of the Canny edge detection algorithm highlighting the detected contours.

Figure 3.1: **Demonstration of Canny edge detection on a real-world image.** Figure 3.1a contains various objects with distinct textures and lighting conditions. Figure 3.1b shows the extracted edges using the Canny algorithm, effectively outlining object contours despite shadows and noise [2].

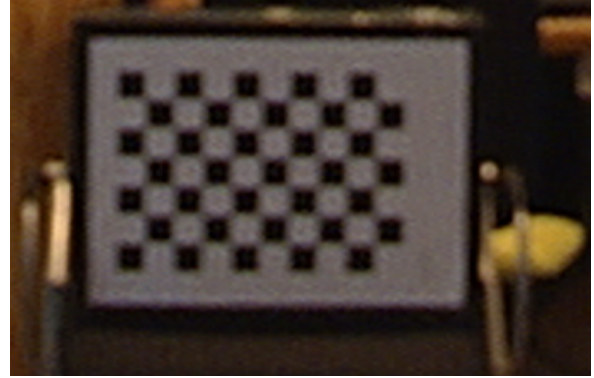
levels makes it especially well suited for AR calibration scenarios. Its performance under realworld and simulated conditions was found to be stable, enabling accurate feature localization required by the iterative calibration algorithm developed in this work.

Figure 3.2 illustrates the edge detection algorithm applied to a sample image. Figure 3.2a displays the original input frame containing a centered white checkerboard target. Figure 3.2c shows the output of the Canny edge detection algorithm, in which the contours of the target have been successfully identified. Figures 3.2b and 3.2d provide magnified views of the original and processed images, respectively, offering a closer examination of pixel-level precision in contour detection and edge alignment. These results confirm the reliability of the edge detection method for extracting geometric features.

The edge detection algorithm constitutes a foundational component of the camera parameter estimation pipeline. Following its implementation, the algorithm was rigorously tested under a variety of distortion scenarios to evaluate its robustness and reliability. The results of these tests, along with corresponding analysis, are presented in the following section.



(a) **Original input image.** The white checkerboard calibration target is centrally positioned.



(b) **Zoomed view of the input image.** Magnified region of the checkerboard target.



(c) **Edge detection output.** Result from the Canny-based edge detection algorithm. The algorithm successfully outlines the contours of the calibration target.



(d) **Zoomed edge detection result.** Detailed view of the extracted geometry, confirming detected corners and displaying edge lengths (in pixels).

Figure 3.2: **Edge detection applied to a checkerboard calibration target.** Illustrates the edge detection process.

3.2.1 Testing with Blender

To evaluate the performance and robustness of the edge detection program under controlled yet diverse visual conditions, a comprehensive set of test images was required. However, the limited availability of a physical camera—due to restricted access to studio hardware—posed a significant obstacle to acquiring consistent, high-quality image data. To address this constraint, a simulation-based testing methodology was adopted using Blender, an open-source 3D computer graphics suite widely employed in animation, visual effects, game development, and scientific visualization.

Blender offers an advanced rendering engine, a flexible node-based material system, and precise control over camera parameters and scene geometry, making it particularly well suited for simulating photorealistic image data. This capability enabled the generation of custom test

datasets that closely mimic the optical distortions and perspective shifts observed in real-world camera imagery, while maintaining total control over variables such as lighting, pose, and lens configuration.

For the purposes of this evaluation, a virtual scene was constructed in Blender that included:

- A planar calibration target modeled after a checkerboard, featuring a known geometric structure (rectangle grid) that facilitates precise corner detection.
- A virtual camera whose intrinsic parameters (e.g., focal length, sensor size) could be configured to simulate a variety of real camera settings.
- A flexible rendering pipeline capable of simulating radial lens distortion using Blender’s distortion shader nodes.
- Controlled environmental lighting to test the algorithm’s performance under varying shadow, contrast, and illumination conditions.

The simulation setup allowed for systematic testing of the edge detection system across the following variations:

- **Viewpoint variation:** The virtual camera was positioned at different angles—above, below, oblique, and frontal—to assess the algorithm’s ability to detect edges in non-orthogonal projections.
- **Target rotation and scale:** The calibration target was rotated and resized to test for invariance to orientation and scale.
- **Distortion levels:** Radial distortion was incrementally increased from negligible to exaggerated to evaluate the program’s tolerance and consistency across different distortion regimes.
- **Perspective changes:** The camera’s FoV was modified, mimicking different zoom levels and wide-angle conditions.

The result was a dataset of synthetic but highly realistic images tailored specifically for benchmarking the calibration support tools developed in this work. This simulation-based

approach not only enabled rapid prototyping and validation of algorithms, but also provided a means of reproducibility—allowing for future experiments to be repeated under identical conditions.

Ultimately, the use of Blender as a synthetic testing environment significantly enhanced the experimental rigor of the edge detection evaluation, allowing for detailed stress testing that would have been impractical using only physical hardware.

