SoftwareX 31 (2025) 102220

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

Automated camera calibration (ACC): A python GUI application for automatically calculating camera intrinsics with Arri camera systems

Nicholas D. Matthews, Michael D. Holm, Anjali Gali, Alex R. Renner, Eliot Winer *

1620 Howe Hall, 537 Bissell Road, Ames, IA 50011-2274, USA

ARTICLE INFO

Keywords:
Camera calibration
Virtual production
Computer vision
Multi-threaded Development
Automation

ABSTRACT

Calibrating camera intrinsics to produce a camera model is an essential, time-consuming and labor-intensive process in virtual film production. Personnel with technical expertise and experience must make manual adjustments to lens configurations and perform repetitive image detection procedures. Since calibration needs to be performed multiple times across different system setups within tight time constraints, automating the process would reduce the time and cost of virtual film production. The Automated Camera Calibration (ACC) software was developed to automate lens adjustments on the most widely used camera system in film product, Arri, through motor-enabled camera components and conduct real-time image detection. Validation methods showed that ACC can complete camera calibration over three times faster than the popular Unreal Engine camera calibration method.

Metadata

Nr	Code metadata description	*Please fill in this column*
C1	Current code version	v1.4.0
C2	Permanent link to code/repository used for this code version	https://github.com/nickdmakes/camera-calibration
C3	Permanent link to reproducible capsule	
C4	Legal code license	MIT License
C5	Code versioning system used	git
C6	Software code languages, tools and services used	Python
C7	Compilation requirements, operating environments, and dependencies	Python 3.9.1, PyQt6
C8	If available, link to developer documentation/manual	
C9	Support email for questions	ewiner@iastate.edu

1. Motivation and significance

The film industry is shifting from traditional filmmaking methods to virtual production, driven by the significant reduction in cost and increase in production efficiency. Newer film technology allows virtual production studios to reduce the number of on-location shoots by digitally replicating visual aspects of a scene in a controlled environment. In 2023, the global market valued virtual production at US $2.9 Billion and the market is projected to reach US $6.8 Billion by 2030 [1]. During virtual film production a virtual camera mimics the behavior of a physical camera, to seamlessly integrate real-world elements with virtual environments displayed on an LED wall. The LED wall is a projection of the virtual camera that displays real-time virtual scenery, visual effects, and 3D elements that can respond to changes from the physical camera. However, the unique characteristics of a physical camera body and lens make it challenging to replicate its properties virtually. These unique characteristics include: 1) principal point, which represents the projected center of the image plane and 2) lens distortion, which warps a projected image due to the curved nature of modern lenses [1–3]. The camera calibration process generates two sets of precise numerical values that represent these important camera parameters. Extrinsic parameters represent the camera's orientation and position in real-world space. Intrinsic parameters describe how the camera's internal geometry projects light onto the camera sensor [4]. Once generated, these parameters are combined to form a mathematical representation or camera model. A camera model enables virtual production software to project images exactly as they appear through the lens of a physical camera [5]. The most prominently used cameras in film production, for over a hundred years, are made by Arri and precise calibration is essential for realizing the scene compositions desired by production personnel.

Unreal Engine is widely used software for virtual film production

* Corresponding author.
E-mail addresses: nickd@iastate.edu (N.D. Matthews), agali1@iastate.edu (A. Gali), arenner@iastate.edu (A.R. Renner), ewiner@iastate.edu (E. Winer).

https://doi.org/10.1016/j.softx.2025.102220
Received 13 December 2024; Received in revised form 22 May 2025; Accepted 26 May 2025
Available online 3 June 2025

since it can project virtual environments onto an LED wall [6]. The projection is done through a virtual camera within Unreal Engine that converts the 3D virtual environment to a 2D image using a camera model. Unreal Engine uses the Brown-Conrady model that includes focal length, principal point, and lens distortion to perform image correction for accurate image projection [7]. Extrinsic parameters such as the camera's orientation and translation in 3D space are not included in Unreal Engine's camera model since cameras used in virtual production systems are dynamically tracked in real-time. Unreal Engine also provides camera calibration tools for calculating intrinsic parameters for the Brown-Conrady model.

Camera calibration methods provided by software like Unreal Engine are very meticulous. The process typically involves capturing several images of calibration patterns such as a checkerboard. An operator holds the checkerboard at distinct positions within the camera's field of view while simultaneously capturing the image to cover the lens's optical range. Additionally, the image capture process must account for varying focal distances since lenses are adjusted during film production and calibration parameters are different at each focal setting. This requires repeating image capture at multiple focal distances to cover the focus range used by the camera during production. Calibrating a camera at 10 different focal distances requires manually pressing a capture button for at least 100 images and manually changing the lens settings 10 times for accurate results.

Decreasing the number of days to shoot a film is one of the most significant factors in reducing overall production cost. However, calibrating cameras is a time-demanding process that requires multiple operators to manually capture precise measurements [8]. The accuracy of the calibration directly impacts the quality of the captured footage. Inaccuracies can create misalignment and unwanted distortion, degrading overall realism. Consequently, these errors lead to costly reshoots that drive up film expenses. Additionally, misalignments are caused by swapping camera components or natural wear on the camera system, requiring periodic recalibration that can significantly increase production costs due to time lost from this slow, but necessary process [9]. A software application that interacts with the camera's hardware has the potential to reduce camera calibration time and subsequently reduce costs. This paper presents the development of the Auto Camera Calibration (ACC) software tool. ACC is an intuitive application that drastically decreases the time to perform camera calibration by automating camera lens movement using motor-enabled components and conducting guided image capture with object detection. The resulting camera model can then be imported into Unreal Engine to align the virtual camera with the physical one.

