

Neural Real-Time Recalibration for Image-based Multi-Camera Systems

Reza Talakoob

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science at

Concordia University

Montréal, Québec, Canada

July 2024

© Reza Talakoob, 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Reza Talakoob**

Entitled: **Neural Real-Time Recalibration for Image-based Multi-Camera Systems**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

Dr. Ching Yee Suen Chair

Dr. Ching Yee Suen Examiner

Dr. Mirco Ravanelli Examiner

Dr. Charalambos Poullis Supervisor

Approved by

Dr. Joey Paquet, Chair
Department of Computer Science and Software Engineering

_____ 2024

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Neural Real-Time Recalibration for Image-based Multi-Camera Systems

Reza Talakoob

Traditionally, multi-camera calibration relies on physical objects and a controlled environment to achieve high-accuracy results, but this cannot be extended to real-time. Recent advancements in deep learning (DL) have enabled image-based camera calibration, offering real-time operation but often sacrificing accuracy for speed. In this thesis, we address this trade-off by proposing a novel approach that leverages DL models for online and real-time multi-camera calibration with high precision. Current DL methods for camera calibration, while fast, often struggle with real images captured in the wild due to varying conditions. Our approach tackles this challenge by introducing a deep learning model designed for online calibration scenarios from images with low inference time. This model adapts to various camera poses efficiently, ensuring robust calibration across diverse viewpoints.

Central to our approach is the introduction of perturbations into the camera parameters, leveraging known initial parameters and 3D fiducial coordinates. This technique allows the model to learn and predict accurate camera parameters even in uncontrolled settings. Extensive experiments demonstrate the effectiveness of our proposed approach, particularly in scenarios requiring real-time calibration with high precision.

Acknowledgments

Hereby I would like to express my gratitude to my supervisor, Professor Charalambos Poullis for the amazing opportunity to work in the Immersive and Creative Technologies Lab. I greatly enjoyed working with and learning along the way of such a person, appreciating his exceptional field knowledge and commitment to professional ethics.

Also, I cannot appreciate my loving parents enough for their unconditional support and love. My friends, especially my partner, Dorsa, whom I could not imagine living and researching far from home without their presence.

Forever grateful!

Contents

List of Figures	viii
List of Tables	xi
1 Introduction	1
1.1 Single-Camera Calibration	1
1.2 Multi-Camera Calibration	2
1.3 Online multi-Camera Calibration	2
1.4 Motivation and Challenges	2
1.5 Proposed Solution	3
1.6 Contribution	3
1.7 Thesis Outline	3
2 Literature Review	4
2.1 Camera Models	4
2.1.1 Pinhole Camera Model	5
2.1.2 Fisheye Camera Model	5
2.2 Relative Camera Poses in Multi-Camera Setups	6
2.3 Deep Learning Models	6
2.3.1 Convolutional Neural Networks(CNNs)	7
2.3.2 Residual Neural Network (ResNet)	9
2.3.3 Transformers and Attention Mechanism	9

3	Neural Real-Time Recalibration for Infrared Multi-Camera Systems	12
3.1	Introduction	13
3.2	Related Work	15
3.3	Camera Model	16
3.4	Methodology	18
3.4.1	Dynamic Camera Pose Synthesis	18
3.4.2	Network Architecture	21
3.4.3	Loss	21
3.4.4	Training	22
3.5	Experimental Results	22
3.5.1	Generalization to Arbitrary Configurations & Calibration Objects	23
3.5.2	Comparison with CMM-calibrated multi-camera system.	24
3.6	Conclusion	24
4	Supplementary Material	26
4.1	Network Architecture	26
4.1.1	Point-based variant.	27
4.1.2	Image-based variant.	28
4.1.3	Ablation Study	28
4.2	Comparison with Standard Calibration and Optimization Techniques	30
4.2.1	Inference vs. Recalibration	30
4.2.2	Inference vs. Minimizing Reprojection Errors	32
4.3	Qualitative Results	33
4.4	Training: Less Effective Strategies	34
4.4.1	Regularization terms	34
4.4.2	Losses	35
4.5	Limitations & Discussion	36
4.5.1	Adhering to Operational Specifications	36
4.5.2	Limited Range of Motion for Image-based	37

4.5.3	Model Customization	37
4.5.4	Decalibration Detection	37
4.6	Model Training Progression Animation	38
5	Extension to Fisheye Camera Model	39
5.1	Fisheye Camera Model Training and Test Results	39
5.2	Detection and Tracking of Bright Points in Image Sequences	41
5.2.1	Image Preprocessing, Blob Detection and Gaussian Fitting	41
5.2.2	Point Tracking Across Images	41
5.3	Evaluation Metrics	42
5.3.1	Normal Vector Comparison	42
5.3.2	Angle Comparison	42
5.3.3	Midpoint Reprojection Error	43
6	Conclusion and Future work	44
	Bibliography	45

List of Figures

Figure 2.1	Same scene captured by a standard and a fisheye lens. Source: [1]	5
Figure 2.2	Convolutional Neural Network pipeline for a classification task. Source: [2]	7
Figure 2.3	Convolutional filtering process. Source: [2]	8
Figure 2.4	Max Pooling vs Average Pooling. Source: [2]	8
Figure 2.5	a residual block showing the concept of residual learning by adding skip connection as described in [3].	9
Figure 2.6	VIT Model review as shown in [4]: Splitting input image into fixed-size patches, linearly embedding each of them, adding positional embedding, and feeding the resulting sequence of vectors to a standard Transformer.	11
Figure 3.1	Discontinuities introduced by quaternions and Euler angle representations hinder network learning efficiency as shown in [5].	17
Figure 3.2	Technical overview. Our methodology begins with the synthesis of dynamic camera poses (see top fig.). Given spherical angles ϕ (azimuth), θ (elevation), along with the intrinsic rotation angle α , the OEM calibration parameters, the maximum perturbation limit κ , and known 3D fiducials (e.g. a cube calibration object), this module performs two primary functions: (i) it synthesizes poses for the multi-camera system, and (ii) it computes the projected 2D points. Subsequently, it employs point splatting to render images of these points. During training (see bottom fig.), the synthesizes poses and projected points (alternatively rendered images) are used to train the neural network. A differentiable projection ensures the propagation of gradients from the loss \mathcal{L} back to the predicted camera parameters.	19

Figure 3.3	Dynamic Camera Pose Synthesis. Our framework supports arbitrary configurations of multiple cameras as well as a wide range of calibration objects. To synthesize camera poses, we employ a random uniform sampling strategy across three dimensions to ensure a comprehensive exploration of the pose space: azimuth (θ), elevation (ϕ), and roll (α), where $\theta \sim \mathcal{U}(0, 2\pi)$, $\phi \sim \mathcal{U}(0, \frac{\pi}{2})$, and $\alpha \sim \mathcal{U}(0, 2\pi)$. Additionally, Original Equipment Manufacturer (OEM) calibration parameters and a predefined maximum perturbation limit (κ) are incorporated. . . .	21
Figure 4.1	Visualization of predicted vs ground truth camera poses. The calibration object is a sphere with 64 fiducials. The multi-camera system configuration is O-shaped comprising 10 cameras. For closer inspection please refer to the interactive visualization in the cameras.html file.	29
Figure 4.2	Runtime for traditional camera calibration. Exponential growth in calibration time with increasing number of images (red; 1 camera). Linear increase w.r.t. LM iterations on 100 images (blue; 1 camera). Ours; real-time ($\tau_{min}^{i=1} = 0.0026s, \tau_{max}^{i=10} = 0.012s$) for increasing number of cameras $2^i, 1 \leq i \leq 10$ (black).	31
Figure 4.3	Effect of decalibration on reprojection error (RMSE). OEM intrinsic parameters are perturbed by 20%, i.e., $\kappa \in [-10\%, 10\%]$, simulating potential decalibration. Reprojection error without intervention (red), with 1 iteration of Bundle Adjustment (green), with 25 iterations (blue); 1000 trials.	32
Figure 4.4	Reprojection of 3D Fiducials with Predicted Camera Poses (With 5% Perturbation). This figure illustrates the resilience and accuracy of our pose estimation model during decalibration of the camera parameters of up to 5% perturbation. The calibration cube and the O-shaped arrangement of 10 cameras remain constant as in Figure 4.5, allowing for a direct comparison across different testing conditions. Note: The reprojected points are shown in white. For optimal visibility, please zoom in.	33

Figure 4.5 **Reprojection of 3D Fiducials with Predicted Camera Poses (Test Data**

Generated Without Perturbation). This figure demonstrates the reprojection accuracy in a O-shaped multi-camera setup comprising 10 cameras, with the calibration object being a cube with 8 fiducials. The comparison between the predicted (in white) and ground-truth (color-coded; enlarged for visualization purposes) projections demonstrates the precision of our model in the absence of perturbation. Note: The reprojected points are shown in white. For optimal visibility, please zoom in. 34

Figure 4.6 **Illustration of the proposed loss compared to auxiliary regularization**

and loss terms. This figure demonstrates that minimizing the proposed loss leads to a reduction in auxiliary losses. However, this is a non-reciprocal relationship where the inclusion of auxiliary losses does not further improve model performance beyond the capabilities of the proposed loss which is a combination of the reprojection error, the RMSE error of the parameters, and the geodesic error. Here, we present the training loss graphs for the point-based variant. As previously described, the distortion coefficients are scaled by scaling factor $\lambda_{scale} = 1000$ 36

Figure 5.1 Synthetic image of the Fisheye camera setup, with 6 cameras and 8 key points. 40

Figure 5.2 The training loss of the fisheye camera model with 5% perturbation. 40

List of Tables

Table 3.1 **Experimental results.** \mathbf{RE}_{avg}^{20K} is the average RMSE reprojection error on 3 different trials (synthetic test sets, each comprising 20,000 data samples). Training on all models includes adding a **Max Perturb.** $\kappa \in [\mathbf{min}\%, \mathbf{max}\%]$ to the OEM camera intrinsic and extrinsic parameters respectively. $N_{fid.}$ and N_C are the number of 3D fiducials and the number of cameras, respectively. The rotation angle α remains the same in all experiments i.e., $\alpha \sim \mathcal{U}(0, 2\pi)$. The parameters for Pt and Img models are $\sim 33\text{m}$ and $\sim 86\text{m}$, respectively. 23

Table 4.1 **Ablation study results.** Training on all models includes a perturbation of **Max Perturb.** $\kappa \in [-\mathbf{2.5}\%, +\mathbf{2.5}\%]$ of the OEM intrinsic parameters, utilizing 10 cameras ($N_C = 10$) and a calibration object with 8 fiducials ($N_{fid.} = 8$). Pt models have been trained for 300k epochs and Img models for 150k epochs. 30

Table 5.1 **Experimental Results:** $\mathbf{RE}_{camera}^{20K}$ is the average RMSE reprojection error on 3 different trials (synthetic test sets, each comprising 20,000 data samples). Training on all models includes adding a **Max Perturb.** $\kappa \in [\mathbf{min}\%, \mathbf{max}\%]$ to the OEM camera intrinsic and extrinsic parameters respectively. The number of 3D fiducials and the number of cameras are 8 and 10 respectively in these experiments. The rotation angle α remains the same in all experiments i.e., $\alpha \sim \mathcal{U}(0, 2\pi)$ 40

Chapter 1

Introduction

In this chapter, we introduce the fundamental concepts of the camera calibration process, emphasizing its application in multi-camera systems. We delve into the motivation behind real-time multi-camera calibration, highlighting the associated challenges. Lastly, we provide a brief overview of our proposed solution to these challenges. This introduction sets the foundation for understanding the complexities in our approach to achieving efficient and accurate calibration in dynamic environments.

1.1 Single-Camera Calibration

Single-camera calibration is the problem of estimating the parameters of a camera, including the intrinsic and extrinsic parameters, which consist of the focal length, principal points, and lens distortion coefficients. Extrinsic parameters describe the camera's rotation and translation with respect to the world coordinates, defining its position and orientation in space. These parameters are essential for accurate 3D reconstruction, object recognition, and other applications requiring precise geometric information. Traditional single-camera calibration involves capturing images of a known calibration pattern, such as a checkerboard, from multiple angles and distances. Such calibration algorithms then process these images to estimate the camera parameters, correcting for distortions and aligning the camera's view with the real-world scene.

1.2 Multi-Camera Calibration

In multi-camera systems, multiple cameras are used simultaneously to capture different perspectives of the same scene. Multi-camera systems are widely used in many applications in computer vision, especially when we need accurate and robust 3d construction and measurements across multiple cameras. In Multi-Camera Calibration we aim to find the relative position and orientation of each camera in the setup with respect to the base camera. This allows us to reconstruct the scene in 3d and determine the location of the cameras within that scene.