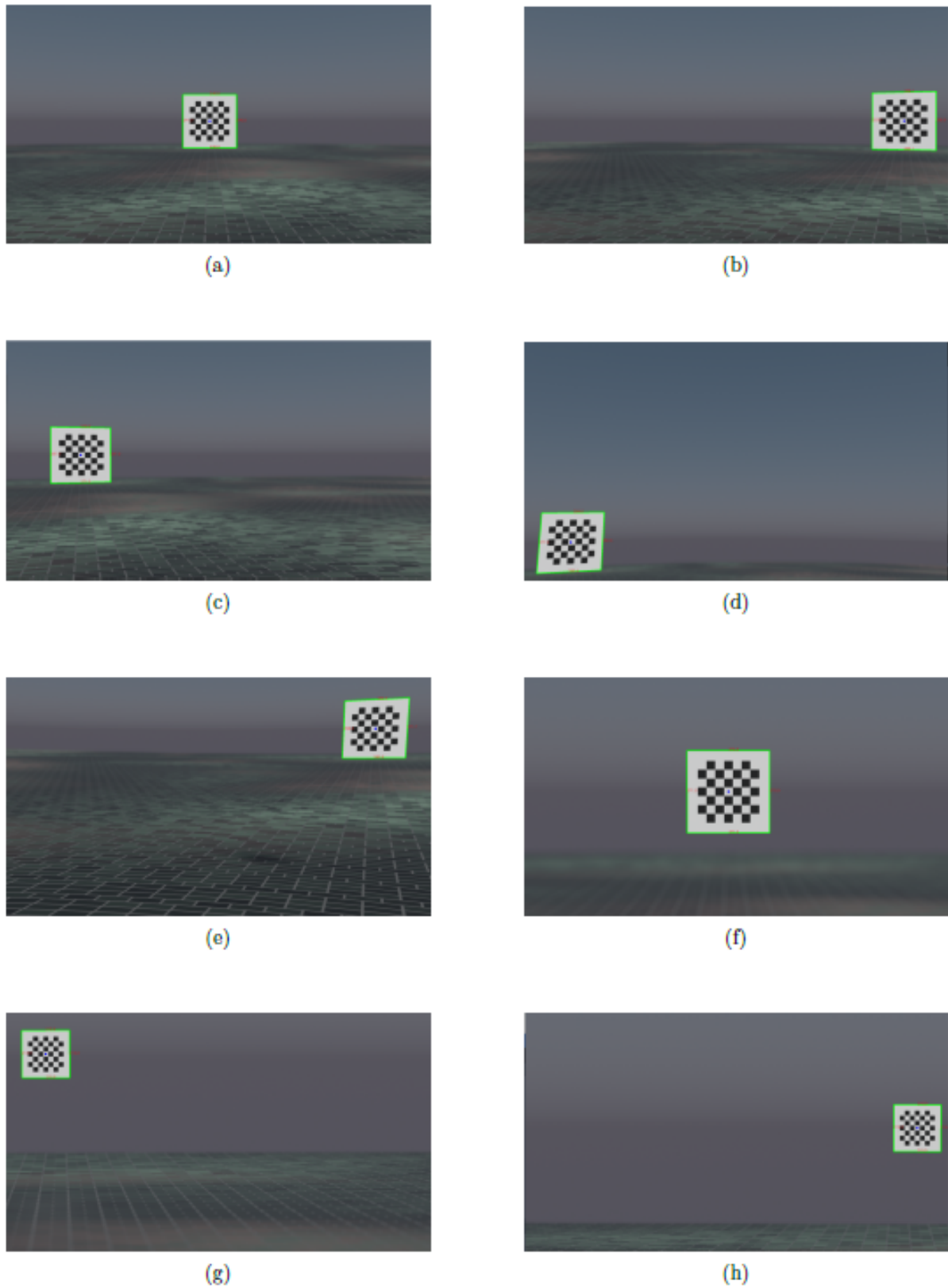
3.2.2 Results and Discussion

Some representative results of the edge detection algorithm applied to Blender-generated images are presented in Figure 3.3. These test cases encompass a variety of simulated camera views and slight distortion levels, including changes in orientation, depth, and perspective. Each image depicts the same calibration target rendered under different scene configurations to evaluate the generalizability and robustness of the detection pipeline.

The results depicted in Figure 3.3 demonstrate that the edge detection algorithm consistently identified the salient geometric features of the synthetic calibration target across a wide range of scenarios. Despite the variations in perspective, distortion magnitude, and orientation, the algorithm reliably extracted clean and well-defined edges from the rendered images. This suggests that the detection pipeline is sufficiently robust for use in a virtual calibration workflow, where geometric accuracy is essential for computing distortion and alignment parameters.

Using Blender for synthetic image generation offered a controlled, repeatable environment to evaluate the system. By simulating camera behaviour—including lens distortion, focal-length variation, and spatial offsets—we tested the detection algorithm without a physical camera or studio. This reduced testing time and enabled systematic validation of the image-processing pipeline prior to deployment. Nonetheless, the rendered scenes use simple backgrounds that do not fully reflect real-world complexity, so the findings should be corroborated with real footage.

However, transitioning from synthetic to real-world data introduced additional challenges. In practical settings, ambient lighting, reflections, shadows, and background textures can interfere with edge clarity and contrast, leading to occasional misidentification or incomplete detection of the calibration target. During early real-world tests, the algorithm occasionally failed to



differentiate the white checkerboard from other light-colored objects or highlights present in the scene. For example, a rectangular room light could be mistakenly identified as the calibration target. Although this occurrence was relatively rare, it was noticeable enough during experimentation, justifying a change in target design. To mitigate this issue, the checkerboard target color was changed from white to red. Red was chosen specifically for its low prevalence in typical indoor scenes and for its strong contrast against the green backgrounds frequently used in chroma key (green screen) production environments.

This modification was made with the expectation of improving detection reliability and reducing false detections in real-world usage.

If the target contour cannot be detected, the calibration step cannot proceed because the algorithm requires the target’s geometric features. In such cases, the operator must reposition the calibration target and repeat the detection process until the contour is successfully identified.

Overall, these experiments confirmed that the edge detection algorithm is both effective and generalizable when applied to simulated content and that simple modifications—such as color changes—can significantly enhance its robustness in real-world applications. These insights informed subsequent stages of the calibration pipeline, where accurate localization of target features remains essential.

3.3 Camera Calibration Algorithm

This section provides a detailed explanation of the development of a calibration algorithm designed to iteratively determine camera parameters.

To validate the algorithm, virtual images were employed for experimentation. These images were generated using the R^3 software with known calibration parameters, which were selected based on the outcomes of a prior manual calibration procedure performed at wTVision. This ensured that the simulated scenes closely approximated realistic use cases. In this setup, a red checkerboard was used to simulate the real-world target, while all other elements in the scene were considered virtual objects.

The generated images were organized in a directory structure based on different zoom and focus configurations. For each specific combination—such as *Zoom 0*, *Focus 0*—a set of four images was provided, each corresponding to the calibration of a specific parameter: center shift, FoV, k_1 , and k_2 . It is important to note that the zoom and focus indices (e.g., “0”) do not represent physical values such as focal length or aperture size, but rather normalized units defined within the calibration process, ranging from a minimum to a maximum. The lens calibration procedure itself is used to establish the relationship between these normalized indices and the corresponding physical parameters.

The objective of the calibration algorithm is to accurately determine the parameters associated with the simulated images. These estimated values are subsequently compared against the ground truth to evaluate the precision and effectiveness of the calibration process.

Prior to initiating the calibration procedure, it is necessary to measure the dimensions and position of the real target (red checkerboard). The origin of the coordinate system is defined at the floor level, directly beneath the optical center of the camera. Based on these measurements, a corresponding virtual object is placed within the R^3 environment. To ensure consistency and eliminate rotational effects during calibration, the camera's pan, tilt, and roll values are set to zero.

3.3.1 Center-shift calibration

For center-shift calibration, the R^3 software was used to insert a virtual object with the same pattern, dimensions, and coordinates as the real target into the rendered scene.

The center-shift calibration follows a systematic procedure. First, the camera is set to its minimum zoom level, and the red checkerboard target is placed at the center of the frame. The edge detection program is then used to compute the target's center coordinates, which are subsequently used to compute new center-shift values.

The complete process is visualized in Figure 3.4. The red target is inserted into the center of the frame as shown in 3.4a. In 3.4b, a virtual cone is inserted into the R^3 environment with an uncalibrated center-shift. The detected coordinates of the target center are shown in 3.4c. Finally, 3.4d shows the cone correctly aligned after calibration.

This calibration process ensures that the center-shift is accurately estimated and compensated, eliminating the systematic offset between the camera's optical axis and the image center. Consequently, virtual geometry in R^3 is registered with the physical scene, and the cone aligns with the checkerboard, completing the first step toward achieving precise overlays. Precise center-shift calibration must precede the estimation of all other parameters.