2. Software description

ACC is written in Python with the PyQt6 UI framework, enabling cross platform functionality for the Windows and MacOS operating systems. Quality attributes pertaining to UI/UX were adhered to while creating the UI [10]. ACC is built for calibrating camera's that utilize Arri hardware [11] as described in Section 2.1. Currently, Arri is the most widely used brand for film production [12]. Therefore, an Arri camera system was used for testing ACC functionality.

2.1. Software architecture

Traditional camera calibration on an Arri camera system takes advantage of several camera hardware components such as Arri CForce lens motors. These motors control critical lens functions on the camera, such as focus, iris, and zoom. Alongside the CForce lens motors is the Arri UMC-4, a networking device that streams current lens configuration data (i.e., focus, iris, and zoom) in JSON format out from its ethernet network port. Socket programming was used to establish a connection to this stream and deserialize the data. The UMC-4 also provides a means for manually controlling the lens motors remotely over the network. A proprietary protocol called LBUS is sent from the UMC-4 to the lens motors to control the lens positions. Existing camera calibration tools require users to perform several manual lens adjustments by hand or using a remote control. However, manual lens adjustment is time-consuming and susceptible to human error. Another key component of camera calibration tools is a live feed of the camera view for collecting image data. Arri camera systems include a MON OUT port on the camera for streaming RAW format video over an SDI cable. This video feed is used to align the physical calibration pattern in the camera's field of view. Users then manually click a user interface to capture an image from the video feed. Even when completing this process with multiple operators, aligning the checkerboard pattern and capturing an image is difficult to coordinate and time-consuming.

ACC mitigates these issues through automating the lens motors and using image processing to automatically capture the calibration pattern without manual intervention. To accomplish lens motor automation, ACC requires another hardware component called the Arri LCUBE CUB-1, a small hardware adapter that converts public RS232 serial protocol to Arri's proprietary data protocol. The camera control system company CMotion, provided the protocol specification for controlling their CForce motors via RS232. A driver was developed for sending and receiving serial data packets using the protocol specification. After completing the lens encoder mapping, as outlined in Section 2.2, ACC can programmatically control the Arri CForce lens motors, eliminating all manual lens adjustment during calibration. For image capture, ACC uses FFmpeg to convert incoming RAW video to BGR 8-pixel format and forward the video to display interfaces of the host system. DeckLink and Windows dshow are the two display interfaces supported by ACC [13]. To automate image capture, two features are added: 1) Detection and capture of a checkerboard target using computer vision techniques and 2) User interface feedback that guides a user through the image capture process. These features allow one operator to complete image capture efficiently. More details on ACC's image capture method are discussed in Section 2.2.

While ACC automates many aspects of the calibration process, it assumes that lens motors do not experience significant drift and that the lens is securely seated and properly aligned. Although occasional misalignment may occur, the system is designed to remain functional. In future versions, additional error-checking features could be implemented to detect motor drift or lens shift in real time.

Hardware connections enable ACC to speed up camera calibration. However, the connections introduce challenges regarding asynchronous data, reliability, and fault tolerance. All three connections are interdependent during calibration runtime and rely on the most recent data output from each hardware component. For example, lens metadata coming from the Arri UMC-4 is often compared against the status packets from the CForce motors to validate motor movement. Latency from concurrent data access to both components would result in comparing data against outdated states. To mitigate issues regarding asynchronous data, a context manager was developed for each hardware connection. Each context manager runs on a separate thread, enabling ACC to offload work from its main thread and receive all three streams of data at the same time. In addition, each context manager maintains the state of the hardware connection (open, closed, sending data, receiving data). UI components can listen to changes made to the connection state and update accordingly. Lastly, the context manager improves fault tolerance by forwarding any connection state changes to the rest of the application with PyQt6 event signals. If one connection is lost during calibration, the application does not crash, and important data is automatically saved.

Fig. 1 is a component diagram of the ACC system that shows the camera hardware, user interface, and calibration logic components.

The diagram in Fig. 1 represents the structure of ACC, detailing the connections between camera hardware, the PyQt6 GUI application, and camera calibration logic. Hardware components (Arri LCUBE CUB-1, Arri UMC-4, and MON OUT) are shown interfacing with their