1.3 Online multi-Camera Calibration

While offline and precise calibration of the multi-camera setup is essential for tasks requiring high precision, online calibration allows adjustments to the camera parameters in real-time. In an online setup, calibration needs to be performed dynamically, as cameras may be moved during an operation. This is particularly needed in scenarios where the multi-camera configuration is subject to change after deployment, such as a surgery room.

1.4 Motivation and Challenges

The primary challenge in online multi-camera calibration lies in both efficiency and the accuracy of estimating the camera parameters without disrupting the system operation. Traditional calibration methods involve capturing calibration patterns from all the cameras in a multi-camera setup simultaneously, followed by an off-line process to compute the cameras' parameters which is not suitable for online setups because it requires system downtime. Instead, the online approach relies on iterative refining of the camera parameters. These methods often employ feature-based tracking algorithms to detect and track calibration patterns or objects in real-time. By leveraging information across multiple cameras, these algorithms can estimate camera parameters efficiently while the system is still in operation.

1.5 Proposed Solution

In response to the challenges discussed above, our approach utilizes deep learning architecture to learn the representation and regress the camera parameters. To have sufficient training data for a deep neural network we also propose a way of generating synthetic data in various views. This allows the model to see the training data in different camera poses and learn the underlying patterns from the images, generalizing well on possible views. This ensures the production of precise and detailed results. The goal of our approach is to achieve precise calibration and robust generalization through extensive training while ensuring efficient real-time inference.

1.6 Contribution

The result of my work was a joint publication¹ titled "Neural Real-Time Recalibration for Infrared Multi-Camera Systems" which is currently under review. Specifically, my contribution to this work was introducing the camera synthesis module given the initial configuration of the camera setup, designing an image-based variant of the neural calibration model, and exhaustive experimentation to achieve the best performance on different scenes and camera configurations given RGB images as input. Beyond the publication, we extend our experiments to also support calibration for another common camera model, such as the fisheye, which is typically used for capturing a wider field of view.

1.7 Thesis Outline

This thesis is organized into 5 chapters. Chapter 2 offers the foundational knowledge necessary for comprehending the proposed method. It covers the topics of different camera models and the deep learning architectures and mechanisms utilized in our proposed solution. In Chapter 3, we present our paper titled Neural Real-Time Recalibration for Infrared Multi-Camera Systems. In Chapter 4, we provide the supplementary materials for our main paper. Finally, in Chapter 5, we conclude the thesis, showing the possible extension to our methods.

¹I share first authorship with Benyamin Mehmandar

Chapter 2

Literature Review

In this chapter, we explain various camera models and their associated parameters that provide an understanding of the fundamental concepts behind camera functionality and imaging. We explore two different camera models, the pinhole camera model, and the fisheye. following this model overview, we shift our focus to the Deep-learning approaches used for the image-based camera calibration. This section provides a detailed review of the commonly used neural network architectures, such as convolutional neural networks (CNNs), residual networks (ResNets), and transformer-based models. These architectures serve as the backbone of our deep learning models, processing input images to extract image features.

By leveraging these deep learning techniques, we aim to enhance a robust and precise real-time camera calibration which is also efficient for inference. This chapter also includes the fundamental concepts behind the architecture and the mathematical theories needed for the camera calibration process.

2.1 Camera Models

Understanding the characteristics of different camera models is paramount for camera calibration. Two widely used camera models are the pinhole camera model and the fisheye camera model, each offering distinct advantages and limitations. Figure 2.1, from [1] shows the effect of a Fisheye lens, enabling the capturing of a wider field of view.



Figure 2.1: Same scene captured by a standard and a fisheye lens. Source: [1]

2.1.1 Pinhole Camera Model

The pinhole camera model, also known as the perspective camera model, is the simplified form of how light rays enter a camera and form an image. In this model, the light passes through a single point that is called the pinhole (camera center) and projects onto an image plane at a certain distance from the center. This projection results in perspective distortion where objects further from the camera would be smaller in image space. Equation 1 shows the distortion equation for the pinhole projection given the radial distortion k_1, k_2, k_3 and p_1, p_2 the tangential distortion coefficients, where $r^2 = x^2 + y^2$. Here (x, y) are the coordinates of the projected point using the projection matrix $\mathbf{M}_{\text{projection}} = \mathbf{M}_{\text{camera-to-image}} \cdot \mathbf{M}_{\text{world-to-camera}}$ and $\mathbf{P}_{\text{image}} = \mathbf{M}_{\text{projection}} \cdot \mathbf{P}_{\text{world}}$

$$\begin{aligned} x_{\text{dist}} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2), \\ y_{\text{dist}} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y^2) + 2p_2 xy \end{aligned} \quad (1)$$

2.1.2 Fisheye Camera Model

In contrast to the Pinhole model, Fisheye cameras provide a wider field of view which can capture a broader range of the scene with reduced perspective distortion. Fisheye lenses are specialized optics designed to provide an ultra-wide field of view, spanning up to 180 degrees. Unlike traditional lenses, these lenses utilize complex optical elements to capture a nearly hemispherical view of the scene, resulting in significant distortion near the edges of the images. Equation 2 shows the distortion equation where $\theta = \tan^{-1}(r)$ and $r^2 = x^2 + y^2$. The Fisheye distortion model only includes radial Fisheye distortion, k_1, k_2, k_3, k_4 . Unlike the rectilinear lens model, the polynomial

describing the radial distortion is a function of an angular distance from the center of perspective, rather than a linear distance in the image

$$\begin{aligned} x_{\text{dist}} &= \frac{\theta}{r}(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8)x, \\ y_{\text{dist}} &= \frac{\theta}{r}(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8)y \end{aligned} \tag{2}$$

2.2 Relative Camera Poses in Multi-Camera Setups

One important aspect of a multi-camera system in our design is the relative pose of each camera with respect to the base camera. This will help us with the reconstruction, triangulation, and parameter optimization. In this part, we assume that we have the Rotation and Translation of each camera in the world coordinate system. Assume R_i and T_i to be the camera i 's Rotation and Translation w.r.t the world, and R_0 and T_0 are the Rotation and Translation of the base camera. Then we can calculate the relative Rotation and Translation using equation 3.

$$\begin{aligned} R_{i0} &= R_0^{-1}R_i, \\ T_{i0} &= R_0^{-1}(T_i - T_0) \end{aligned} \tag{3}$$

2.3 Deep Learning Models

Convolutional Neural Networks(CNNs) have revolutionized Computer Vision to tackle complex image-based tasks with high accuracy. Since the introduction of CNNs, many variations have been proposed to learn the underlying representation of the raw images for tasks such as image classification, Object Detection, and Segmentation. CNNs capture spatial and temporal dependencies within data and find the underlying patterns and structures. With the emergence of large-scale datasets such as ImageNet [6], different architectures including AlexNet [7], VGG [8], Resnet [3] and others have further improved performance and efficiency, candidating them as potential backbones for image-based tasks. While Resnet benefits from skip connections and addresses the vanishing gradient issue, VIT [4] adopts a transformer-based architecture inspired by the improvements in the field of Natural Language Processing, demonstrating remarkable performance in image classification tasks by processing image patches. This alternative family of architectures offers further exploration in

image-based tasks, suggesting a departure from the conventional CNN paradigm.

2.3.1 Convolutional Neural Networks(CNNs)

One of the most popular deep neural networks is convolutional neural networks (also known as CNN or ConvNet), which are mostly applied to visual imagery in deep learning. ConvNets are specifically designed to process an image in a grid-like manner, with notable efficiency and effectiveness. ConvNets, as shown in 2.2 from [2] consists of interconnected layers, each serving a specific function in the feature extraction process. The core components are convolutional layers, pooling layers, and fully connected layers. These networks perform feature extraction by applying convolution operation to input images, and then the pooling layers reduce spatial dimensions to overcome overfitting. Activation functions tend to introduce some non-linearity to the ConvNets. Rectified Linear Unit (Relu) ensures sparse activation and accelerates convergence during training. Other functions such as tanh and sigmoid are among other nonlinear options. Finally, fully connected layers are the head for classification or regression. In the following subsections, we delve deeper into the functionality of the convolution and pooling layers.

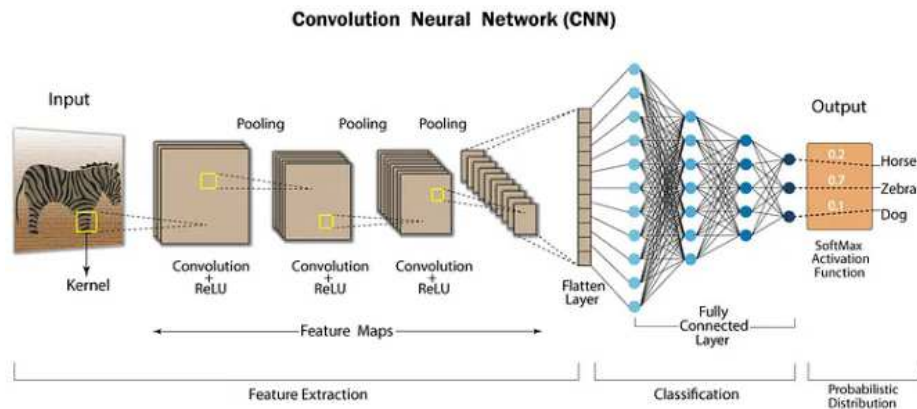


Figure 2.2: Convolutional Neural Network pipeline for a classification task. Source: [2]

Convolutional Operation

The convolutional operation involves sliding a kernel over the input image to extract local patterns. Each convolutional layer may have several filters or kernels that capture different aspects of

their input. The output of the filters are called feature maps which highlight the spatial information of the input. Figure 2.3, from [2] demonstrates the convolutional filtering process while sliding a kernel window on it. When we slide a kernel filter over an input in a CNN, the output size may reduce depending on the size of the filter. To manage the output size changes, there are different padding strategies such as zero padding and full padding.

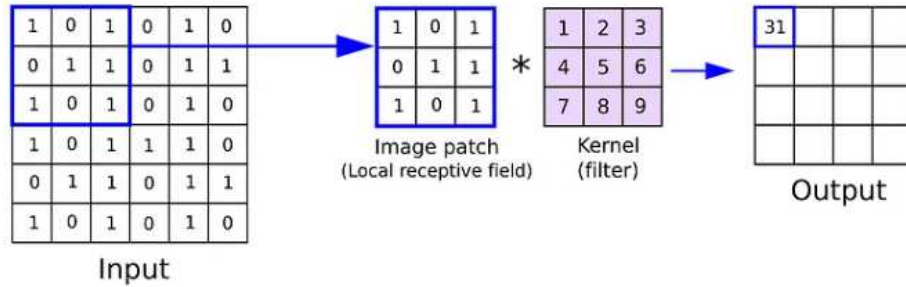


Figure 2.3: Convolutional filtering process. Source: [2]

Pooling Layers

Pooling layers serve the purpose of downsampling the feature map size, reducing the complexity while keeping the essential information. Max pooling and average pooling are the two most commonly employed to create compact representations. Pooling also enhances robust feature extraction. Figure 2.4 from [2] shows the max pooling and average pooling process, demonstrating their functionality as suggested by their names. In max pooling, the maximum value from each region is selected, whereas in average pooling, the average value of each region is calculated.

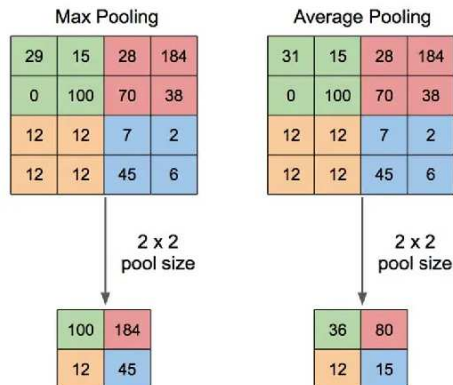


Figure 2.4: Max Pooling vs Average Pooling. Source: [2]

2.3.2 Residual Neural Network (ResNet)

Residual Networks (ResNet) addresses a fundamental challenge in training deep neural networks. It has revolutionized the field by presenting a novel architectural paradigm that helps us to train exceedingly deep networks, surpassing the limitations caused by vanishing gradients. The core idea of Resnets lies in the introduction of residual blocks, as shown in 2.5, which enables propagation through deep layers while mitigating the problem encountered in traditional deep networks. By adding skip connections bypassing one or more layers, Resnets pass information from earlier layers to the subsequent ones, enabling the network to learn the residual mappings. The added Identity mapping helps preserve the input information of the residual block, ensuring that the deep layers can learn the residuals without missing the information flow. This means the gradient can flow effectively through the network, improving the overall performance and addressing the vanishing gradient problem. Equation 4 shows the formula for the l -th residual block, with the input denoted as x_l and the output x_{l+1} .