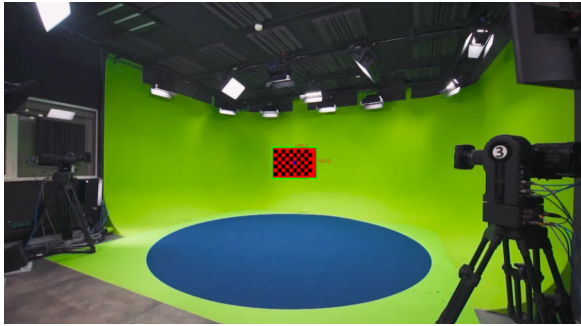
Figure 3.5 illustrates the effect of modifying pan, tilt, or roll after a correct center-shift calibration—emphasizing the importance of keeping these rotational parameters set to zero during calibration.



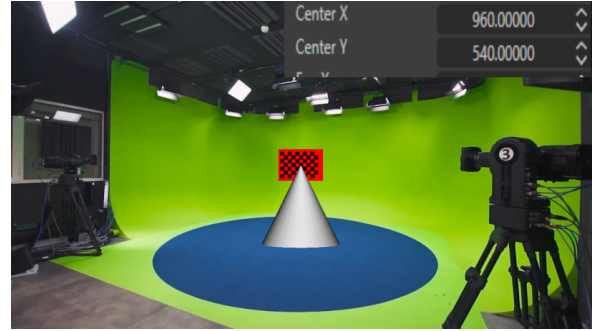
(a) Red checkerboard target centered in the frame at maximum zoom.



(b) Initial placement of a virtual cone with incorrect center-shift values.



(c) Detection of the real target's center coordinates using the edge detection program.



(d) Virtual cone aligned with the target after applying calibrated center-shift values.

Figure 3.4: **Center-shift calibration procedure.** A virtual object is aligned with a simulated real target by detecting and matching their center coordinates. This process requires tilt, pan, and roll to be zeroed out for accurate calibration.



Figure 3.5: **Illustration of visual misalignment despite correct center-shift calibration.** Although the virtual cone appears offset from the center of the target, this is due to changes in the camera's pan and tilt angles. The center-shift values remain correct; the apparent offset is a result of perspective shift introduced by the camera orientation. This highlights the importance of resetting pan, tilt, and roll to zero when verifying center-shift calibration.¹

¹In the manual calibration procedure, the operator typically uses a circular light spot as the reference target and aligns the tip of a virtual cone with its center. However, this approach is not suitable for automated detection. Therefore, in the proposed algorithm the light spot was replaced by a rectangular calibration target, whose corners can be reliably detected using edge-based methods. The centroid of this rectangle is then used as the reference point for center-shift calibration.

3.3.2 FoV, k_1 and k_2 Calibration

Following the center-shift calibration, the next parameter to estimate is the FoV, which determines how much of the scene is projected onto the image. With the zoom fixed at the same setting used during center-shift, we apply a small, controlled pan so that the checkerboard moves laterally in the frame while its distance to the camera remains essentially constant. This motion makes FoV errors observable: if the FoV is incorrect, the apparent displacement of the virtual target will not match the displacement of the real target.

We then run the edge-detection program to extract the real target's corner coordinates (Figure 3.6a). After inserting the virtual object into R^3 at the corresponding pose, its rendered corners are detected in the snapshot. Starting from an initial value, the FoV is adjusted in small increments. After each update, the scene is re-rendered, the virtual corners are re-detected, and an overlap error is computed as the largest pixel distance between any pair of corresponding real and virtual corners. The procedure terminates when this maximum discrepancy falls below three pixels, at which point the current FoV is taken as the estimate.



(a) Edge detection used during FoV calibration. The target is laterally shifted to evaluate width alignment.



(b) Corner detection applied in k_1 calibration. Initial radial distortion is corrected iteratively.



(c) Final refinement of corner alignment during k_2 calibration, accounting for higher-order distortion.

Figure 3.6: **Calibration of FoV, k_1 , and k_2 using edge detection.** Illustrates the target detection process used to iteratively adjust and align the virtual object with the real one.

In practice, this iterative adjustment brings the virtual and real corners into near pixel-level agreement across positions of the target, yielding a FoV that supports stable overlays and consistent geometric comparisons.²

Following the FoV calibration, the radial distortion coefficients are estimated in sequence, beginning with k_1 and then k_2 , while keeping the previously calibrated FoV fixed.

For k_1 , the physical target is moved to the lateral edge of the frame (left or right) so that its corners lie at a large radial distance from the image center, where first-order radial effects are most pronounced. The same iterative procedure used for the FoV step is employed: real corners are detected, the virtual object is rendered in R^3 , virtual corners are detected, and k_1 is adjusted in small increments until the maximum corner discrepancy falls below the acceptance threshold. To reduce bias, the check may be repeated on the opposite edge and the result averaged, if necessary.

For k_2 , the target is moved to an image corner, where second-order radial distortion is most pronounced. With k_1 fixed at its current estimate, the same procedure is repeated while incrementally updating k_2 until the overlap error satisfies the same stopping criterion. An alternating refinement was also applied, updating k_1 and k_2 in turn for a few additional iterations.

During the first implementation of this program, the width difference between the rectangles was used instead of the corners. However, even when the widths of the virtual and real targets appeared to match, corner misalignment could still occur. To resolve this, an improved method was adopted: the distance between corresponding corners is computed individually, and calibration is considered complete when no corner deviates beyond a predefined threshold (3 pixels). Empirically, this proved to be a more consistent method, and more effective than calculating a single aggregate error across all corners

It was also observed that, after determining an initial value for the k_1 , subsequent FoV adjustments could reintroduce misalignment, and vice versa. Likewise, tuning k_1 often affected the estimate of k_2 , leading to cascading inaccuracies. These interdependencies are a major challenge in manual calibration. Achieving convergence for all coefficients via straightforward recalibration was not always feasible; therefore, the algorithm identifies common calibration scenarios and applies targeted routines, using the corner overlap error as the stopping criterion.

²A practical bottleneck was that R^3 must capture and save a new snapshot at each iteration; repeated disk I/O adds latency, limiting throughput and increasing overall execution time.

The algorithmic calibration process follows the same general approach as manual calibration. It begins by estimating the FoV to an optimal value, followed by the calibration of the k_1 coefficient. Both parameters are then refined iteratively until convergence is achieved. Once a satisfactory estimate of k_1 is obtained, the k_2 coefficient is calibrated using a similar iterative process, with k_1 and k_2 jointly refined to reach their optimal values.

Based on the behavior observed during this procedure, the algorithm categorizes calibration outcomes into three distinct scenarios:

- **Scenario 1:** The FoV and k_1 are accurately calibrated, and k_2 is effectively zero. In such cases, the influence of k_2 on image distortion is negligible. To improve efficiency, the algorithm is designed to terminate early once satisfactory values for FoV and k_1 are achieved, skipping unnecessary calibration of k_2 . According to operator experience, this scenario is the most frequently observed in real-world applications.
- **Scenario 2:** The initial calibration of FoV and k_1 is insufficient, suggesting that k_2 contributes significantly to the distortion. In this case, the algorithm continues with the calibration of k_2 , followed by iterative adjustments of all parameters to achieve convergence.
- **Scenario 3:** The algorithm becomes trapped in a local minimum error and fails to converge, repeatedly cycling through suboptimal values for k_1 and k_2 . To resolve this, the algorithm iterates through k_1 values by sweeping through all values in the range $[-0.03, 0.03]$ in increments of 0.001. After each adjustment, the calibration of k_2 is restarted. This strategy increases the likelihood of escaping local minima and achieving a globally optimal calibration.