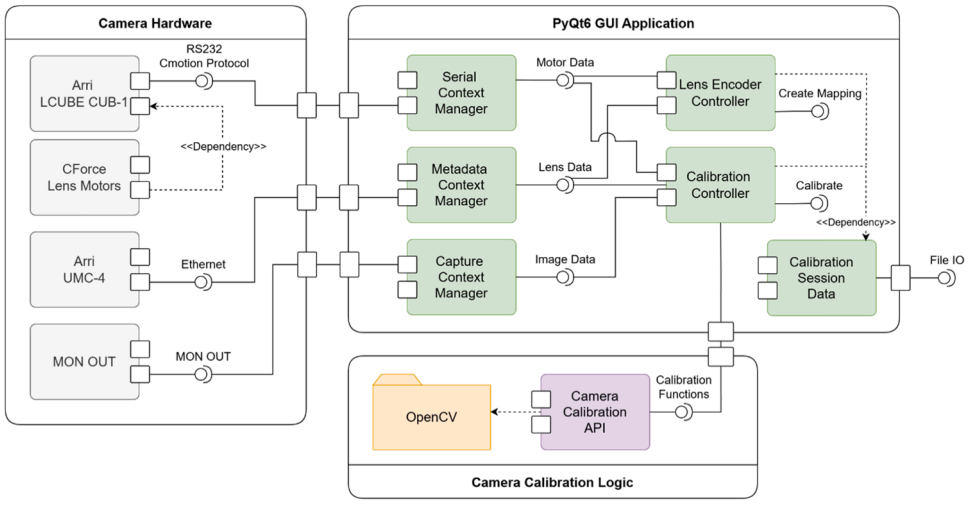


Fig. 1. ACC component diagram.

respective context managers. The PyQt6 GUI application has many UI elements that change based on user events and data read from these context managers. Namely motor data, lens data, and image data from Serial, Metadata, and Capture Context Managers as shown in Fig. 1. As more functionality was added, the complexity of interaction between elements increased drastically. Therefore, modularity was implemented by grouping elements with similar functions into controllers. Simple controllers are responsible for updating the UI or sending signals based on events and user input. More complex controllers, like the Lens Encoder Controller and Calibration Controller shown in Fig. 1 perform computationally intensive functions that execute on separate threads. Detailed descriptions of the Lens Encoder Controller and Calibration Controller functions are provided in Section 2.2. Multithreading controllers allow demanding functions to run separately from the main thread, preventing the UI from locking and waiting on the functions to execute.

ACC utilizes OpenCV for calibration and computer vision functions [14]. An internal camera calibration API was created to expose only the necessary functions of OpenCV to the user interface. This separation of package-dependent calibration logic introduces encapsulation, allowing changes to be made to the internal calibration API without affecting other parts of ACC [15].

2.2. Software functionalities

ACC operates by completing three major tasks that must be performed in the sequence shown in Fig. 2. In step one, regression techniques create a mathematical relationship between the lens motor positions and their corresponding lens values for focus distance, iris, and zoom. Creating this mathematical relationship allows ACC to programmatically adjust the lens to desired configurations during calibration runtime. Step two is a repeated image capture process where a physical checkerboard target is captured at different locations within the camera's field of view. Computer vision functions from OpenCV that perform feature and motion detection are used to automatically find and capture the checkerboard target [14]. In Step three, image data collected during step two is analyzed and used in the calibration algorithm to calculate the intrinsic parameters for the camera.

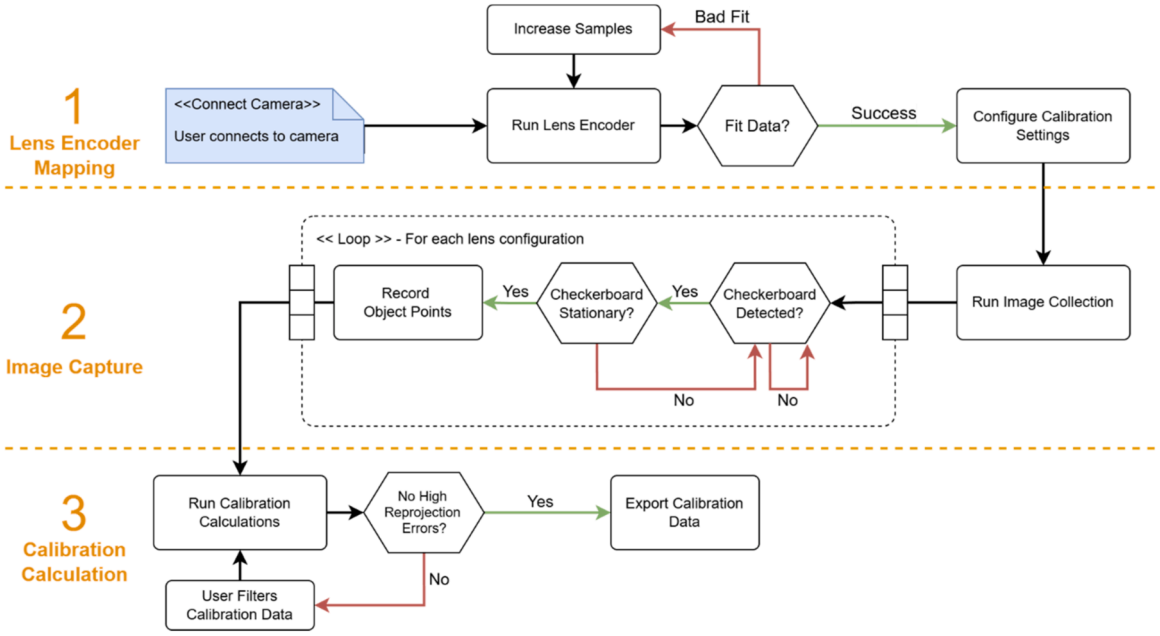


Fig. 2. Calibration activity diagram.