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \{W_{l,i}\}) \quad (4)$$

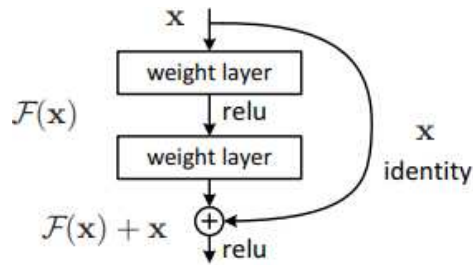


Figure 2.5: a residual block showing the concept of residual learning by adding skip connection as described in [3].

2.3.3 Transformers and Attention Mechanism

Transformers [9] introduced in the context of sequence-to-sequence tasks like machine translation, are a type of neural network architecture designed to capture long-range dependencies and information effectively. Unlike recurrent or convolutional architectures, transformers rely on the self-attention mechanism to process input sequences and generate the embedded representations.

Self-Attention Mechanism

The self-attention mechanism is a key component of transformers. It allows the model to focus only on relevant parts of the sequence by assigning different weights to the tokens based on their importance. This enables the model to capture the patterns within the data. Mathematically self-attention is described in the equation 5. Given an input sequence represented by the matrix $X \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the dimensionality of the embeddings, the self-attention mechanism computes three matrices: Query Q , Key K , and Value V , which are linear transformations of X :

$$\begin{aligned} Q &= XW_Q, \quad K = XW_K, \quad V = XW_V. \\ \text{Attention}(Q, K, V) &= \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \end{aligned} \tag{5}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}$ are learned parameter matrices and d_k is the dimensionality of the query, key, and value vectors. The attention scores are computed by taking the dot product of the query and key vectors, scaling by the square root of d_k , and finally applying a softmax function.

Positional Encoding

Transformers lack the inherent sequential nature of recurrent neural networks. To address this, positional encoding is added to the input embeddings to provide information about the position of each token in the sequence. The positional encoding can be defined using sine and cosine functions of different frequencies:

$$\begin{aligned} PE_{(pos, 2i)} &= \sin \left(\frac{pos}{10000^{2i/d}} \right) \\ PE_{(pos, 2i+1)} &= \cos \left(\frac{pos}{10000^{2i/d}} \right) \end{aligned} \tag{6}$$

where pos is the position and i is the dimension. These encodings are added to the input embeddings to incorporate positional information and make use of the order.

Vision Transformer (ViT)

Vision Transformers have recently emerged as an alternative to CNNs. In many computer vision tasks, the attention mechanism is used in conjunction with CNNs or used to substitute certain aspects of CNNs. However, a pure transformer-based model applied to image patches can work exceptionally well on image tasks. ViT extends the capabilities of transformers to the domain of visual data. Self-attention is the fundamental building block of transformer architectures, which enables global interactions between the input tokens. ViT treats images as sequences of 16 flattened patches. These patches are further embedded into high-dimensional features, which are processed by transformer layers to capture dependencies within the entire image. By leveraging self-attention, ViT mitigates the need for architectural components such as convolutional filters. ViT shows remarkable generalization capabilities, surpassing the performance of CNN-based models on various benchmarks, often requiring fewer parameters. Figure 2.6 shows an overview of the ViT model and the transformer encoder block used within it.

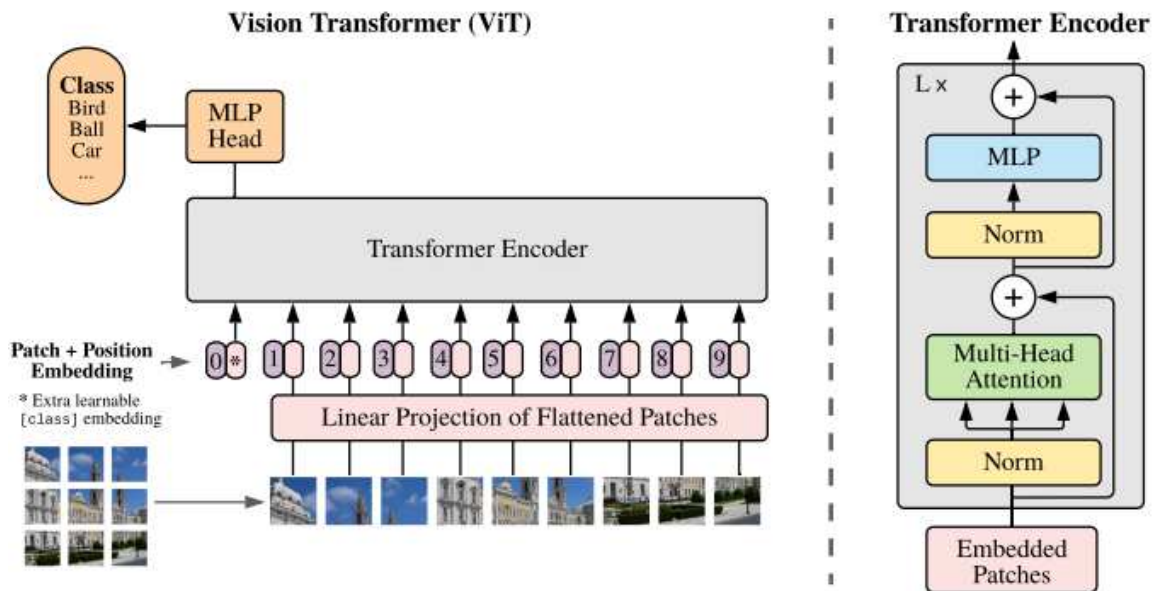


Figure 2.6: ViT Model review as shown in [4]: Splitting input image into fixed-size patches, linearly embedding each of them, adding positional embedding, and feeding the resulting sequence of vectors to a standard Transformer.

Chapter 3

Neural Real-Time Recalibration for Infrared Multi-Camera Systems

The following is a verbatim copy of the manuscript currently under review, titled "Neural Real-Time Recalibration for Infrared Multi-Camera Systems", authored by Benyamin Merhandar¹, Reza Talakoob², and Charalambos Poullis.

Abstract

In this work, we address the challenge of real-time, highly-accurate calibration of multi-camera infrared systems, a critical task for time-sensitive applications. Unlike traditional calibration techniques that lack adaptability and struggle with on-the-fly recalibrations, we propose a neural network-based method capable of dynamic real-time calibration. The proposed method integrates a differentiable projection model that directly correlates 3D geometries with their 2D image projections and facilitates the direct optimization of both intrinsic and extrinsic camera parameters. Key to our approach is the dynamic camera pose synthesis with perturbations in camera parameters, emulating realistic operational challenges to enhance model robustness. We introduce two model variants: one designed for multi-camera systems with onboard processing of 2D points, utilizing the direct 2D projections of 3D fiducials, and another for image-based systems, employing color-coded projected

¹equal contribution

²equal contribution

points for implicitly establishing correspondence. Through rigorous experimentation, we demonstrate our method is more accurate than traditional calibration techniques *with or without* perturbations while also being *real-time*, marking a significant leap in the field of real-time multi-camera system calibration.

3.1 Introduction

Camera calibration is the fundamental process of estimating the camera’s intrinsic and extrinsic parameters and is an essential part of many computer vision systems. Accurate calibration is important in various applications, ranging from 3D reconstruction to augmented reality, especially in settings demanding high accuracy, like surgical environments. Traditional calibration methods provide analytical frameworks for addressing camera calibration. However, they require capturing an object of known geometry from multiple viewpoints, then extracting points and establishing correspondences. The inherent computational complexity of these conventional methods tends to increase dramatically with the number of cameras, images, and correspondences, making them impractical for real-time applications.

In time-critical applications requiring high accuracy, standard commercial multi-camera systems have a fixed camera configuration and come pre-calibrated from manufacturers. However, the calibration of these systems deteriorates over time because of wear and tear and, more commonly, because of the buildup of debris on critical components like the fiducials or lenses. In the field of vision-based computer-assisted surgery, calibration problems frequently prevent multi-camera systems from meeting the exact specifications and stringent accuracy standards required for surgical procedures. Such discrepancies can compromise the effectiveness and safety of medical procedures.

This deterioration in calibration emphasizes the need for sophisticated calibration methods. Coordinate Measuring Machines (CMM) are commonly used to improve calibration accuracy, offering a potential solution to this problem. Nonetheless, CMMs are expensive, and manufacturers typically calibrate the multi-camera system before delivery, failing to account for potential discrepancies after deployment. Traditional methods for detecting calibration errors, such as those based on epipolar

geometry, face significant computational challenges in multi-camera setups and do not support on-the-fly recalibration, making them ineffective in dynamic environments.

System Context and Problem Statement. This work addresses the calibration challenges inherent in infrared multi-camera systems designed for time-critical applications, specifically vision-based computer-assisted surgery. These systems function exclusively within a fixed distance from a central point of interest, allowing rotation while maintaining a constant radius to ensure comprehensive visual coverage. This operational design is very important for three main reasons: first, to comply with the manufacturer’s operating specifications and maintain system integrity and performance; second, to ensure that the area of interest is in-focus across all cameras for accurate tracking of medical instruments; and third, to minimize occlusions and reduce instances where tracked markers are not visible in all camera views. By customizing our approach to accommodate these multi-camera configurations and optimizing calibration for a fixed range of motion, we are able to significantly improve the accuracy and achieve the real-time calibration required for such time-critical applications.

In this work, we introduce a novel real-time multi-camera calibration method that leverages neural networks to provide on-the-fly recalibration. Our model is trained on synthesized camera poses resulting from OEM calibration parameters, with perturbations applied to the intrinsic and extrinsic parameters. The perturbations emulate real-world operational challenges, thereby enhancing the model’s practical applicability. We demonstrate, through rigorous experimentation, that our method not only adapts to alterations in calibration parameters in real time but also surpasses conventional calibration techniques in accuracy. Our key technical contributions are threefold:

- First, we introduce a real-time neural calibration method for multi-camera systems, marking a departure from traditional offline calibration methods. Our method employs a differentiable projection model to flow gradients between 3D geometries and their 2D projections, allowing for direct optimization of camera parameters.
- Second, we enhance the robustness and applicability of our method by dynamically synthesizing camera poses at each epoch and incorporating perturbations to the OEM calibration intrinsic and extrinsic parameters to simulate realistic operational challenges.

- Finally, we introduce two variants of our model: the first is designed for multi-camera systems equipped with onboard processing, directly outputting the 2D projections of the 3D fiducials; the second variant is designed for image-based multi-camera systems.

The work is structured as follows: Section 3.3 provides a brief overview of the pinhole camera model. Section 3.4 discusses our methodology, including the dynamic camera pose synthesis for multi-camera systems and details on the loss functions and training strategy. Finally, Section 4.2 reports on our experimental results.

3.2 Related Work

Camera calibration is fundamental in computer vision for determining geometric parameters essential for image capture, playing a vital role in applications that require accurate scene measurements. Despite the development of various methods to calculate camera parameters, traditional techniques like the Radial Distortion Model [10], Direct Linear Transform (DLT) [11], Tsai’s method [12], Zhang’s approach [13], and [14] rely heavily on handcrafted features and model assumptions. These methods, while effective, are often labor-intensive and not suited for real-time or multi-camera calibration due to their complexity and the static nature of their required setup. The perspective-n-point (PnP) problem has seen advancements with [15], [16], [17], and further improvements in [18] through direct optimization methods. However, these solutions still face challenges in terms of real-time execution.

The advent of artificial neural networks has prompted a significant shift in camera calibration research. Techniques like [19], [20], and PoseNet [21] for extrinsic calibration and others [22], [23], [24], [25] for intrinsic parameter estimation leverage deep learning for more adaptable and potentially real-time calibration. Yet, these neural network approaches require extensive datasets for training, have low accuracy, and cannot generalize well across varying conditions.

In multi-camera calibration, research has extended towards self-calibration methods for setups in shared environments, as seen in the works of Svoboda . [26], Heikkilä and Silvén [27], and also [28], [29], [30]. These methods facilitate 3D reconstruction and multi-view analysis but remain challenged by the demands of real-time processing and dynamic scenes. Recent developments

have also explored joint estimation of intrinsic and extrinsic parameters, as demonstrated by [31], [32], [33], [34]. These approaches promise more integrated calibration processes through deep learning, highlighting the potential for efficient real-time calibration. Nonetheless, achieving a balance between computational efficiency and accuracy remains a critical challenge for these advanced methods.

In contrast to the aforementioned techniques, our method enables real-time recalibration, effectively predicting camera parameters in the presence of perturbations to the OEM intrinsic calibration parameters. Additionally, a streamlined procedure for dynamic camera pose synthesis facilitates its generalization to arbitrary configurations of multi-camera systems and arbitrary calibration objects.