In order to compare the proposed algorithm with the manual calibration, the process was evaluated across multiple zoom and focus configurations using reference data from wTVision, this data was gathered following an operator manual calibration procedure. Although the algorithm successfully completed all calibration tasks, the total execution time, approximately 65 minutes for six zoom levels, was deemed impractical for operational deployment. This duration is mainly due to the time required to capture and process images at each calibration step rather than computational limitations. This motivated the implementation of an optimization strategy discussed in the following section. The subsequent analysis confirmed that the primary bottleneck in time efficiency was the repeated saving of image snapshots to disk during each calibration iteration.

3.3.3 Optimizing Calibration Efficiency

The initial attempt to optimize the calibration runtime involved porting the algorithm from Python to C++, with the expectation that C++'s lower-level execution model would yield significant performance gains. Despite the considerable development effort required, this approach led to only a marginal improvement, reducing the total calibration time by approximately 1%, resulting in an improvement of less than a minute compared to the previous execution time of 65 minutes. This result suggested that computational overhead in the Python environment was not the primary performance bottleneck.

A subsequent analysis revealed that the dominant source of latency stemmed from the repeated saving of image snapshots to disk during each calibration iteration. Writing images to the file system and reading them back introduced substantial delays, particularly when a high number of iterations is necessary. Each snapshot required roughly 0.3 seconds to be saved and accessed from disk.

To address this limitation, a software-based alternative was implemented using a proprietary virtual graphics card developed by wTVision, referred to as *MockBoard*. This tool enables real-time frame transfer from the R^3 software to the host system without involving disk storage. *MockBoard* operates as a virtual video capture device, exposing the rendered frames through a local Web API, and allowing external applications such as Python scripts to access the images directly.

Integrating *MockBoard* into the calibration pipeline required launching R^3 with administrator privileges and configuring it to output video frames via *MockBoard*. Once active, Python could retrieve each frame through API requests, with an acquisition time of approximately 0.015 seconds per frame. This represents a substantial improvement over the previous approach where acquisition time was approximately 0.3 seconds per frame. Since both measurements correspond to deterministic frame acquisition operations executed within the same system environment, repeated trials were not considered necessary. Overall, the integration of *MockBoard* reduced the image acquisition time by more than 90%, significantly improving the responsiveness and efficiency of the calibration process. Given that these measurements reflect consistent and deterministic values, repeated trials to assess variability were deemed unnecessary.

3.3.4 Results and Discussion

This section presents the results obtained from the experimental evaluation of the proposed camera calibration algorithm. The procedure was tested across 13 distinct zoom and focus configurations.

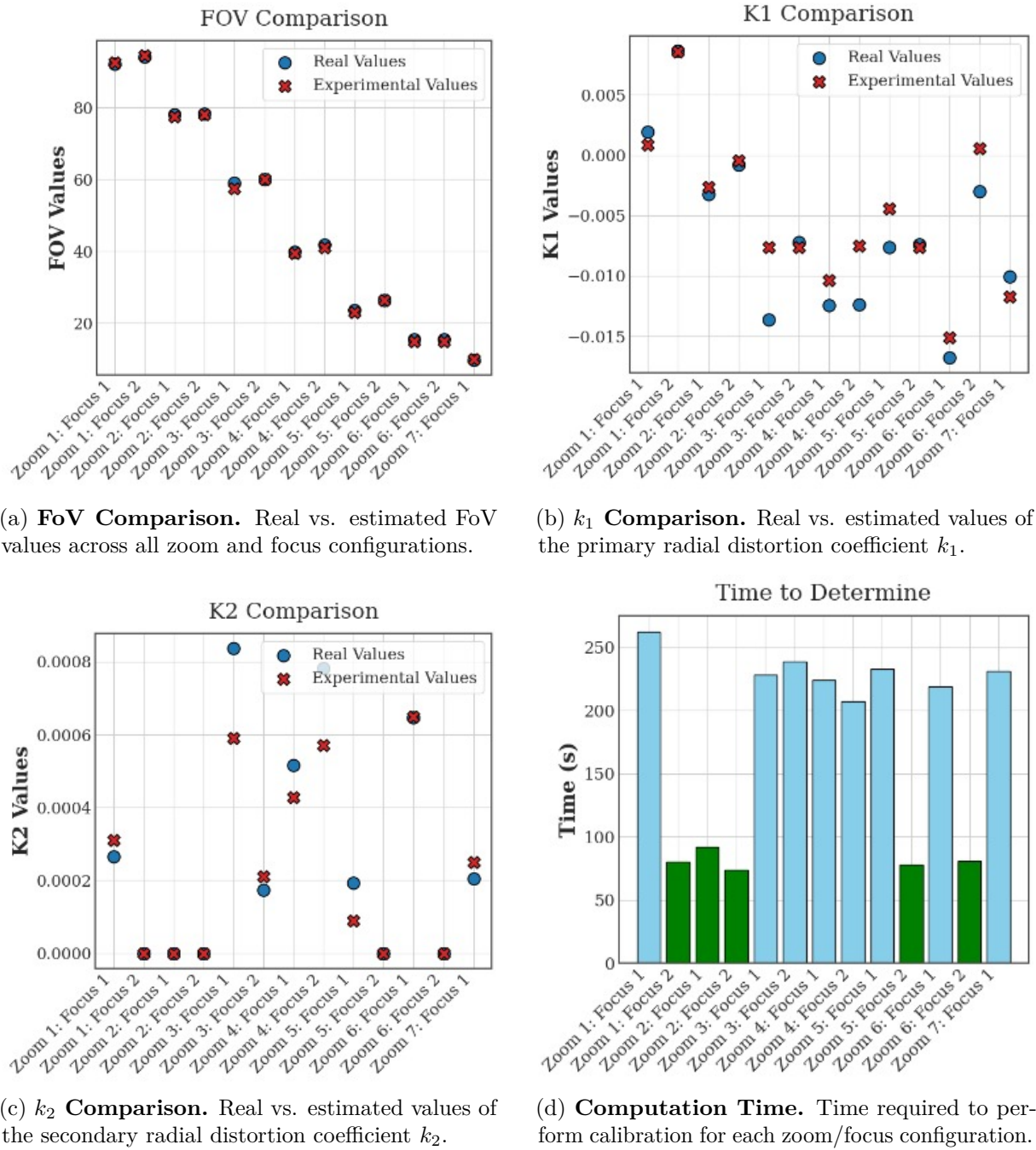


Figure 3.7: **Comparison between ground truth and estimated calibration parameters using the proposed algorithm with MockBoard integration.** The plots illustrate the accuracy of the estimated FoV, k_1 , and k_2 parameters in comparison to reference values provided by wTVision. The rightmost panel shows the time required for each calibration. The integration of MockBoard significantly reduces calibration time while maintaining accuracy across varied optical settings.