2.2.1. Lens encoder mapping

A lens encoder mapping process needs to be performed for ACC to automatically move camera lens components during the calibration process. To control the motors, ACC sends serial data packets containing the desired position of the motor to the LCUBE CUB-1. However, the motor encoders only track their internal position, and do not store information about the lens rings for focus, iris, and zoom. For ACC to understand which motor positions correspond to values on the lens rings, a relationship was created between the motor encoder and lens ring position data that is emitted from the camera hardware. First, enough evenly spaced data points are automatically recorded over the range of the minimum and maximum positions of the motors. Each data point is a recording of the motor positions, captured from the Serial Context Manager, and the current lens value for focus, iris, and zoom, captured from the Metadata Context Manager. This data collection process creates a discrete, non-linear relationship between motor encoder position and lens ring position. Next, regression is used to fit an exponential function to the data points, converting the relationship from discrete to continuous. The tools for this predictive modeling approach come from the Python package SciPy [16] and the R-Squared test is used to determine accuracy of fit. 98 % was determined as an acceptable accuracy since the goal is to achieve a near-perfect fit, rather than generalize the data. Using the model with an accuracy lower than 98% could result in the lens not aligning correctly during image capture. This value was chosen through empirical testing. The value can also be adjusted by a user if desired. This error is especially impactful at higher focus distances due to the asymptotic nature of the camera's focus distance range. Fitting the data points to a curve presented multiple challenges. A low degree polynomial model was too simple and did not match the range of the data points. Only polynomials of around 10 degrees or higher achieved an acceptable R-Squared value. However, the high degree polynomial produced a model that contained erratic curves between points. Therefore, a custom function shown in (1) was formulated to match the curved behavior of a lens. Another challenge arises from the variability in manufactured lenses. Focal distance and iris curves or ranges can differ based on the lens's brand, focal length, and usage over time, meaning (1) may not generalize and fit every lens. To mitigate this, linear interpolation is performed between data points if they do not achieve an R-Squared of 98 % using (1). Consequently, more samples are needed with the linear interpolation approach to achieve the acceptable R-Squared value. Once a relationship is established with motor position as the dependent variable, ACC can predict the motor position for any given lens ring value.

$$f(x) = \frac{a}{x^b} + c \tag{1}$$

2.2.2. Image capture

The next part of ACC requires two repeated steps which are the most time-consuming processes of camera calibration: collecting several images containing a target calibration pattern and moving the lens rings. The image collection process follows closely to the steps outlined in OpenCV's camera calibration documentation [17], but includes automation to speed up the collection process. The image capture step of Fig. 2 shows the looped procedure. To start, a user holds the physical checkerboard within the camera's field of view. ACC automatically detects and captures an image when the checkerboard is stationary for a user-defined duration. Users are notified of checkerboard detection and motion through real-time feedback in the UI. It is recommended that 10 images be captured for a single focus distance [18]. Between each set of captured images, the lens rings are programmatically changed using the Lens Encoder Mapping process. OpenCV's *findChessboardCorners* function was used to locate the checkerboard pattern and motion detection was developed using OpenCV's functions for image processing. Notably, Unreal Engine's calibration tool also uses *findChessboardCorners* function from the OpenCV library, providing a consistent basis for comparison when evaluating calibration speed. ACC reduces the time to capture images and enables one person to complete the calibration process that is typically performed by two people. Results are presented in the Validation section.

2.2.3. Calibration calculation

The collected image data is fed into the OpenCV function *CalibrateCamera* to produce the estimated focal length, image center, and lens distortion parameters for each focus distance. This function is an implementation of Zhang's method for calculating intrinsic parameters [19]. In addition, The *CalibrateCamera* function calculates a training accuracy value for each image, termed "reprojection error". Reprojection error is known as the discrepancy between the actual 2D points captured by the camera and the projected points estimated by the camera model [20]. Removing outlier images is a fundamental way of improving the camera model [21]. To support this, ACC flags images with high reprojection errors using a configurable threshold and presents visual feedback to the user. Users can review and remove problematic frames, after which ACC will automatically recalculate the calibration parameters. This interactive error-handling mechanism helps reduce the likelihood of poor calibration results caused by motion blur, lighting reflections, or checkerboard detection failures [21]. While the system does not yet perform automatic outlier rejection, its semi-automated approach provides transparency and control for the user. Once calculated, camera parameters are exported as an Unreal Engine Lens Asset (.ulens) file. This file can be directly imported into Unreal Engine's Lens Calibration plugin, which reads intrinsic parameters and distortion profiles for use in virtual camera pipelines. Since ACC uses the same Brown-Conrady distortion model as Unreal Engine, the exported .ulens files are natively compatible. While most standard use cases are supported, edge cases—such as unusual sensor sizes or off-center principal points—may require manual adjustments inside Unreal to ensure full fidelity of the camera model.

3. Validation

A validation process consisting of two methods is performed to evaluate the performance of the ACC software in terms of speed and lens encoder accuracy. While this study does not directly assess the calibration accuracy across different focal lengths, lighting conditions, or sensor configurations, it emphasizes efficiency improvements by comparing calibration runtimes with a baseline method.