3.3 Camera Model

For the sake of completeness, this section provides an overview of the classic pinhole camera model, incorporating lens distortion to establish the relationship between the 3D world coordinates of a point \mathbf{P} and its 2D image projection $\mathbf{p} = [x, y]$. Consider $\mathbf{P} = [X, Y, Z]^T$ as a point in world coordinates, with \mathbf{R} and \mathbf{t} representing the camera’s rotation matrix and translation vector, respectively. The transformation of the 3D point into camera coordinates $\mathbf{P}^C = [X^C, Y^C, Z^C]^T$ is given by $\mathbf{P}^C = \mathbf{R}\mathbf{P} + \mathbf{t}$. The projection from 3D to 2D coordinates is given by:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{Z^C} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{P}^C, \quad (7)$$

where Z^C is the third component of \mathbf{P}^C , f_x and f_y the focal lengths, and c_x, c_y the principal point coordinates.

Lens Distortion. In order to accurately represent lens distortion, which commonly occurs in real camera systems, we include both the radial and tangential distortion. The distortion is represented using a polynomial model, where the radial distortion is captured by a sixth-order polynomial and the tangential distortion is addressed through first-order terms. The equations for the distorted

coordinates x_{dist} and y_{dist} are given as follows:

$$\begin{aligned} x_{\text{dist}} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2), \\ y_{\text{dist}} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y^2) + 2p_2 xy, \end{aligned} \quad (8)$$

where $r^2 = x^2 + y^2$, and k_1, k_2, k_3 are the radial distortion coefficients, and p_1, p_2 are the tangential distortion coefficients.

6D Rotation Parameterization. In line with the approach proposed in Zhou . [5], our work adopts a 6D parameterization for the representation of 3D rotations, diverging from traditional quaternion and Euler angle representations. The latter are known to introduce parameter space discontinuities, as depicted in Figure 3.1, complicating the learning process for neural networks due to the inherent discontinuities. Furthermore, a critical limitation of these traditional approaches is their inability to guarantee that the network outputs are orthogonal rotation matrices, which is often a desirable property for ensuring the physical plausibility of rotations in 3D space. These limitations are particularly pronounced in tasks requiring the learning of continuous and smooth rotation spaces. For the purpose of training, a 3D rotation matrix R is parameterized by a 6D vector r_{6D} , which encapsulates the elements of R 's first two columns. This parameterization provides a direct mapping to a rotation matrix without the need for complex conversions or the risk of introducing discontinuities. Specifically, during loss computation, the r_{6D} vector is seamlessly transformed back into an orthogonal rotation matrix R . This approach not only aligns with the findings from [5] that direct regression on 3x3 rotation matrices can lead to larger errors, but also addresses the need for generating orthogonal matrices directly from the network.

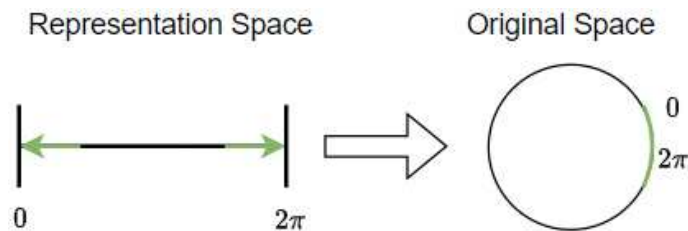


Figure 3.1: Discontinuities introduced by quaternions and Euler angle representations hinder network learning efficiency as shown in [5].

Differentiable Projection. The image formation process with the pinhole camera model is

designed to be differentiable, facilitating the backpropagation of gradients from the loss -a function of the difference between the observed and projected points- to the camera parameters.

3.4 Methodology

Our methodology employs a neural network designed to estimate the intrinsic and extrinsic parameters of a multi-camera system, leveraging a known calibration object in the scene. This tailored approach ensures that, once trained, the model is specifically attuned to the multi-camera system and calibration object utilized during its training phase. We invert the traditional image formation process, enabling our model to deduce camera parameters—such as rotations (R), translations (t), focal lengths (f_x, f_y), principal points (c_x, c_y), and distortion coefficients ($k_{1,2,3}, p_{1,2}$)—from 2D projections of 3D fiducial points on the calibration object.

At the core of our methodology is the dynamic camera pose synthesis, imperative for simulating realistic conditions that a multi-camera system might encounter. We introduce controlled perturbations into the OEM camera parameters, dynamically synthesizing diverse training samples. Given the synthesized camera parameters, the process starts with projecting the 3D fiducials of the calibration object onto the image plane. These 2D projected points serve as input to our model, which then predicts the intrinsics and extrinsics of the multi-camera system. Utilizing these predicted camera parameters, we project the known 3D fiducials back onto the image plane through a differentiable process. The deviation between the projections from the synthesized and predicted camera parameters is quantified using a loss function. Figure 3.2 provides an overview of our method.

3.4.1 Dynamic Camera Pose Synthesis

Dynamic camera pose synthesis begins with the OEM calibration parameters of a multi-camera setup, typically determined by the manufacturing process. We represent a multi-camera setup of N_C cameras as $C_i, 0 \leq i < N_C$, where C_i^{OEM} denotes the initial calibration of each camera.

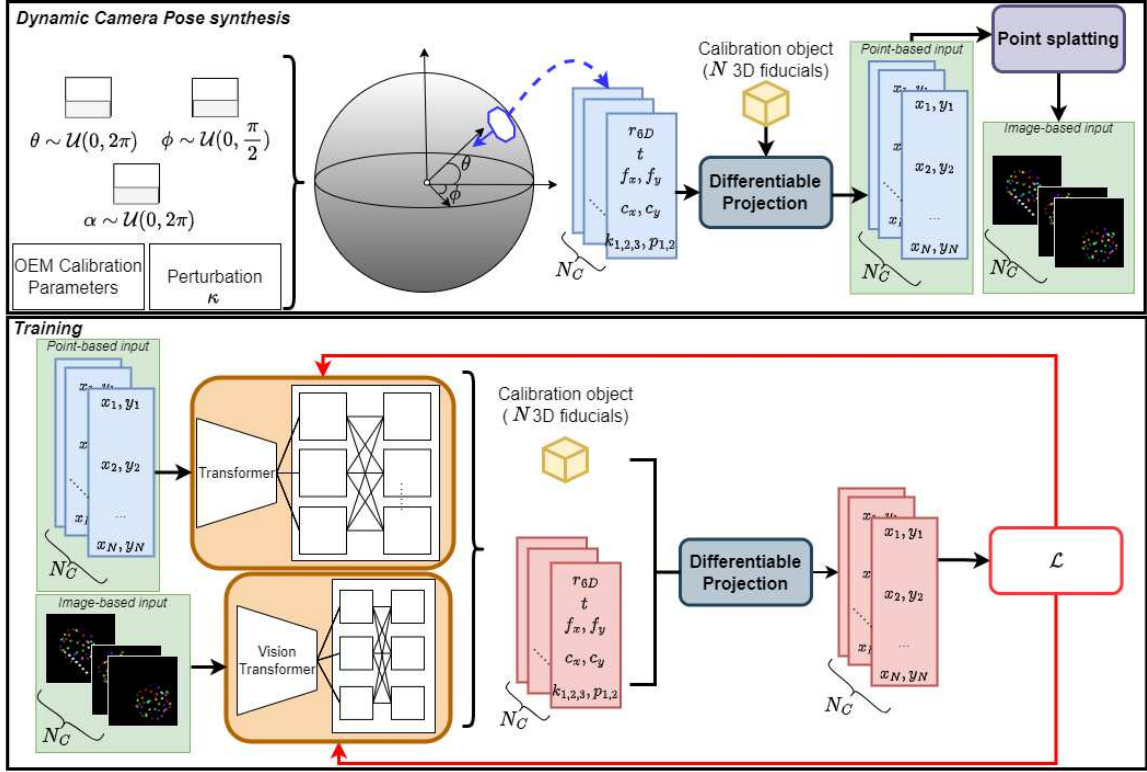


Figure 3.2: **Technical overview.** Our methodology begins with the synthesis of dynamic camera poses (see top fig.). Given spherical angles ϕ (azimuth), θ (elevation), along with the intrinsic rotation angle α , the OEM calibration parameters, the maximum perturbation limit κ , and known 3D fiducials (e.g. a cube calibration object), this module performs two primary functions: (i) it synthesizes poses for the multi-camera system, and (ii) it computes the projected 2D points. Subsequently, it employs point splatting to render images of these points. During training (see bottom fig.), the synthesizes poses and projected points (alternatively rendered images) are used to train the neural network. A differentiable projection ensures the propagation of gradients from the loss \mathcal{L} back to the predicted camera parameters.

Perturbations

To create a robust model capable of handling changes in calibration, we introduce perturbations to the OEM camera parameters, generating perturbed parameters $C_i^{pert} = C_i^{OEM} \times (1 + \delta)$, where δ represents the perturbation defined as $\delta = \kappa \times \mathcal{U}(0, 1)$, with κ controlling the maximum desired perturbation.

Camera Pose Synthesis

For the synthesis of camera poses, we employ a two-step process, ensuring both focus towards the fiducial points and variability in camera orientation to prevent model overfitting. The procedure

is as follows.




The centroid of the camera system is dynamically positioned on a hemisphere’s surface, ensuring varied perspectives. The radius of the hemisphere is determined in advance based on the application requirements and the range of motion of the multi-camera system. Specifically, the centroid’s position $P_{centroid}$ on the hemisphere is determined by:

$$P_{centroid} = (x_c, y_c, z_c) = \rho \cdot (\sin(\phi) \cdot \cos(\theta), \sin(\phi) \cdot \sin(\theta), \cos(\phi)), \quad (9)$$

where ρ denotes the hemisphere radius, and $\theta \sim \mathcal{U}(0, 2\pi)$, $\phi \sim \mathcal{U}(0, \pi/2)$ are angles sampled from a uniform distribution, ensuring the centroid is randomly positioned over the hemisphere.

Next, a rotation R_{focus} is applied to align the camera’s viewing direction towards the fiducials’ centroid, ensuring that the camera is oriented towards the area of interest. This alignment is critical for simulating realistic camera setups where the fiducials are within the camera’s field of view. To introduce additional randomness and prevent the network from overfitting to specific camera locations, a secondary random rotation R_{random} is applied by an intrinsic rotation angle $\alpha \sim \mathcal{U}(0, 2\pi)$ around the centroid point. The combination of R_{focus} and R_{random} ensures that each camera is not only oriented towards the fiducials but also positioned and rotated in a manner that provides a diverse set of viewing angles and positions. This diversity is imperative for training a robust model capable of generalizing across various camera orientations and positions.

As a final validation step, we conduct a visibility check to ensure all fiducials are within the field of view of all cameras. This step is essential, since the random placement of cameras on the hemisphere might result in scenarios where not all fiducials are visible from all cameras.

The dynamic camera pose synthesis for each training epoch diversifies the dataset, improving model generalization. Figure 3.3 illustrates synthesized camera poses for various multi-camera systems e.g. T-shape ($N_C = 4$ ) , U-shape ($N_C = 7$ ) , O-shape ($N_C = 10$ ) , and different calibration objects e.g. cube ($N_{fid.} = 8$), cube ($N_{fid.} = 27$), sphere ($N_{fid.} = 64$).

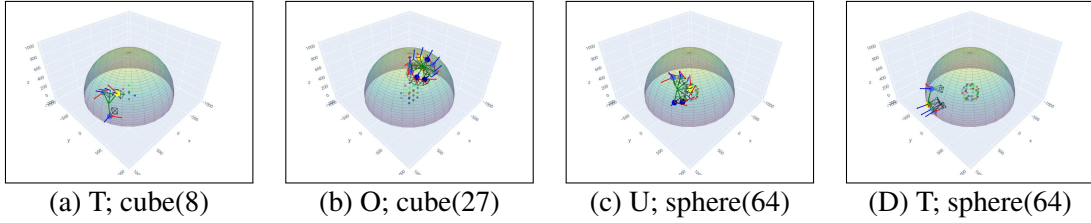


Figure 3.3: **Dynamic Camera Pose Synthesis.** Our framework supports arbitrary configurations of multiple cameras as well as a wide range of calibration objects. To synthesize camera poses, we employ a random uniform sampling strategy across three dimensions to ensure a comprehensive exploration of the pose space: azimuth (θ), elevation (ϕ), and roll (α), where $\theta \sim \mathcal{U}(0, 2\pi)$, $\phi \sim \mathcal{U}(0, \frac{\pi}{2})$, and $\alpha \sim \mathcal{U}(0, 2\pi)$. Additionally, Original Equipment Manufacturer (OEM) calibration parameters and a predefined maximum perturbation limit (κ) are incorporated.

3.4.2 Network Architecture

We introduce two variants: the first one, known as the point-based model (Pt), is intended for multi-camera systems with onboard processing that directly outputs the 2D projections of the 3D fiducials, which is the typical case for IR multi-camera systems; the second one is the image-based model (Img), designed for multi-camera systems that output images for subsequent processing, rather than the point coordinates. The architecture of both variants is detailed in the supplementary material along with an ablation study on the components.