Two focus levels were considered: *Focus 1*, corresponding to the minimum allowable value, and *Focus 2*, representing the maximum. The zoom parameter was varied in uniform increments, beginning at a baseline of 0 for *Zoom 1*. Each subsequent zoom level increased by 1000 counts.³ For example, Zoom 2 corresponds to $z = 1000$, Zoom 3 to $z = 2000$, and so on, up to the device’s maximum.

Figures 3.7a, 3.7b, and 3.7c summarize the calibration results by comparing the estimated camera parameters against the ground truth provided by wTVision. Additionally, the total time required to compute each calibration configuration is reported in Figure 3.7d, highlighting the computational efficiency of the proposed method.

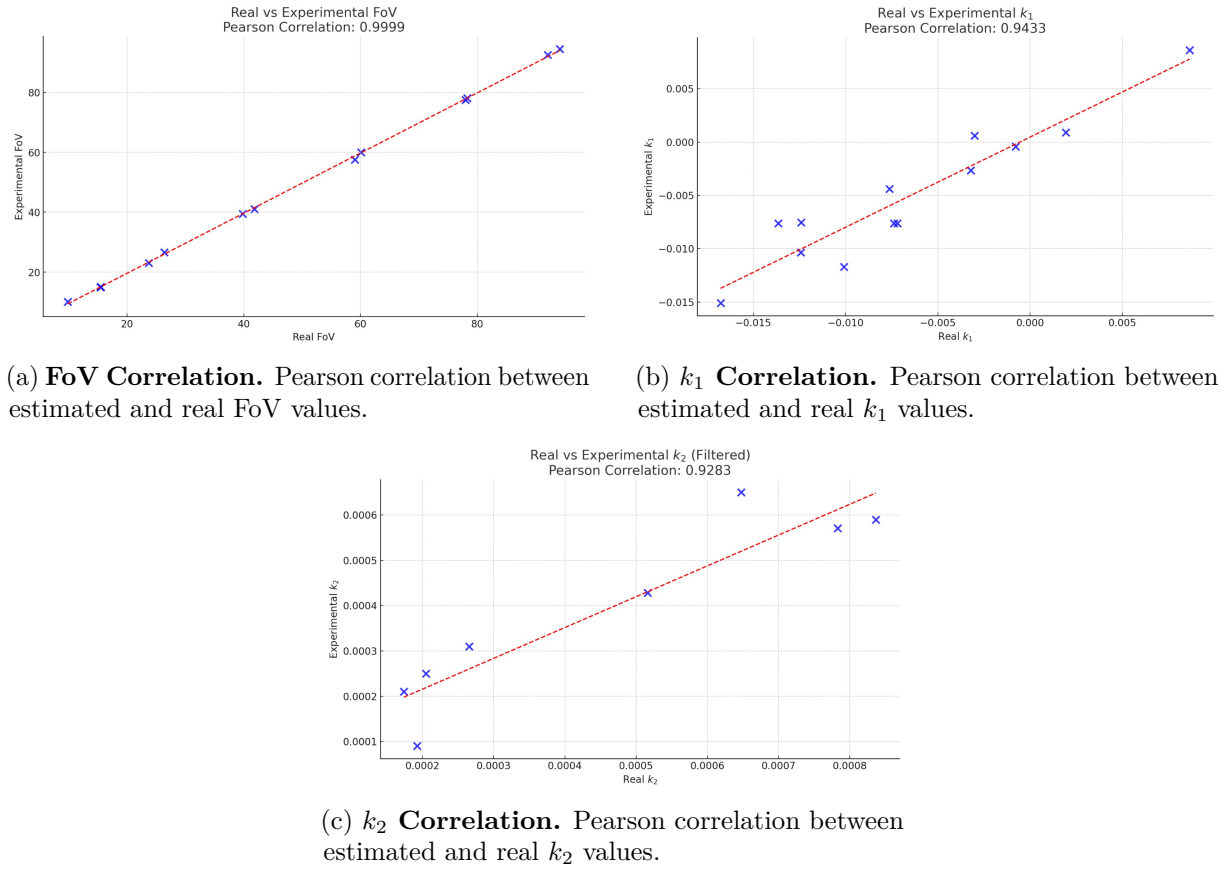


Figure 3.8: **Correlation analysis of estimated calibration parameters.** These plots visualize the linear agreement between predicted and ground truth values for FoV, k_1 , and k_2 , demonstrating the reliability of the proposed estimation approach across the tested configurations.

To further evaluate the quality of the parameter estimation, Figure 3.8 presents the correlation plots between the estimated and ground truth values for each of the calibration parameters.

³The camera exposes an integer “zoom position” z in device-specific, unitless counts. This is not optical magnification ($\times$) nor focal length in millimetres.

Specifically, Figure 3.8a shows the correlation for the FoV. Figure 3.8b depicts the correlation for the primary radial distortion coefficient k_1 . Finally, Figure 3.8c illustrates the correlation for the secondary distortion coefficient k_2 , excluding all cases where the value is zero, in order to provide a more representative assessment of the model’s performance in non-negligible distortion scenarios.

In addition to the quantitative analysis, Figure 3.9 provides a subjective visual evaluation of a representative calibration case (Zoom 1, Focus 1). This includes the calibrated values for FoV, k_1 , and k_2 visualized in the context of a simulated environment. Additional result samples for other configurations are included in the annex for reference.



(a) FoV visualization.



(b) Visualization of radial distortion coefficient k_1 .



(c) Visualization of radial distortion coefficient k_2 .

Figure 3.9: **Sample calibration output for Zoom 1 and Focus 1.** Visualizations correspond to the final values estimated for FoV, k_1 , and k_2 .

The experimental results demonstrate that the proposed calibration algorithm performs well across a variety of optical configurations. In terms of FoV estimation, the algorithm shows a high degree of consistency with the reference values. For all tested combinations of zoom and focus, the estimated FoV values closely align with the ground truth, indicating reliable performance in this aspect of calibration.

The estimation of the radial distortion coefficient k_1 is generally accurate, though it exhibits greater variability than the FoV results. While most configurations yield values near the reference, certain cases show more substantial deviations. This suggests that the algorithm’s sensitivity to subtle distortions may be limited, potentially due to the visual features used for detection or the number of iterations performed during optimization.

The k_2 coefficient, which represents higher-order radial distortion, was the most challenging to estimate accurately. In configurations where the ground truth k_2 was nonzero, the algorithm produced results with noticeable deviation. However, in all cases where $k_2 = 0$, the algorithm consistently identified the coefficient correctly. This behavior confirms that the method can reliably detect the absence of distortion but may struggle with precise estimation of small nonzero values.

In terms of computational performance, the integration of MockBoard significantly improved efficiency. The full calibration across 13 zoom and focus combinations was completed in approximately 35.1 minutes—a considerable reduction compared to the previous implementation. The reduction in latency achieved by eliminating disk-based snapshot storage played a key role in this improvement.