The first method validates the accuracy of the lens encoder mapping model discussed in Section 2.2. The accuracy is evaluated by measuring the root mean square error (RMSE) between the actual values on the camera physical lens and the predicted values from the regression model. The data was collected by constructing 96 models, each with a different number of evenly spaced data points taken over the range of focus distances. Fig. 3 depicts the RMSE of the various models created for three different lenses with varying focal lengths. Based on the curves in Fig. 3, collecting a small number of samples is enough to predict the exponential curve of the focus on the lenses. In the worst case, where the Zeiss Compact Prime 35 mm lens is sampled only four times, the RMSE is measured at 526.44 motor encoder units, which is roughly equivalent to 3 cm of error at shorter focus distances (around 100 cm) and 41 cm of error at longer focus distances (around 800 cm). This validation method also reveals that the RMSE is negligible for sample sizes above 50. Therefore, ACC sets the default number of samples to complete a lens encoder mapping to 50.

The second validation method includes functional testing to compare the time taken to produce calibration results between ACC and the Unreal Engine calibration tool using a Zeiss CP 50 mm lens. A stopwatch was used to measure the two repeated portions of the calibration process: 1) image collection and 2) adjusting the focus distance of the camera. For both ACC and Unreal Engine, two time trials were completed for three different number of focus distances (i.e., 12 full

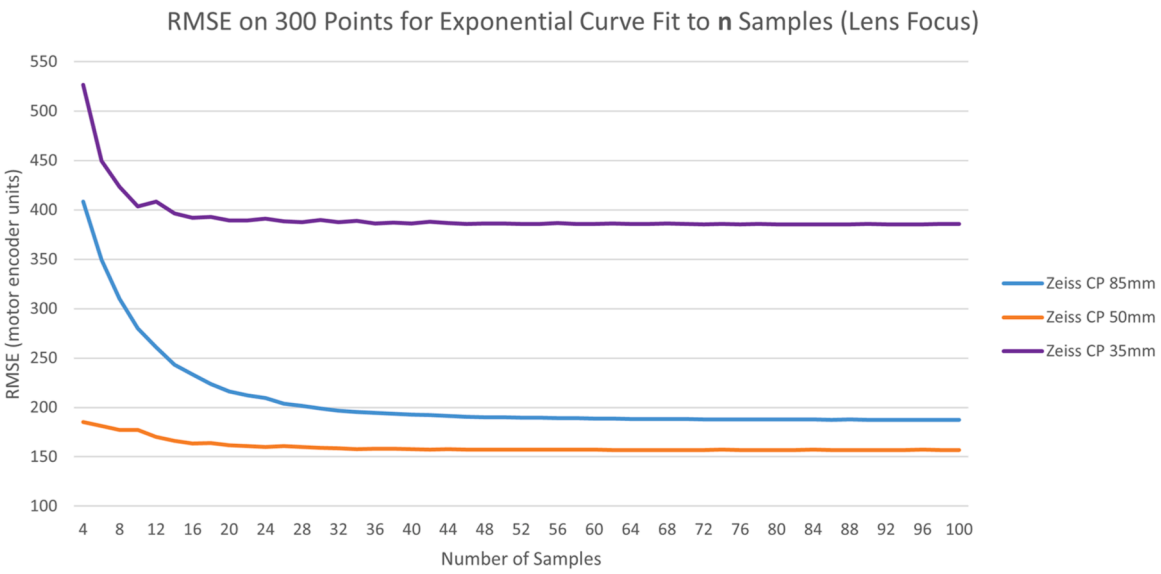
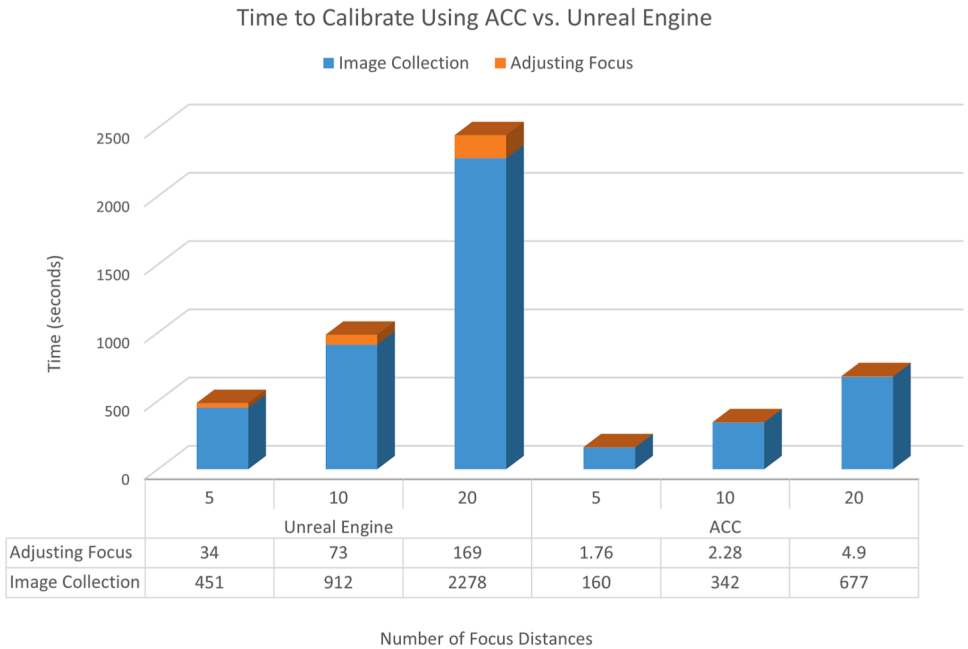