3.4.3 Loss

Our loss function is crafted to encapsulate multiple facets of camera parameter estimation. The primary term \mathcal{L}_ϵ is incorporated to enforce geometric consistency. It measures the RMSE between 2D points x, x' obtained by projecting the 3D points using the predicted and ground-truth parameters, respectively, and is given by $\mathcal{L}_\epsilon = RMSE(x, x')$. For rotation, we use geodesic loss \mathcal{L}_{geo} , computed from the predicted and ground-truth rotation matrices, to penalize deviations in the orientation. Additionally, \mathcal{L}_{diff} is the root mean square error (RMSE) that quantifies the difference between the predicted and ground-truth parameters. We also employ a scaling value for the rotation and distortion coefficients since their values are very small compared to other camera parameters, as illustrated by the experiments in the ablations in the supplementary material.

3.4.4 Training

In the initial phase of 10,000 epochs, we utilize a simplified loss function focused on \mathcal{L}_{diff} and \mathcal{L}_{geo} to establish a robust baseline, formulated as $\mathcal{L} = \lambda_1 \times \mathcal{L}_{diff} + \lambda_1 \times \mathcal{L}_{geo}$. Subsequently, we introduce the full compound loss, expressed as $\mathcal{L} = \lambda_1 \times \mathcal{L}_{diff} + \lambda_1 \times \mathcal{L}_{geo} + \lambda_2 \times \mathcal{L}_\epsilon$. In all experiments, the coefficients are set to $\lambda_1 = 100$, $\lambda_2 = 0.01$, with the batch size of 512 and 8 for Pt and Img variants, respectively.

Our training strategy is based on introducing a random perturbation to the camera parameters in every epoch, to simulate a wide spectrum of deviations that may occur due to buildup of debris on critical components like the fiducials or lenses. This methodical addition of perturbation aids in better exploration of the parameter space and enhances the robustness and accuracy of the estimated camera parameters. We employ an Adam optimizer with parameter-specific learning rates. The rates are adaptively adjusted via a ReduceLROnPlateau scheduler based on the compound loss function. Furthermore, the backpropagation step employs gradient clipping for stability.

3.5 Experimental Results

There are no learning-free or neural techniques for *real-time* recalibration of *infrared multi-camera* systems. In the supplementary material, we adopt standard calibration and optimization techniques for infrared cameras and demonstrate the impracticality of using traditional methods for on-the-fly calibration. Below, we provide a comprehensive evaluation of our method’s accuracy across various scenarios, demonstrating its suitability for time-sensitive, high-accuracy applications. We conclude with an assessment of the generalizability of our approach to diverse multi-camera system configurations and calibration objects.































In Table 3.1, we detail the performance of our model under various conditions, reporting the average RMSE reprojection error RE_{avg}^{20K} across three trials on synthetic test sets, each comprising 20,000 data samples. Our model demonstrates robust adaptability to different levels of perturbations, where performance gracefully degrades as perturbations reach extreme values. For example, with an O-shape camera system comprising 10 cameras ($N_C = 10$ ) , and a calibration object with 8 fiducials ($N_{fid.} = 8$), our method achieves a RE_{avg}^{20K} of $9.93 \pm 1.4 \times 10^{-4}$, $11.53 \pm 1.5 \times 10^{-4}$,

Table 3.1: **Experimental results.** RE_{avg}^{20K} is the average RMSE reprojection error on 3 different trials (synthetic test sets, each comprising 20,000 data samples). Training on all models includes adding a **Max Perturb.** $\kappa \in [\text{min}\%, \text{max}\%]$ to the OEM camera intrinsic and extrinsic parameters respectively. $N_{fid.}$ and N_C are the number of 3D fiducials and the number of cameras, respectively. The rotation angle α remains the same in all experiments i.e., $\alpha \sim \mathcal{U}(0, 2\pi)$. The parameters for Pt and Img models are $\sim 33\text{m}$ and $\sim 86\text{m}$, respectively.

Row #	Variant	θ	ϕ	$N_{fid.}$ (object)	N_C (config.)	Max Perturb. $\kappa_{int.}, \kappa_{ext.} \in [\text{min}\%, \text{max}\%]$	RE_{avg}^{20K} (pixels)
1	Pt	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	8 	10 	$\pm 2.5, \pm 2.5$	$9.93 \pm 1.4 \times 10^{-4}$
2	Pt	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	8 	10 	$\pm 5, \pm 5$	$11.53 \pm 1.5 \times 10^{-4}$
3	Pt	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	8 	10 	$\pm 10, \pm 10$	$14.21 \pm 1.8 \times 10^{-4}$
4	Pt	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	8 	7 	$\pm 2.5, \pm 2.5$	$14.67 \pm 3.5 \times 10^{-4}$
5	Pt	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	8 	4 	$\pm 2.5, \pm 2.5$	$12.89 \pm 2.1 \times 10^{-4}$
6	Pt	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	64 	10 	$\pm 2.5, \pm 2.5$	$14.92 \pm 7.3 \times 10^{-5}$
7	Pt	0	0	8 	6 	0, 0	$1.12 \pm 5 \times 10^{-7}$
8	Pt	0	0	8 	6 	$\pm 2.5, 0$	$3.08 \pm 9.5 \times 10^{-6}$
9	Pt	0	0	8 	6 	$\pm 10, 0$	$8.04 \pm 7.2 \times 10^{-4}$
10	Img	0	0	8 	6 	$\pm 2.5, 0$	$4.55 \pm 1.8 \times 10^{-4}$
11	Img	0	0	8 	10 	$\pm 2.5, \pm 2.5$	$4.12 \pm 2.4 \times 10^{-6}$
12	Img	0	0	8 	10 	$\pm 5, \pm 5$	$6.01 \pm 1.6 \times 10^{-5}$
13	Img	0	0	8 	10 	$\pm 10, \pm 10$	$6.97 \pm 1.1 \times 10^{-5}$

and $14.21 \pm 1.8 \times 10^{-4}$, for perturbations of up to 5%, 10%, 20%, respectively, as shown in rows 1-3. As we explain in the subsequent section, the range of motion is often constrained in real-world scenarios due to operational limitations. Nevertheless, our approach demonstrates robust performance and effectively predicts camera poses across a wide range of motion, i.e., $\theta \sim \mathcal{U}(0, 2\pi)$, $\phi \sim \mathcal{U}(0, \pi/2)$, and $\alpha \sim \mathcal{U}(0, 2\pi)$.


3.5.1 Generalization to Arbitrary Configurations & Calibration Objects

We conducted comprehensive experiments with various camera configurations and calibration objects to evaluate the generalization of our method. Cameras were positioned in geometric configurations common to real-world scenarios, such as the O-shaped ($N_C = 10$ ), U-shaped ($N_C = 7$ ) and T-shaped ($N_C = 4$ ) arrangements. As shown in Table 3.1 (in rows 1, 4, and 5), even with a small number of fiducials i.e., $N_{fid.} = 8$, the reprojection errors RE_{avg}^{20K} are low with $9.93 \pm 1.4 \times 10^{-4}$, $14.67 \pm 3.5 \times 10^{-4}$, $12.89 \pm 2.1 \times 10^{-4}$, for the O-shape, U-shape, and T-shape configurations, respectively.

The versatility of our approach is further demonstrated through tests involving calibration objects with different shapes and numbers of fiducials. Specifically, we evaluated our model using a

calibration cube with $N_{fid.} = 8$ and a calibration sphere with $N_{fid.} = 64$ fiducials. As shown in Table 3.1 (rows 1, 6), our method maintains a consistent error profile for both calibration objects.

3.5.2 Comparison with CMM-calibrated multi-camera system.

As previously stated, it is important to recognize that regressing camera poses under the conditions that $\theta \sim \mathcal{U}(0, 2\pi)$, $\phi \sim \mathcal{U}(0, \pi/2)$ is more challenging than in typical real-world scenarios where multi-camera systems have predefined operational ranges for θ and ϕ . Our method, when practically tested in a surgical setting using a multi-camera system with a predefined operational range of motion (specifically, an overhead fixed O-shaped system ($N_C = 6$ )) where $\theta = \phi = 0$, and a calibration object with $N_{fid.} = 8$), demonstrated superior performance. This system was previously calibrated (offline) using a high-end CMM. Our approach surpassed the CMM calibration in terms of reprojection error within this motion range. CMM calibration parameters *without perturbations* led to an average reprojection error of 1.80 pixels. In contrast, our method led to an average reprojection error of $1.12 \pm 5 \times 10^{-7}$ on a model trained *without perturbations* (Table 3.1; row 7), and an average reprojection error of $3.08 \pm 9.5 \times 10^{-6}$ (Table 3.1; row 8) and $8.04 \pm 7.2 \times 10^{-4}$ (Table 3.1; row 9) for *perturbations of up to 5% and 20%*, respectively. Additionally, with an average inference time of 0.0026 seconds on a system equipped with a Nvidia RTX 4090 GPU, our method proves capable of supporting real-time applications.

3.6 Conclusion

Our work presents a neural calibration method tailored for the real-time and adaptive calibration of multi-camera systems. Central to our approach is the combination of dynamic camera pose synthesis with a differentiable projection model, which facilitates the direct optimization of camera parameters from image data. Comprehensive experimental analysis demonstrated the robustness of the method and its capacity to accurately predict calibration parameters while accommodating random perturbations.

We further elaborated on the practicality of our method, contrasting the impracticality of recalibrating or optimizing camera parameters at each step in real-time applications. Our evaluations

revealed our method’s superior accuracy and generalizability across different scenarios. The introduction of two variants—catering to systems with either direct 2D projections of 3D fiducials or color-coded 2D projected points—proves our method’s flexibility and broad applicability in diverse operational contexts. Future research directions include the integration of perturbations into the extrinsic parameters of the camera models. This holds the potential to significantly broaden the applicability of our approach in more dynamically varied environments.

Chapter 4

Supplementary Material

The following is a verbatim copy of the manuscript submitted as supplementary material for the paper titled "Neural Real-Time Recalibration for Infrared Multi-Camera Systems".

Abstract

In the supplementary material, we present details on the network architectures and an ablation on their components, comparisons with standard calibration and optimization techniques, qualitative results and training nuances of the real-time neural multi-camera system calibration method. Additionally, through a series of visualizations, we illustrate the precision of our model in predicting camera poses against ground truth, with an emphasis on robustness in the presence of perturbations. Lastly, we address the limitations and discussions.

4.1 Network Architecture

We introduce two variants: the first one, known as the point-based model (Pt), is intended for multi-camera systems with onboard processing that directly outputs the 2D projections of the 3D fiducials, which is the typical case for IR multi-camera systems; the second one is the image-based model (Img), designed for multi-camera systems that output images for subsequent processing, rather than the point coordinates. The architecture of both variants is summarized in the supplementary material.

4.1.1 Point-based variant.

Given an input tensor $X \in \mathbb{R}^{N_C \times N_{fid.} \times 2}$, where N_C is the number of cameras, and $N_{fid.}$ is the number of fiducials, the point-based model performs the following operations:

- (1) **Embedding Layer** f_{embed} : Maps input fiducial points to a higher-dimensional space: $X_{enc} = f_{embed}(X)$, where $X_{enc} \in \mathbb{R}^{N_C \times 512}$.
- (2) **Camera Identity Encoding** CIE : Incorporates camera-specific information: $X'_{enc} = X_{enc} + CIE(N_C, 512)$, where $X'_{enc} \in \mathbb{R}^{N_C \times 512}$.
- (3) **Transformer Encoder** f_{TE} : Processes embeddings through a transformer encoder to model complex relationships: $X''_{enc} = f_{TE}(X'_{enc})$, $X''_{enc} \in \mathbb{R}^{N_C \times 512}$.
- (4) **Downstream Heads** h_*^M : Process the encoder output through separate prediction heads for different outputs $X_{out}^* \in \mathbb{R}^M$ (rotation r_{6D} , translation t , focal lengths $fc = (f_x, f_y)$, principal point $pp = (c_x, c_y)$, and distortion coefficients $kc = (kc_{1,2,3}, p_{1,2})$):

$$\begin{aligned} X_{out}^{r_{6D}} &= h_R^6(X''_{enc}), & X_{out}^t &= h_t^3(X''_{enc}), \\ X_{out}^{fc} &= h_{fc}^2(X''_{enc}), & X_{out}^{pp} &= h_{pp}^2(X''_{enc}), & X_{out}^{kc} &= h_{kc}^5(X''_{enc}) \end{aligned}$$

- (5) In a final step, the outputs X_{out}^* are combined to form the output tensor \hat{X}_{out} representing the combined camera parameters:

$$\hat{X}_{out} = (\gamma(X_{out}^{r_{6D}}), X_{out}^t, X_{out}^{fc}, X_{out}^{pp}, X_{out}^{kc})$$

where $\hat{X}_{out} \in \mathbb{R}^{N_C \times 21}$, $\gamma(\cdot)$ is a function that expands the 6D parameterized rotation to a rotation matrix as described in Section 3 in the main paper, and 21 is the number of calibration parameters per camera.