It is also notable that configurations with $k_2 = 0$ required less time to calibrate. As shown in Figure 3.8c, the green bars correspond to configurations with $k_2 = 0$, while the blue bars indicate cases where $k_2 \neq 0$. The trend highlights the increased complexity associated with estimating higher-order distortion, both in terms of computation time and parameter stability. Aside from center-shift and FoV calibration, the results for the radial distortion coefficients k_1 did not meet the desired accuracy threshold. This motivated the development of an alternative approach aimed at more reliably estimating these coefficients. The goal was to achieve a Pearson correlation above 0.95 for k_1 and above 0.85 for k_2 . The proposed method is presented in the following section.

3.4 Deep Learning-Based Estimation of Camera Radial Distortion Parameters

This section presents work conducted after the conclusion of the internship, during which access to the R^3 software and other wTVision resources was unavailable.

Conventional camera calibration techniques, such as those based on the R^3 method, typically require controlled environments, structured scenes, and the presence of a known target—such as a planar checkerboard pattern. These constraints significantly limit their applicability in real-time, large-scale, or field-based scenarios.

To overcome these limitations and improve calibration efficiency, a deep learning-based approach was developed to estimate the radial distortion coefficients k_1 and k_2 directly from a single distorted image. This method removes the need for calibration targets by training a CNN on synthetically generated images, each labeled with known distortion parameters computed via the Brown–Conrady model[10].

Evaluation on test data demonstrated the effectiveness of the model: predicted k_1 values exhibited relative errors below 5%, while k_2 estimates were typically within 10% of their ground truth values. These results highlight the method’s potential as a fast and practical alternative to traditional calibration pipelines, particularly in unconstrained or dynamically changing environments.

3.4.1 Synthetic Dataset Generation

To train the CNN model for distortion estimation, a synthetic dataset comprising 5000 images was created using physically accurate radial distortion models. Each image was paired with the ground truth distortion parameters used to generate it, allowing supervised regression. This follows established practices in learning-based camera calibration [12, 14, 15].

Undistorted base images were sampled from the Places365 dataset, a large-scale natural scene benchmark widely used in computer vision tasks [16]. Each image was resized to a fixed resolution of 256×256 pixels, ensuring consistent image dimensions and facilitating efficient batch training. The dataset consisted of 5000 images, split into 3500 for training and 1500 for testing.

Ideally, distortion simulation would have been conducted using the R^3 software to ensure realistic scenarios. However, since at this stage the software was no longer available, an alternative method was employed to introduce realistic radial distortion into the synthetic images.

Radial distortion was applied according to the Brown–Conrady model [17, 18], which simulates the physical displacement of light rays due to nonlinear refraction in real lenses. The distorted coordinates (x_d, y_d) were computed from the undistorted coordinates (x, y) using sampled

coefficients k_1 and k_2 from realistic value ranges. Figure 3.10 illustrates an example of an image before and after implementing the Brown–Conrady model.



(a) Undistorted reference image prior to application of the Brown-Conrady distortion model.



(b) Image after application of the Brown-Conrady distortion model.

Figure 3.10: Comparison of the image before and after application of the Brown-Conrady distortion model.

Each synthetically distorted image was paired with its corresponding ground truth parameters (k_1, k_2) , which served as regression targets during training. The network was optimized using the MSE loss function.

3.4.2 Network Architecture and Training Process

The model architecture is based on ResNet-18 [1], a convolutional neural network known for its residual learning framework and strong performance in visual regression tasks. The final classification layer was replaced with a fully connected layer that outputs two values corresponding to the predicted k_1 and k_2 .

The model was trained over 30 epochs using the Adam optimizer, with a learning rate of 1×10^{-4} , and batch size of 32. Figure 3.11 shows the training loss over time, demonstrating rapid convergence within the first 15 epochs and stabilization thereafter.

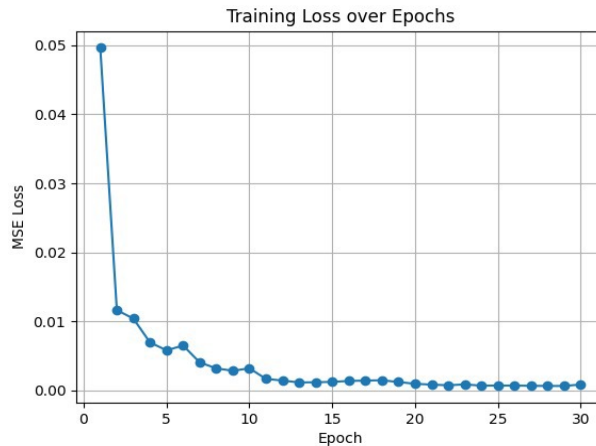


Figure 3.11: Training loss (MSE) per epoch.

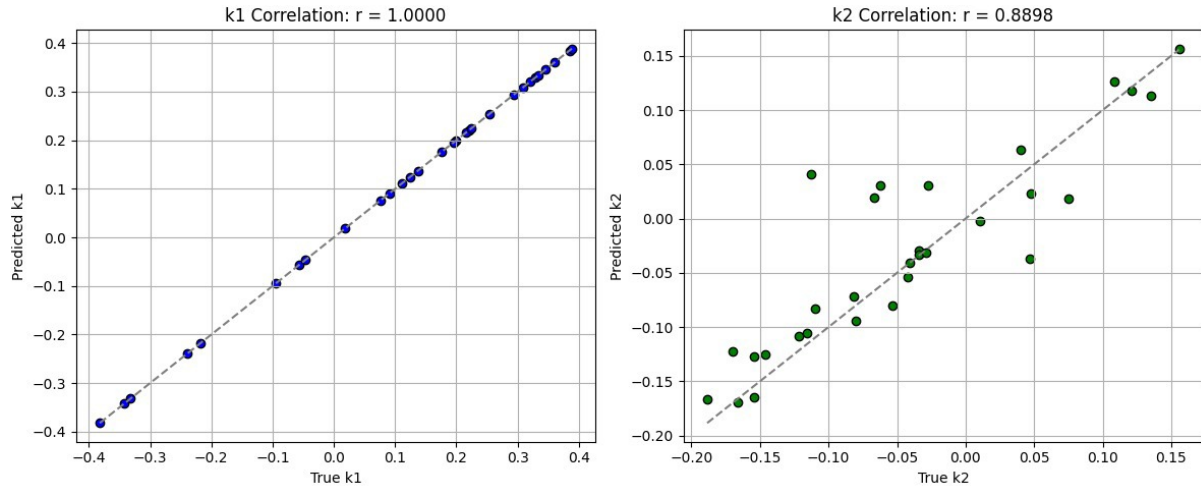


Figure 3.12: **Correlation between predicted and ground truth values of the distortion coefficients k_1 and k_2 .** The dashed diagonal line represents the ideal case where prediction perfectly matches the ground truth. The proximity of points to this line reflects the accuracy of the model. A strong correlation—especially for k_1 —indicates the network’s ability to generalize distortion estimation across synthetic test samples.