Fig. 3. RMSE evaluation of lens encoder mapping for focus distance.

camera calibrations). Fig. 4 shows the average of the time trials for each focus distance amount. ACC performs 3.59 times (26.68 min) faster than Unreal Engine when collecting images at 20 different focus distances (200 images collected). A larger contrast is shown in the time taken to adjust the lens focus rings during calibration. Unlike the image collection portion, the time taken to adjust lens focus using ACC does not increase significantly with higher number of focus distances. This is due to the camera lens rings being adjusted programmatically rather than manually.

4. Illustrative examples

This section describes the major user interfaces of ACC for completing camera calibration. The main window of ACC, shown at the top of Figs. 5 and 7, contain the elements necessary for establishing connections to camera hardware, configuring the calibration settings, and executing the major function such as image collection and calibration calculation. Fig. 5 specifically shows the lens encoder mapping user interface, which includes visual scatterplots of the relationship between lens metadata and the CForce lens encoders. Users may input the number of samples to be collected before fitting the data. Additionally, a verbose debug window details the lens encoder mapping process. Fig. 6 is a window that is shown during image capture. Typically, this interface is on a large monitor or television that can be seen as the checkerboard pattern is placed in front of the camera. Guided feedback is shown to the user, in this interface, such as the current focus distance, capture instructions, and the number of images remaining to be captured. Fig. 7 is the software interface once image capture is complete. All the captured image data can be viewed and any images that contain a high reprojection error can be deleted before performing numerical calibration. As the users of this software, in virtual production, are not experts in computer vision or computer programming, an easy to use interface,



	Unreal Engine			ACC		
	5	10	20	5	10	20
Adjusting Focus	34	73	169	1.76	2.28	4.9
Image Collection	451	912	2278	160	342	677

Fig. 4. Time to calibrate comparison.

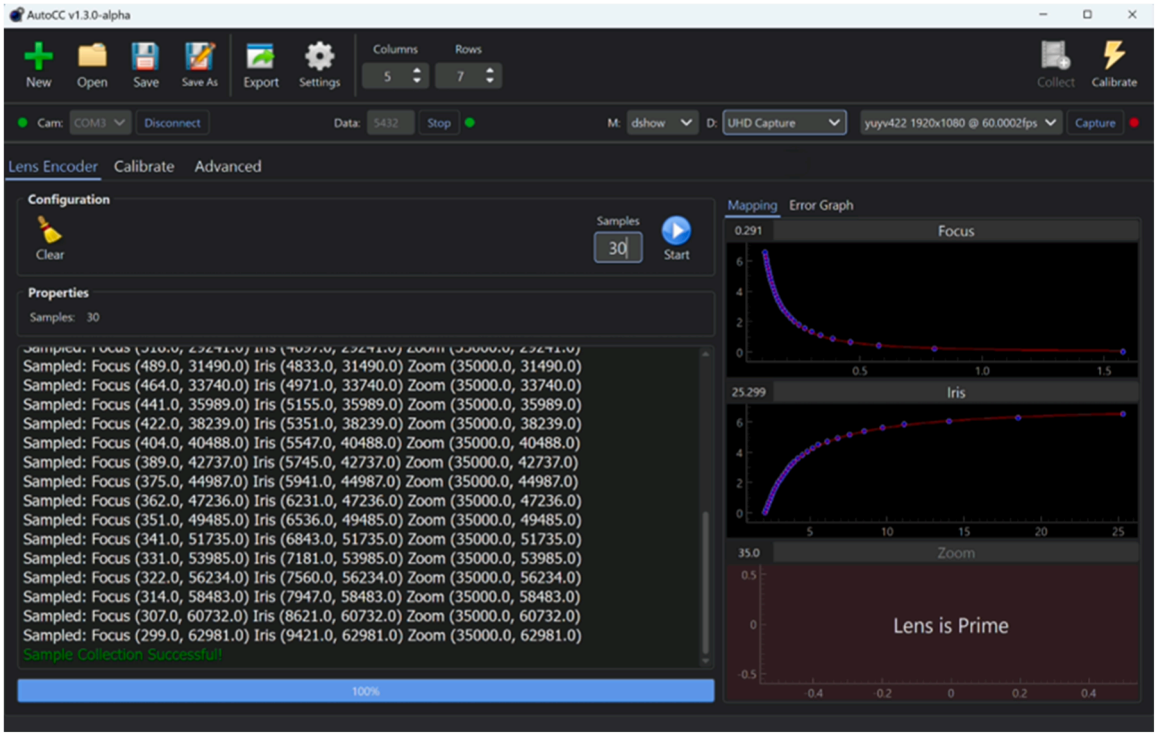


Fig. 5. Software interface for setting lens encoder mapping, including visual for fitting data (right) and debug window (bottom left).