Incorporating camera identity encoding CIE into our architecture significantly improves performance by disambiguating camera perspectives, as shown in the ablations in Section 4.1.3. The CIE applies a simple one-hot encoding strategy where each camera is assigned a unique identifier

in this embedding space. To accommodate scenarios where the number of cameras exceeds the embedding dimension, it introduces a small amount of noise to each encoding, ensuring that each camera’s encoding remains distinct. This enhances robustness and accuracy for spatially-aware tasks and supports generalization to new configurations and arbitrary camera setups.

4.1.2 Image-based variant.

The image-based variant utilizes a Vision Transformer (ViT) to process the input image tensor $X_{in} \in \mathbb{R}^{N_C \times Ch \times H \times W}$ and predict camera parameters.

- (1) **Encoder** $f_{encoder}$. The input image tensor is resized to 224×224 pixels and passed through a ViT encoder. The ViT model is adapted by replacing its classification head with an identity layer, allowing the model to output a feature representation directly $X_{enc} = f_{encoder}(X_{in})$, where $X_{in} \in \mathbb{R}^{N \times 3 \times 224 \times 224}$, and $X_{enc} \in \mathbb{R}^{N_C \times 768}$.
- (2) **Bottleneck Layer** $f_{bottleneck}$. The encoded features are further processed through a bottleneck layer to reduce dimensionality and focus on relevant features for parameter prediction: $X_{bottleneck} = f_{bottleneck}(X_{enc})$, and comprises a linear layer, reducing features to $\mathbb{R}^{N_C \times 128}$.
- (3) **Downstream Heads**: Similar to the point-based model variant, separate heads are used for predicting the camera parameters from the bottleneck features $X_{bottleneck}$, and combined to form the output tensor $\hat{X}_{out} \in \mathbb{R}^{N_C \times 21}$.

4.1.3 Ablation Study

We conducted an ablation study to demonstrate the impact and significance of individual components of our network. All models have been trained following the training strategy in Section 4.4 in the main paper, given the conditions explained in Table 4.1.

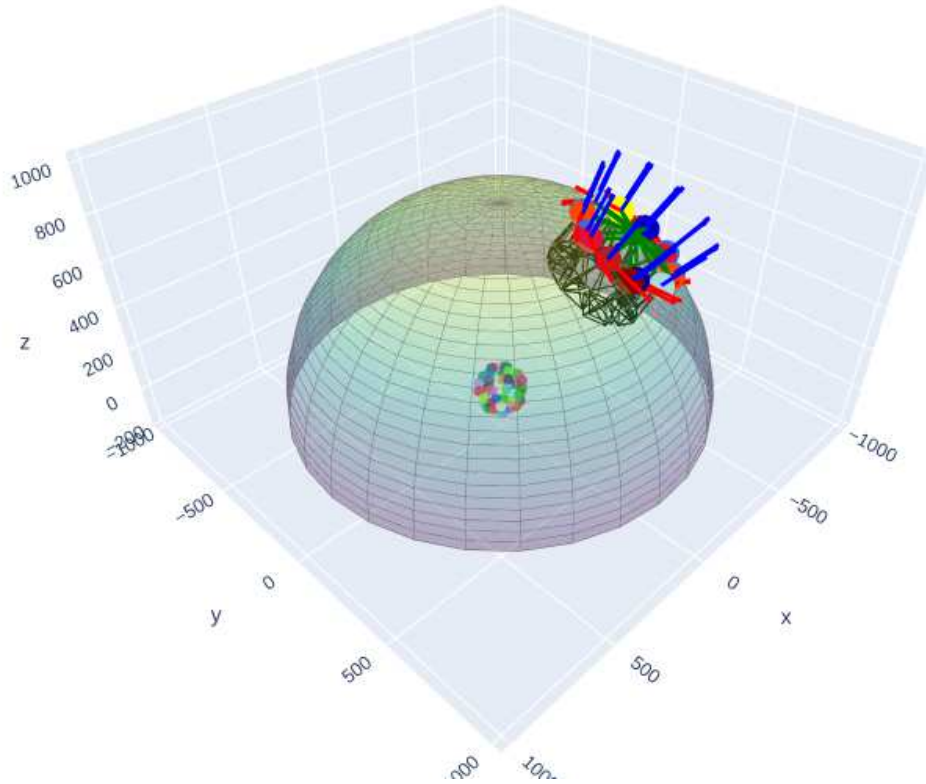


Figure 4.1: **Visualization of predicted vs ground truth camera poses.** The calibration object is a sphere with 64 fiducials. The multi-camera system configuration is O-shaped comprising 10 cameras. For closer inspection please refer to the interactive visualization in the **cameras.html** file.

Impact of Value Scaling

We highlight the significant role of scaling rotation and distortion coefficients prior to RMSE loss calculation. Given the inherently small magnitudes of these coefficients—elements of the rotation matrix ($0 < r_i < 1$) and distortion parameters ($k_{c_{1,2,3}}, p_{1,2} \lesssim 0.1$)—their contribution to the overall loss is minimal. Introducing a scaling factor, λ_{scale} , amplified their effect, as evidenced by the increased prediction errors observed when omitting this factor, detailed in Table 4.1.

Camera Identity Encoding *CIE*

We showcase the significance of Camera Identity Encoding (*CIE*). The results in Table 4.1 clearly indicate that incorporating *CIE* leads to notable performance improvements. By integrating *CIE*, our model distinctly differentiates between camera perspectives, leading to marked improvements in accuracy for spatially-aware tasks.

Table 4.1: **Ablation study results.** Training on all models includes a perturbation of **Max Perturb.** $\kappa \in [-2.5\%, +2.5\%]$ of the OEM intrinsic parameters, utilizing 10 cameras ($N_C = 10$) and a calibration object with 8 fiducials ($N_{fid.} = 8$). Pt models have been trained for 300k epochs and Img models for 150k epochs.

Row #	Variant	# Params	θ	ϕ	Encoder	Scaled $\lambda_{scale} = 1000$	Camera Identity Encoding CIE	RE_{avg}^{20K} (pixels)
1	Pt	$\sim 33m$	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	Transformer	✓	✓	$12.47 \pm 6 \times 10^{-4}$
2	Pt	$\sim 33m$	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	Transformer	✓	✓	$421.26 \pm 9.6 \times 10^{-3}$
3	Pt	$\sim 33m$	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	Transformer	✓	-	$25.49 \pm 2 \times 10^{-2}$
4	Pt	$\sim 8m$	$\sim \mathcal{U}(0, 2\pi)$	$\sim \mathcal{U}(0, \pi/2)$	CNN-1D	✓	✓	$96.93 \pm 3.9 \times 10^{-2}$
5	Img	$\sim 86m$	0	0	ViT	✓	-	$6.89 \pm 5.8 \times 10^{-5}$
6	Img	$\sim 24m$	0	0	CNN-2D	✓	-	$20.62 \pm 2.8 \times 10^{-5}$

Impact of Encoder

We contrast the performance impact between utilizing a transformer encoder and a 1D convolutional (CNN-1D) encoder within our model. As detailed in Table 4.1, the integration of a transformer encoder significantly enhances model accuracy. Similarly, for the image-based model, the ViT encoder results in increased precision when compared with CNN-2D.

4.2 Comparison with Standard Calibration and Optimization Techniques

In this section, we explore the differences between recalibrating and optimizing infrared cameras at every step in real-time applications. We start our discussion in Section 4.2.1, examining the drawbacks of recalibrating at each step using standard methods. Through experiments, we demonstrate the significant time required for this process, thereby illustrating the impracticality of using traditional methods for on-the-fly calibration in multi-camera systems. Following this, Section 4.2.2 presents experiments that emphasize the difficulties in minimizing reprojection errors using Bundle Adjustment (BA).

4.2.1 Inference vs. Recalibration

In this section, we address the limitations of traditional calibration methods. Traditional calibration methods require a large number of images, especially for applications demanding high

accuracy. In such scenarios, the requirement can escalate to thousands of images, leading to a prohibitive increase in execution time (i.e., > 1000 images for vision-based computer-assisted surgical multi-camera systems). Even in the hypothetical scenario where capturing thousands of images instantaneously were possible, the calibration process itself remains time-consuming and scales poorly as image quantity increases. As depicted in Figure 4.2 for a single camera, this execution time grows exponentially with the number of images, rendering a recalibration at every step impractical within real-time applications. Additionally, increasing the number of iterations in the Levenberg-Marquardt refinement step leads to a linear time increase, as illustrated by the blue curve in Figure 4.2.

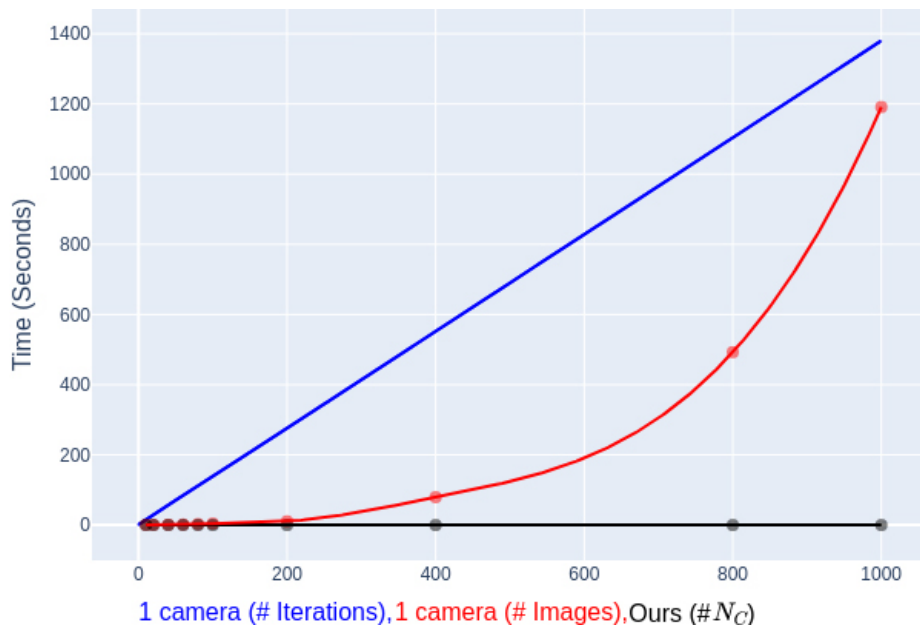


Figure 4.2: **Runtime for traditional camera calibration.** Exponential growth in calibration time with increasing number of images (red; 1 camera). Linear increase w.r.t. LM iterations on 100 images (blue; 1 camera). Ours; real-time ($\tau_{min}^{i=1} = 0.0026s$, $\tau_{max}^{i=10} = 0.012s$) for increasing number of cameras 2^i , $1 \leq i \leq 10$ (black).

In contrast, our method operates on a fixed number of cameras, N_C , which inherently caps the inference time to $\tau_{min}^{i=1} = 0.0026s$, $\tau_{max}^{i=10} = 0.012s$ for increasing number of cameras 2^i , $1 \leq i \leq 10$ on an Nvidia RTX 4090, as shown in Figure 4.2 by the black curve. This translates to ~ 387 inferences per second, demonstrating a significant advantage in the context of real-time applications.

4.2.2 Inference vs. Minimizing Reprojection Errors

We now pivot our experiments to investigate the feasibility of minimizing reprojection error at every step using Bundle Adjustment (BA) as an alternative to recalibrating at every step, in the context of real-time optimization amidst decalibration.

We conducted 1000 trials, each applying random perturbations within a range of $\kappa \in [-10\%, 10\%]$ to the OEM intrinsic parameters. For every trial, our objective was to minimize the reprojection error with BA, utilizing data from $N_{fid.} = 8$ points and $N_C = 6$ cameras.

Our findings highlight a significant constraint: the time taken for a single BA iteration, averaging 0.0385 seconds, substantially exceeds our execution time of 0.0026 seconds for a single inference, and proves that optimizing camera parameters by minimizing reprojection errors at every step is not a viable choice for real-time applications.