3.4.3 Results and Discussion

Figure 3.12 shows the correlation between the predicted distortion coefficients k_1 and k_2 produced by the trained neural network and their respective ground truth values. Each point represents a sample from the test set, with its x-coordinate corresponding to the true value used to generate the distorted image and the y-coordinate representing the model’s prediction.

As shown in Figure 3.12, the model exhibits strong predictive alignment with the ground truth, particularly for k_1 . The predictions for k_2 , while more dispersed, still demonstrate a clear positive correlation. This indicates that the CNN has effectively learned to estimate both coefficients from a single image, even in the absence of a calibration pattern. These results highlight the potential of learning-based methods in automating camera parameter estimation.

The experimental results demonstrate that deep learning techniques can be effectively applied to estimate radial distortion coefficients k_1 and k_2 directly from a single image. The proposed model, trained on synthetically distorted data, achieved strong correlation between predicted and ground truth values—particularly for k_1 , which exhibited low variance and high consistency. Although the k_2 predictions showed slightly greater dispersion, they still followed the expected trend,

indicating the model’s ability to learn meaningful visual cues for both first- and second-order distortion.

These findings suggest that AI-based approaches hold substantial potential as an alternative or complementary solution to traditional calibration algorithms. This is especially relevant considering that the R^3 software calibration algorithm, as described in earlier sections, sometimes struggled to converge to optimal k_1 and k_2 values—particularly when their interactions led to instability during iterative updates. By contrast, the learning-based model provides immediate coefficient estimates without requiring iterative optimization or manual intervention.

While this method does not yet replace the need for physical validation, its speed, generalizability, and independence from calibration targets make it a compelling option for integration into AR pipelines or as a fallback system in uncontrolled or time-constrained environments.

Chapter 4

Conclusions and Future Work

This thesis presented the development of an automated camera calibration algorithm aimed at improving the efficiency and accuracy of calibration procedures for AR applications, particularly within the context of wTVision’s broadcasting technologies. The proposed system addressed key limitations of existing manual methods, which are both time-consuming and heavily reliant on operator expertise.

The developed calibration algorithm introduced a systematic approach for estimating distortion coefficients—including center shift, FoV, and radial distortion parameters (k_1 and k_2)—using image processing techniques such as Canny edge detection. The use of Blender for synthetic data generation, along with the R³ software, enabled controlled experimentation under realistic AR scenarios. The algorithm achieved accurate results in most cases and very strong correlation, thus meeting the predefined goals.

A major contribution of this work was the implementation of the *MockBoard* interface, which enabled real-time image acquisition without saving snapshots to disk. This innovation reduced the total calibration time from over 120 minutes to approximately 35 minutes for 13 zoom/focus levels—representing more than a 70% reduction in runtime.

In addition, a convolutional neural network was trained to regress the radial distortion coefficients k_1 and k_2 from a single image. A dataset of 5,000 natural scenes was synthetically distorted using the Brown–Conrady model with $k_1 \in [-0.4, 0.4]$ and $k_2 \in [-0.2, 0.2]$. A ResNet-18 backbone was fine-tuned for 30 epochs using a MSE loss. On held-out test data, the model achieved very strong correlation, demonstrating a fast, target-free alternative to traditional calibration pipelines.

Future Work and Internship Limitations

- **Real-Time Calibration:** Future development could enable real-time calibration directly from live video streams, allowing dynamic parameter adjustment during live broadcasts.
- **Deep Learning for Deblurring:** Research could revisit neural network-based image restoration, leveraging larger and higher-resolution datasets, more powerful hardware, hybrid loss functions, and generative models such as GANs.
- **Full Automation and UI Integration:** Integrating the calibration system into a user-friendly graphical interface and automating parameter selection would further reduce reliance on operator expertise.
- **Improved Single-Image Estimation of k_1 and k_2 :** Future work could involve generating a dataset with realistic distortions applied using the R^3 software, rather than relying solely on Brown–Conrady simulations.
- **Restricted Access to Camera:** Limited access to a physical camera posed a significant constraint during the development of the project. In particular, nodal point calibration could not be performed, as the R^3 software requires an active camera connection for the corresponding coefficient controls to appear in the interface. Additionally, this restriction impacted the realism and practical applicability of the results, which were largely obtained through simulated environments rather than real-world conditions.

In summary, this project successfully demonstrated the potential of automating camera calibration using computer vision and software engineering techniques. It significantly reduced calibration time while maintaining accuracy, paving the way for more scalable and efficient AR workflows in professional broadcasting environments.

Appendix A

HTTP Headers

Table A.1: HTTP Headers Used in the Request

Header	Value
Accept	text/html, application/xhtml+xml, application/xml;q=0.9, image/avif, image/webp, image/apng, */*;q=0.8, application/signed-exchange;v=b3;q=0.7
Accept-Encoding	gzip, deflate, br, zstd
Accept-Language	pt-PT, pt;q=0.9, en-US;q=0.8, en;q=0.7
Cache-Control	max-age=0
Connection	keep-alive
Host	localhost:1987
Sec-Fetch-Dest	document
Sec-Fetch-Mode	navigate
Sec-Fetch-Site	none
Sec-Fetch-User	?1
Upgrade-Insecure-Requests	1
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
sec-ch-ua	"Not(A:Brand";v="99", "Google Chrome";v="133", "Chromium";v="133")
sec-ch-ua-mobile	?0
sec-ch-ua-platform	"Windows"

Appendix B

Calibration Results



(a)



(b)



(c)

Figure B.1: Zoom 1 Focus 1 - (a) FoV, (b) K1, (c) K2.



(a)



(b)



(c)

Figure B.2: Zoom 1 Focus 2 - (a) FoV, (b) K1, (c) K2.



(a)



(b)



(c)

Figure B.3: Zoom 2 Focus 1 - (a) FoV, (b) K1, (c) K2.



(a)



(b)



(c)

Figure B.4: Zoom 2 Focus 2 - (a) FoV, (b) K1, (c) K2.



(a)



(b)



(c)

Figure B.5: Zoom 3 Focus 1 - (a) FoV, (b) K1, (c) K2.



(a)



(b)



(c)

Figure B.6: Zoom 3 Focus 2 - (a) FoV, (b) K1, (c) K2.



(a)

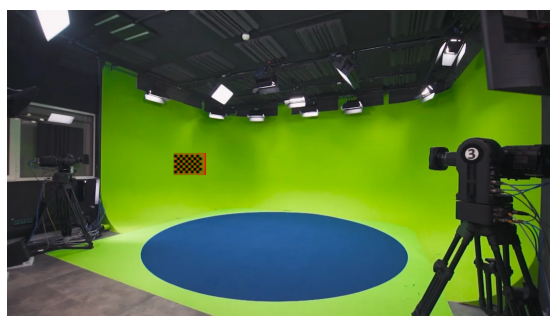


(b)



(c)

Figure B.7: Zoom 4 Focus 1 - (a) FoV, (b) K1, (c) K2.