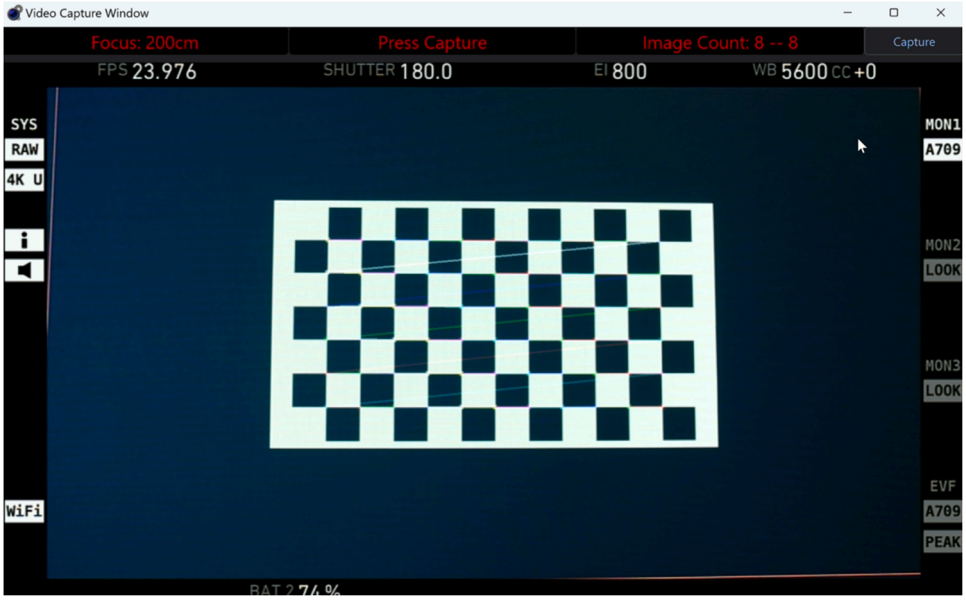


Fig. 6. User interface for image capture, including live video and guided feedback. This interface is for use on a screen as a user moves the checkerboard pattern to know that a sucessful capture has occurred.

showing enough detail for these users to complete calibration quickly and accurately was another needed outcome of this research.

5. Impact

This paper presents the Automated Camera Calibration tool (ACC), which addresses how automating lens movement and image capture can decrease the time to perform camera calibration compared to traditional methods that require extensive human involvement and manual adjustments. Consider a practical scenario where a virtual production studio needs to calibrate three distinct camera and lens combinations at 20 focal distances. Traditional calibration methods, like Unreal Engine's tool, can take over two hours, assuming no interruptions or recalibrations are necessary due to potential misalignments. In contrast, using the ACC tool, calibrating these three camera systems is estimated to take only 34 min, significantly reducing both time and cost for virtual production studios. This work validates improvements in calibration speed. However, it does not evaluate calibration accuracy and error across different lighting conditions, focal distances, or sensor configurations. Future research could explore these variables to better understand the

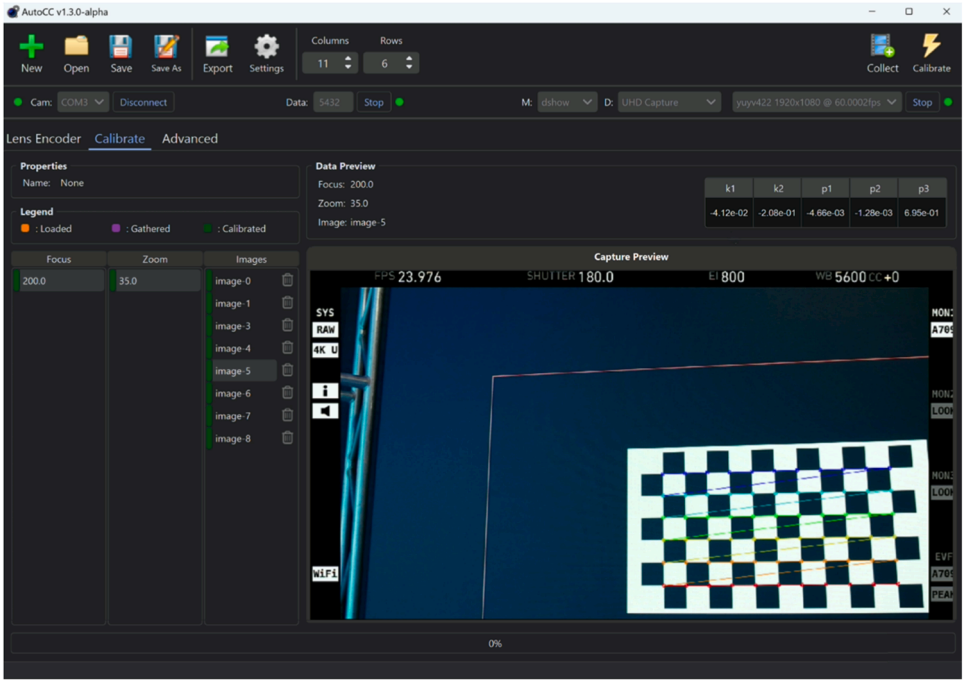


Fig. 7. Software interface for viewing and filtering calibration data. Once all captures are complete (from Fig. 6), this interface is used to include / omit image captures and then begin calibration computation.

accuracy and robustness of ACC under real-world scenarios. One question we asked was how other camera calibration operations, such as optimal positioning of the checkerboard target during image capture, could be automated to achieve more efficient and accurate calibration results. Another question asked was how an investigation into calibration practices for increasing accuracy, as outlined in [21], could supplement ACC's validation and create the potential for analyzing or configuration camera models within apps ACC.