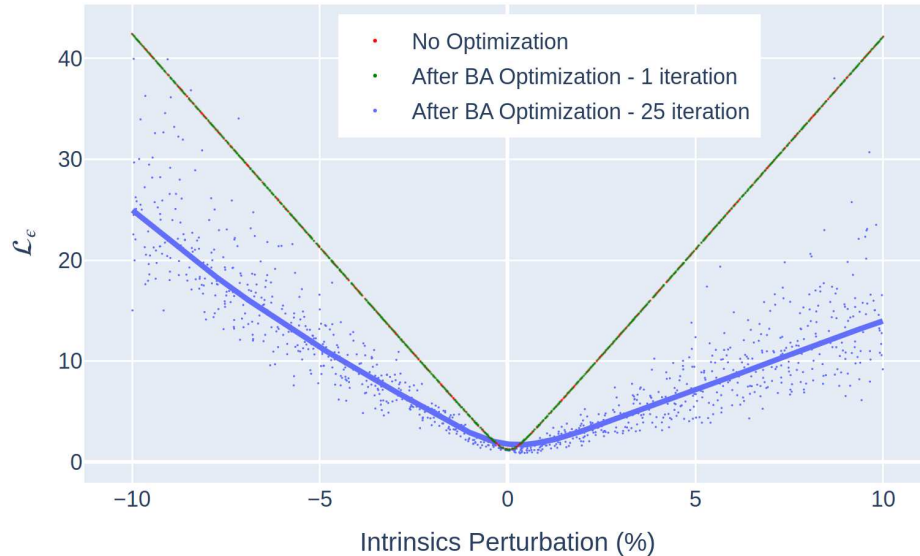


Figure 4.3: **Effect of decalibration on reprojection error (RMSE).** OEM intrinsic parameters are perturbed by 20%, i.e., $\kappa \in [-10\%, 10\%]$, simulating potential decalibration. Reprojection error without intervention (red), with 1 iteration of Bundle Adjustment (green), with 25 iterations (blue); 1000 trials.

Furthermore, a single iteration of BA resulted in minimal improvement in reprojection error, as illustrated in Figure 4.3 in green vs red. Subsequent experiments showed that achieving a noticeable reduction in reprojection error required an average of 25 BA iterations per system’s capture, culminating in an untenable average execution time of 1.2892 seconds for real-time processing. Despite

these efforts, the reprojection error remained significantly high (Figure 4.3; blue), compared to our method (Table 1 in the main paper; rows 9, 10).

4.3 Qualitative Results

In Figure 4.1, we show a qualitative result of the calibration from the multi-camera system calibration. The blue-shaded spheres represent the ground truth values. The red-shaded spheres represent the predicted values. The single yellow sphere per multi-camera system represents the first camera, and is singled out for visualization purposes only, to demonstrate the in place rotation R_{random} i.e. that the multi-camera system does not always have the same vertical orientation. For closer inspection please refer to the interactive visualization in the supplementary material.

Figure 4.5 illustrates the reprojection accuracy of 3D fiducials using predicted camera poses (white dots) contrasted against the ground-truth projections (color-coded). In this example, a point-based model was trained with 5% perturbation, and the input points to the model were generated with *no perturbation* to the camera parameters. The multi-camera system is arranged in an O-shape, consisting of 10 cameras, with a cube with 8 fiducials as the calibration object.

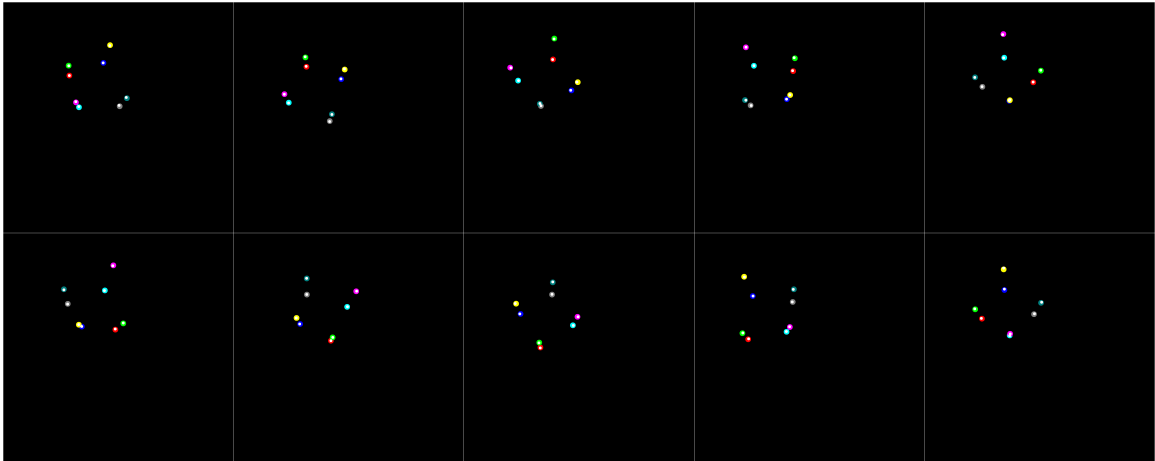


Figure 4.4: **Reprojection of 3D Fiducials with Predicted Camera Poses (With 5% Perturbation)**. This figure illustrates the resilience and accuracy of our pose estimation model during decalibration of the camera parameters of up to 5% perturbation. The calibration cube and the O-shaped arrangement of 10 cameras remain constant as in Figure 4.5, allowing for a direct comparison across different testing conditions. Note: The reprojected points are shown in white. For optimal visibility, please zoom in.

Figure 4.4 shows the same model as Figure 4.5, i.e., trained with a 5% perturbation. However,

in this example we simulate a decalibration by *generating the test data from perturbed camera parameters of up to 5%*. Beyond the quantitative evaluation presented in the previous section, these figures facilitate a direct qualitative comparison of model performance and clearly demonstrate the robustness of our pose estimation technique under conditions of uncertainty.

4.4 Training: Less Effective Strategies

In this section, we describe our exploration of various loss and regularization terms intended to improve model performance. Despite their theoretical potential, these methods did not enhance our results in practice.

4.4.1 Regularization terms

- A regularization term, denoted as \mathcal{L}_{κ_c} , was introduced to encourage minimal distortion coefficients (k_1, k_2, k_3, k_4, k_5) by minimizing the sum of the absolute values of these coefficients.
- A regularization term, \mathcal{L}_{f_c} , was designed to ensure consistency in the focal lengths f_x and f_y by minimizing the squared difference between these focal lengths.

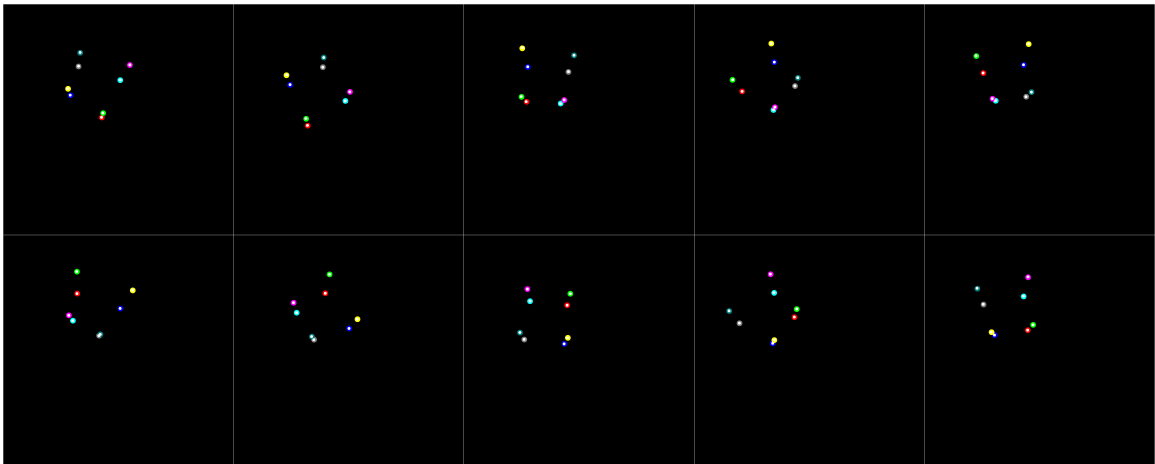


Figure 4.5: **Reprojection of 3D Fiducials with Predicted Camera Poses (Test Data Generated Without Perturbation)**. This figure demonstrates the reprojection accuracy in a O-shaped multi-camera setup comprising 10 cameras, with the calibration object being a cube with 8 fiducials. The comparison between the predicted (in white) and ground-truth (color-coded; enlarged for visualization purposes) projections demonstrates the precision of our model in the absence of perturbation. Note: The reprojected points are shown in white. For optimal visibility, please zoom in.

- The regularization term \mathcal{L}_{pp} aimed to align the principal points (c_x, c_y) with the image center by minimizing the squared Euclidean distance between the principal points and the center of the image.
- A regularization term, $\mathcal{L}_{largefc}$, is formulated to discourage small focal lengths by imposing a penalty on the inverse of the sum of the focal lengths along both the x and y axes.
- Applying an L1 norm regularization to our model’s parameters aimed to encourage sparsity and reduce overfitting by penalizing large weights, thus simplifying the model. This technique, intended to improve generalization, unfortunately did not yield the expected performance enhancements in our camera parameter prediction task.

4.4.2 Losses

We investigated the log-cosh loss function, attracted by its theoretical benefits such as smooth gradients, outlier robustness, and balanced error sensitivity. These characteristics suggested that log-cosh could enhance prediction precision and stability. The log-cosh loss is given by,

$$\mathcal{L}_{logcosh} = \frac{1}{N} \sum_{i=1}^N ((y_i - \hat{y}_i) \tanh(y_i - \hat{y}_i) - \log(2) + \log(1 + e^{-2|y_i - \hat{y}_i|}))$$

where y_i represents the target values, and \hat{y}_i represents the predicted values. However, contrary to our expectations, empirical testing revealed that log-cosh loss did not outperform our existing loss function described in Section 4.3 in the main paper.

Figure 4.6 illustrates a key observation: minimizing our proposed final loss also reduces the auxiliary losses we initially investigated-except from the L1 norm which relates to the network weights rather than the calibration error, and whose purpose is to enforce sparsity. However, this effect is not reciprocal; directly incorporating these auxiliary terms into our model did not further enhance performance beyond the improvements achieved with the final loss alone described earlier. This indicates that while our final loss effectively captures the essence of the auxiliary losses, adding them explicitly does not provide additional benefits. In the figure, we present the training loss

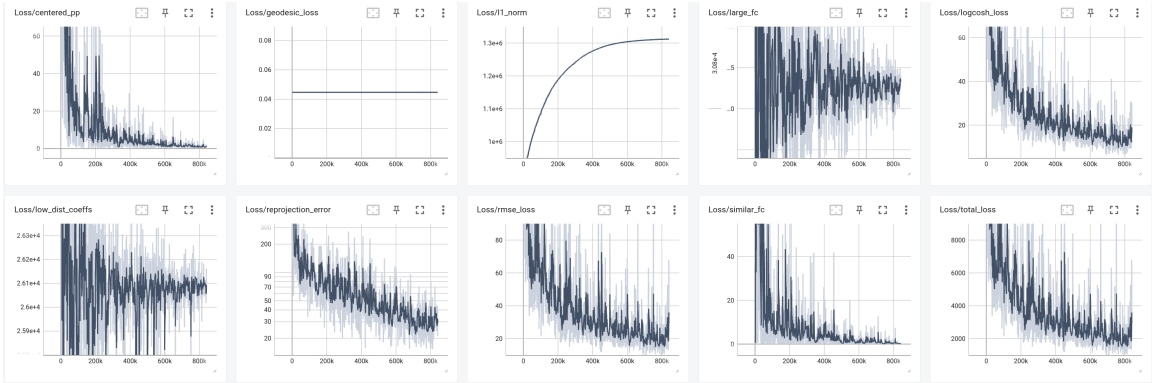


Figure 4.6: **Illustration of the proposed loss compared to auxiliary regularization and loss terms.** This figure demonstrates that minimizing the proposed loss leads to a reduction in auxiliary losses. However, this is a non-reciprocal relationship where the inclusion of auxiliary losses does not further improve model performance beyond the capabilities of the proposed loss which is a combination of the reprojection error, the RMSE error of the parameters, and the geodesic error. Here, we present the training loss graphs for the point-based variant. As previously described, the distortion coefficients are scaled by scaling factor $\lambda_{scale} = 1000$.

patterns for the point-based variant. Comparable trends are observed with the image-based variant.

4.5 Limitations & Discussion

4.5.1 Adhering to Operational Specifications

Our camera pose synthesis methodology is specifically designed to meet the manufacturer’s usage requirements, prioritizing high-accuracy predictions in real-world applications, particularly in time-critical contexts such as surgical applications. An important component of our methodology involves simulating a field of view that ensures comprehensive coverage of the area of interest, closely mimicking the actual setup. This is accomplished by ensuring a constant radius for the hemisphere, in accordance with the operational guidelines of the multi-camera system. Unlike some multi-camera systems that are statically affixed with no or limited range of motion, our methodology introduces a significant enhancement by allowing an extended range of motion—albeit at a fixed distance—over the entire hemisphere. The decision to adhere to the specified fixed radius is a deliberate design choice aimed at ensuring that our synthesized camera poses accurately reflect the real-world configuration.