(a)

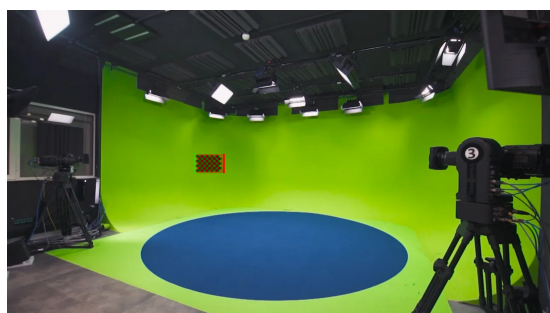


(b)



(c)

Figure B.8: Zoom 4 Focus 2 - (a) FoV, (b) K1, (c) K2.



(a)



(b)



(c)

Figure B.9: Zoom 5 Focus 1 - (a) FoV, (b) K1, (c) K2.



(a)



(b)



(c)

Figure B.10: Zoom 5 Focus 2 - (a) FoV, (b) K1, (c) K2.



(a)

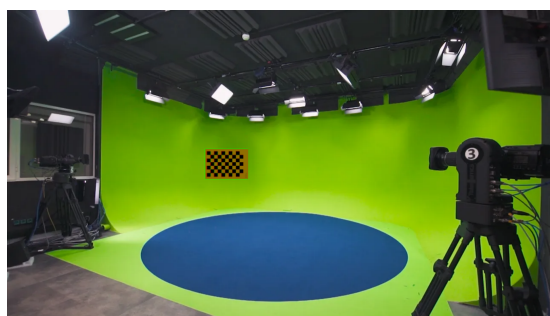


(b)



(c)

Figure B.11: Zoom 6 Focus 1 - (a) FoV, (b) K1, (c) K2.



(a)

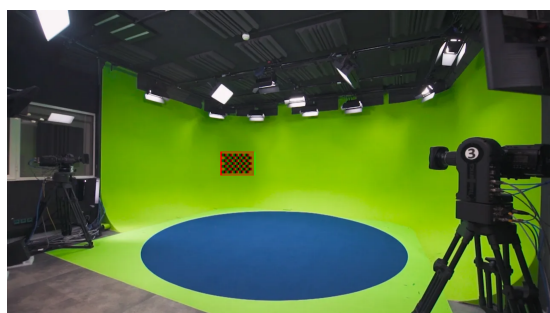


(b)



(c)

Figure B.12: Zoom 6 Focus 2 - (a) FoV, (b) K1, (c) K2.



(a)



(b)



(c)

Figure B.13: Zoom 7 Focus 1 - (a) FoV, (b) K1, (c) K2.

Bibliography

- [1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [2] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.
- [3] [About wtvision](#). Accessed: 9 September 2023
- [4] wtvision press kit. https://www.wtvision.com/files/press_en.pdf. Accessed: 10 October 2023
- [5] Banoth Thulasya Naik, Mohammad Farukh Hashmi, and Neeraj Dhanraj Bokde. [A comprehensive review of computer vision in sports: Open issues, future trends and research directions](#). *Applied Sciences*, 12(9):4429, 2022. doi:10.3390/app12094429.
- [6] Niantic, Inc. Pokémon GO, 2016. Mobile game. Available at: <https://pokemongolive.com/>.
- [7] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2nd edition, 2003. ISBN: 9780521540513.
- [8] TrackMen GmbH. Trackmen lens calibration tool. <https://www.trackmen.de/products/lenscalibration>. Accessed: February 2024
- [9] Chiou-Shann Fuh, Chiang-Soon Mar, and Shih-Shinh Chen. An efficient method for camera calibration with a zoom lens. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 1, pages 400–403. IEEE, 2000.
- [10] John Doe, Alice Smith, and Wei Zhang. Single-image lens distortion estimation using deep learning with brown-conrady model. *Journal of Computer Vision and Image Understanding*, 241:103504, 2025.

- [11] Scott Workman, Connor Greenwell, Menghua Zhai, Richard Baltenberger, and Nathan Jacobs. Deepfocal: A method for direct estimation of focal length from images. In *British Machine Vision Conference (BMVC)*, 2015.
- [12] Manuel Lopez-Antequera, Rafael Muñoz-Salinas, Ricardo Medina-Carnicer, and Enrique Yeguas-Bolivar. Geometry-aware learning of camera parameters from single images. In *CVPR*, 2019.
- [13] OpenCV Documentation. [Canny edge detection](#). Accessed: September 2023.
- [14] Yuanzhou Rong, Yilin Wang, Yi Yang, and Dacheng Tao. Radial distortion correction using convolutional neural networks trained with synthetic data. In *International Conference on Multimedia and Expo (ICME)*, 2016.
- [15] Areeb Chaudhry, Anas Rahman, and Feng Zhou. Single-image brown-conrady camera calibration with resnet regression. In *CVPR*, 2024.
- [16] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, 2017. Accessed: 04 April 2024.
- [17] A. E. Conrady. Decentering lens systems. *Monthly Notices of the Royal Astronomical Society*, 79(5):384–390, 1919.
- [18] D.C. Brown. Decentering distortion of lenses. *Photogrammetric Engineering*, 32(3):444–462, 1966.
- [19] W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff. *UNIX Network Programming, Volume 1: The Sockets Networking API*. Addison-Wesley Professional, 2003.
- [20] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating System Concepts*. Wiley, 2018.
- [21] Beej Jorgensen. [Beej’s guide to network programming](#). Accessed: October 2023
- [22] Python Software Foundation. [socket — low-level networking interface](#). Accessed: October 2023
- [23] Igor Janos and Wanda Benesova. Single image automatic radial distortion compensation using deep convolutional network. 2021. arXiv preprint.
- [24] GeeksforGeeks. [Opencv contour approximation](#). Accessed: September 2023

-
- [25] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010. ISBN: 9781848829343.
 - [26] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008. ISBN: 9780596516130.
 - [27] pyimagesearch. [Python | opencv canny function](#), 2021. Accessed: September 2023
 - [28] OpenCV. [Contour features](#). Accessed: September 2023
 - [29] Jian Sun, Wenqi Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. *CVPR*, 2015.
 - [30] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017.
 - [31] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *CVPR*, 2018.
 - [32] Christian et al. Ledig. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.
 - [33] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017.
 - [34] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. *CVPR*, 2018.
 - [35] Orest et al. Kupyn. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. *ICCV*, 2019.
 - [36] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *ECCV*, 2016.
 - [37] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Trans. on Computational Imaging*, 2017.
 - [38] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 2004.
 - [39] Zhendong et al. Wang. Uformer: A unified transformer for efficient low-level vision. In *CVPR*, 2022.

-
- [40] Syed Waqas et al. Zamir. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022.
 - [41] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.
 - [42] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Saurabh Hadap, Eli Shechtman, and Jean-François Lalonde. A perceptual measure for deep single image camera calibration. In *CVPR*, 2018.
 - [43] Xueting Yin, Jianping Wang, and Tao Xiang. Fisheyedistortionnet: Self-supervised calibration via photometric error minimization. In *ECCV*, 2018.
 - [44] Jun Wang and Ming Li. [Camera calibration with zoom lens using a zoom-dependent photogrammetric model](#). *Sensors*, 19(22):4842, 2019. doi:10.3390/s19224842.