Further developing ACC could expand its functionality and support more camera systems. Currently, ACC exports camera calibration parameters exclusively in Unreal Engine's .ulens format. Expanding support to other software platforms with virtual cameras like Unity would broaden its use. In addition, support could be added for camera models that include fisheye and wide-angle lenses.

CRediT authorship contribution statement

Nicholas D. Matthews: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Michael D. Holm:** Investigation, Conceptualization. **Anjali Gali:** Software, Methodology. **Alex R. Renner:** Writing – review & editing, Writing – original draft. **Eliot Winer:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Nicholas Daniel Matthews reports equipment, drugs, or supplies was provided by Renovo Media Group. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] "Virtual production." MarketResearch.com, 2024. [Online]. Available: https://www.marketresearch.com/Global-Industry-Analysts-v1039/Virtual-Production-38345593/.
[2] Clarke TA, Wang X, Fryer JG. The principal point and CCD cameras. Photogramm Rec Oct. 1998;16(92):293–312. https://doi.org/10.1111/0031-868X.00127.
[3] Ricolfe-Viala C, Sanchez-Salmeron A-J. Lens distortion models evaluation. Appl Opt Oct. 2010;49(30):5914. https://doi.org/10.1364/AO.49.005914.
[4] Zhang Z. Camera parameters (intrinsic, extrinsic). In: Ikeuchi K, editor. Computer vision. Cham: Springer International Publishing; 2021. p. 135–40. https://doi.org/10.1007/978-3-030-63416-2_152. editor.
[5] Zhang Y-J. Camera calibration. 3-D Computer vision. Singapore: Springer Nature Singapore; 2023. p. 37–65. https://doi.org/10.1007/978-981-19-7580-6_2.
[6] Epic Games. Unreal engine. Apr. 25, https://www.unrealengine.com; 2019.
[7] Brown D. Decentering distortion of lenses [Online]. Available: https://api.semanticscholar.org/CorpusID:117271607; 1966.
[8] K. Liao et al., "Deep learning for camera calibration and beyond: a survey," 2023, *arXiv*. doi: 10.48550/ARXIV.2303.10559.
[9] Perek P, Makowski D, Mielczarek A, Napieralski A, Sztoch P. Towards automatic calibration of stereoscopic video systems. In: 2015 22nd International Conference Mixed Design of Integrated Circuits & Systems (MIXDES). IEEE; Jun. 2015. p. 134–7. https://doi.org/10.1109/MIXDES.2015.7208496.
[10] Tan L, Lin Y, Ye H. Modeling quality attributes in software product line architecture. In: 2012 Spring Congress on Engineering and Technology. IEEE; May 2012. p. 1–5. https://doi.org/10.1109/SCET.2012.6341971.
[11] "ARRI | inspiring images. since 1917". 1917. Accessed: Nov. 04, 2024. [Online]. Available: https://www.arri.com/en.
[12] Y. Mendelovich, "The cameras behind Oscar 2024," Y.M. Cinema Magazine. 2024. Accessed: Nov. 04, 2024. [Online]. Available: https://ymcinema.com/2023/10/20/the-cameras-behind-oscar-2024-arricam-makes-a-huge-comeback/.
[13] FFmpeg developers. FFmpeg. Apr. 25, https://www.ffmpeg.org/; 2024.
[14] Bradski G. The OpenCV library. Dr Dobb's J Softw Tools 2000.
[15] Robillard MP. Encapsulation. Introduction to software design with java. Cham: Springer International Publishing; 2022. p. 13–41. https://doi.org/10.1007/978-3-030-97899-0_2.
[16] Virtanen P. SciPy: fundamental algorithms for scientific computing in Python. Nat Methods 2020;17:261–72. https://doi.org/10.1038/s41592-019-0686-2.
[17] OpenCV developers, *camera calibration*. Accessed May 2025. [Online]. Available: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html.
[18] HTC Vive, "Collecting calibration data using the camera calibration tool." Accessed May 2025. [Online]. Available: https://www.vive.com/us/support/camtrack/category_howto/collecting-calibration-data-using-the-camera-calibration-tool.html.

[19] Zhang Z. A flexible new technique for camera calibration. IEEE Trans Pattern Anal Mach Intell Nov. 2000;22(11):1330–4. https://doi.org/10.1109/34.888718.
[20] Liu Y, et al. Comparison study of three camera calibration methods considering the calibration board quality and 3D measurement accuracy. Exp Mech Feb. 2023;63 (2):289–307. https://doi.org/10.1007/s11340-022-00905-y.
[21] Semeniuta O. Analysis of camera calibration with respect to measurement accuracy. Procedia CIRP 2016;41:765–70. https://doi.org/10.1016/j.procir.2015.12.108.