4.5.2 Limited Range of Motion for Image-based

Our assessment of the image-based variant shows promising results in multi-camera setups having a limited range of motion. However, to achieve broader generalization across diverse vantage points -especially at glazing angles-, a more advanced architecture is imperative. This is based on the observation that accuracy drops when using the full range of motion on the hemisphere, attributed to the complexity of input images. Specifically, the challenge arises as only a fraction, N_{fid} pixels, are relevant to the task amidst the backdrop of the remaining $H \times W - N_{fid}$ pixels, which significantly narrows the dataset’s utility for network learning.

4.5.3 Model Customization

Our methodology demonstrates a tailored approach, achieving high accuracy in predicting camera parameters when applied to the particular multi-camera system, calibration object, and operating distance utilized during its training phase. This high level of accuracy reflects how well the method is customized to fit its specific training conditions, including the perturbations, up to the extent the model has been trained to withstand. However, it is important to note that the model’s performance is finely tuned to this particular setup. Any deviation from the original camera setup, use of a different calibration object, or change in the desired maximum perturbation level requires the training of a new model tailored to those new conditions.

4.5.4 Decalibration Detection

Our method also extends beyond real-time calibration to effectively identify decalibration. By continuously comparing the predicted calibration parameters in real-time with those of the OEM, one can swiftly identify any deviations indicative of decalibration. This dual functionality positions our method as both a calibration tool and an operational integrity monitor, ensuring continuous accuracy and reliability of multi-camera systems through early detection and prompt recalibration response.

4.6 Model Training Progression Animation

Included in the supplementary materials is a fast-forward animation, which illustrates the evolution of our network’s learning process throughout the training phase. This animation presents key stages of the network’s training progression at intervals of every 20 epochs, for a total of 50K epochs.

Chapter 5

Extension to Fisheye Camera Model

In this chapter, we first show an extension of the work presented in Chapters 3 and 4, which is the fisheye camera model training and test results. Later we explore an alternative approach to extract 2D points from a sequence of images, captured from a scene containing 8 fiducials in 3D space, with the goal of tracking them consistently as the camera moves around the scene. This will ensure a reliable method for identifying the key points in our image sequence. Finally, we discuss other metrics used for the evaluation of the calibrated camera setups, which can be used besides the usual reconstruction and reprojection error.

5.1 Fisheye Camera Model Training and Test Results

While the work in Chapter 3 shows the result for multi-camera calibration for the pinhole camera model, here we tried to achieve similar results for the fisheye camera model. As discussed in Chapter 2, the fisheye camera model differs in the way it distorts the projected points so that we get a wider -close to 180-degree- field of view. Figure 5.1 is a sample image captured from a scene with 6 cameras with fisheye lens and 8 fiducials in 3D space. A fisheye lens creates a wide-angle view with significant barrel distortion, causing straight lines to appear curved and objects near the edges to be stretched while compressing the center, resulting in an exaggerated perspective.

For this experiment, we had to modify the projection function and other related components of



Figure 5.1: Synthetic image of the Fisheye camera setup, with 6 cameras and 8 key points.

the code, as well as the multi-camera configuration file, to train for such a setup. The training process converges delicately as depicted in figure 5.2 for the training with up to 5% perturbation. Other training strategies remain unchanged and similar to the pinhole camera model training process.

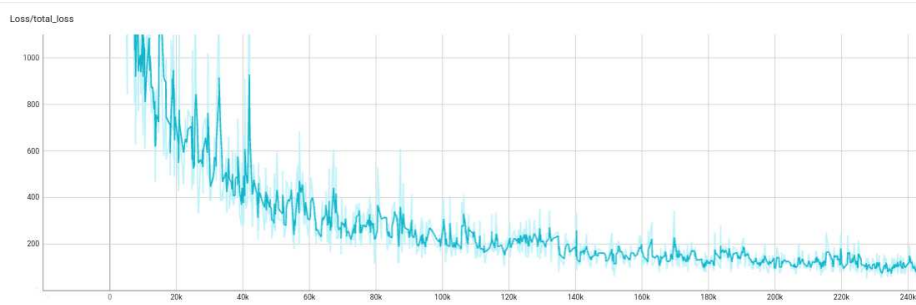


Figure 5.2: The training loss of the fisheye camera model with 5% perturbation.

Finally, we conducted several tests on 20000 samples with different levels of perturbation which is shown in table 5.1. In these experiments, the number of cameras is equal to 10 and there are 8 fiducials captured in the common field of view of those cameras. This shows the capability of the proposed model and solution to support regressing the camera parameters of a camera setup with the fisheye lens, achieving a comparable error rate as the pinhole camera model.

Table 5.1: **Experimental Results:** $\mathbf{RE}_{camera}^{20K}$ is the average RMSE reprojection error on 3 different trials (synthetic test sets, each comprising 20,000 data samples). Training on all models includes adding a **Max Perturb.** $\kappa \in [\mathbf{min}\%, \mathbf{max}\%]$ to the OEM camera intrinsic and extrinsic parameters respectively. The number of 3D fiducials and the number of cameras are 8 and 10 respectively in these experiments. The rotation angle α remains the same in all experiments i.e., $\alpha \sim \mathcal{U}(0, 2\pi)$.

Max Perturb. $\kappa_{int.}, \kappa_{ext.} \in [\mathbf{min}\%, \mathbf{max}\%]$	$\mathbf{RE}_{Pinhole}^{20K}$ (pixels)	$\mathbf{RE}_{Fisheye}^{20K}$ (pixels)
2.5	$4.12 \pm 2.4 \times 10^{-6}$	$5.50 \pm 3.2 \times 10^{-6}$
5	$6.01 \pm 1.6 \times 10^{-5}$	$6.43 \pm 3.4 \times 10^{-5}$
10	$6.97 \pm 1.1 \times 10^{-5}$	$7.20 \pm 1.5 \times 10^{-5}$

5.2 Detection and Tracking of Bright Points in Image Sequences

In this section, we present a method for image processing, detecting and tracking a set of bright points across a series of images. This is a mandatory step for a special case we encountered with IR emitters on an IR multi-camera setup. This robust and efficient method enables precise tracking of bright points in various applications, including medical setups. The main steps involved in this process are as follows:

5.2.1 Image Preprocessing, Blob Detection and Gaussian Fitting

Each image is converted to grayscale, and a threshold is applied to isolate the brightest regions, which correspond to the points of interest. This binary image highlights the potential bright points as distinct blobs. The connected components (blobs) in the binary image are then labeled, with each blob representing a candidate point. For each detected blob, a two-dimensional Gaussian function is fitted to the brightness distribution. This involves first estimating the initial parameters of the Gaussian and then applying a fitting algorithm, such as the Levenberg-Marquardt optimization, to refine the estimated parameters. This fitting improves the precision of the bright point's location to sub-pixel accuracy, which is essential for accurate tracking.

5.2.2 Point Tracking Across Images

The identified points are tracked across consecutive images by matching them based on their spatial proximity. This ensures that each bright point is consistently identified throughout the image sequence, allowing for accurate tracking. The sampling rate is generally constant during image data capture, providing a suitable environment for applying point matching algorithms, such as the simple nearest neighbor approach, while effectively detecting and discarding any instantaneous movements of the setup. This step is crucial for establishing the 2D-3D correspondence of key points in the camera calibration process.

5.3 Evaluation Metrics

Inspired by the design of real markers in our tests, we explore the case with four fiducial points with known 3D coordinates on a marker. We propose the following metrics to evaluate different modes of camera de-calibration. These metrics are used to assess the accuracy of the reconstructed points compared to the actual 3D points. They comprehensively evaluate reconstruction performance by considering both geometric and reprojection errors.

5.3.1 Normal Vector Comparison

The first metric compares the normal vectors of triangles formed by the reconstructed points with those formed by the actual points. The steps are as follows:

- (1) Identify two triangles from the set of reconstructed points and the corresponding triangles from the actual points.
- (2) Compute the normal vectors for both sets of triangles.
- (3) Compare the normal vectors to assess the alignment and orientation of the reconstructed points relative to the actual points.

This comparison can be quantified using the dot product of the normal vectors, which indicates the cosine of the angle between them. A dot product close to 1 signifies a high degree of alignment, while a dot product of zero shows no alignment in the vector space.

There are 4 different possible choices of triangles that we can use to perform an averaging or maximization process on the errors among them.

5.3.2 Angle Comparison

The second metric involves comparing the angles within the identified triangles:

- (1) Calculate the internal angles of the triangles formed by the reconstructed points.
- (2) Calculate the internal angles of the corresponding triangles formed by the actual points.

- (3) Compare these angles to evaluate how closely the reconstructed geometry matches the actual geometry. This can be done by computing the absolute differences between corresponding angles.

Same as the previous metric, here there are 4 different possible choices of the triangles, each containing three different angles to compare with the corresponding angle in the actual reconstructed triangle. We can again perform an averaging or maximization process on the errors among them.

5.3.3 Midpoint Reprojection Error

The third metric focuses on the reprojection error of midpoints between actual and reconstructed points:

- (1) Compute the midpoints between pairs of corresponding actual and reconstructed points.
- (2) Reproject these midpoints back into the image plane using the camera parameters.
- (3) Calculate the error between the reprojected points and the actual image points.

This reprojection error provides insight into the accuracy of the 3D reconstruction in relation to the 2D image coordinates. The error can be quantified using the Euclidean distance between the reprojected points and the actual 2D points in the image.

Chapter 6

Conclusion and Future work

In this thesis, we began with a comprehensive literature review of the networks and backbones utilized for the image-based camera calibration task. Subsequently, we introduced the online camera calibration problem and proposed our solution for dynamic environments. Our method demonstrates promising results and compatibility with various camera setups, models, and scenes. While our method effectively addresses the problem of a camera setup with known initial parameters, we achieved an acceptable error rate even with different percentages of perturbation, making it suitable for critical medical applications. A key aspect of our design is the development of the camera synthesis module, which provides access to unlimited training samples. This capability allows us to train our models extensively while maintaining low inference times when deployed in real-time.

The results indicate desirable error rates for the image-based task with the overhead setup. However, there remains room for improvement in more general cases. This challenge arises from the inherent difficulty in addressing sparse images with few key points and some color-coded key points using the current method and backbones tested in our work. Future research can focus on designing more complex artifacts and scenes, which will enable the model to train on richer, more complex data points. This approach could lead to successful calibration in general cases with different multi-camera setup positions and orientations.

Bibliography

- [1] idigitaldarwin, “Normal lens vs fisheye –.” <https://shorturl.at/10mXn>. Accessed: 2024-5-31.
- [2] Koushik, “Understanding convolutional neural networks (CNNs) in depth.” <https://shorturl.at/HrQYp>, Nov. 2023. Accessed: 2024-5-31.
- [3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.
- [4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” 2021.
- [5] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, “On the continuity of rotation representations in neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.
- [7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.
- [8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.

- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.
- [10] C. B. Duane, “Close-range camera calibration,” *Photogramm. Eng.*, vol. 37, no. 8, pp. 855–866, 1971.
- [11] Y. Abdel-Aziz and H. M. Karara, “Direct linear transformation into object space coordinates in close-range photogrammetry,” in *Proceedings of the Symposium Close-Range Photogrammetry*, pp. 1–18.
- [12] R. Tsai, “A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.
- [13] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.
- [14] S. Shah and J. Aggarwal, “A simple calibration procedure for fish-eye (high distortion) lens camera,” in *Proceedings of the 1994 IEEE international Conference on Robotics and Automation*, pp. 3422–3427, IEEE, 1994.
- [15] G. Nakano, “A versatile approach for solving pnp, pnpf, and pnpfr problems,” in *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III 14*, pp. 338–352, Springer, 2016.
- [16] S. Urban, J. Leitloff, and S. Hinz, “Mlpnp—a real-time maximum likelihood solution to the perspective-n-point problem,” *arXiv preprint arXiv:1607.08112*, 2016.
- [17] S. Haner and K. Astrom, “Absolute pose for cameras under flat refractive interfaces,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1428–1436, 2015.

- [18] M. W. Cao, W. Jia, Y. Zhao, S. J. Li, and X. P. Liu, "Fast and robust absolute camera pose estimation with known focal length," *Neural Computing and Applications*, vol. 29, pp. 1383–1398, 2018.
- [19] A. D. Nguyen and M. Yoo, "Calibbd: Extrinsic calibration of the lidar and camera using a bidirectional neural network," *IEEE Access*, vol. 10, pp. 121261–121271, 2022.
- [20] R. Itu, D. Borza, and R. Danescu, "Automatic extrinsic camera parameters calibration using convolutional neural networks," in *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 273–278, IEEE, 2017.
- [21] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, pp. 2938–2946, 2015.
- [22] S. Workman, C. Greenwell, M. Zhai, R. Baltenberger, and N. Jacobs, "Deepfocal: A method for direct focal length estimation," in *2015 IEEE International Conference on Image Processing (ICIP)*, pp. 1369–1373, IEEE, 2015.
- [23] A. Cramariuc, A. Petrov, R. Suri, M. Mittal, R. Siegwart, and C. Cadena, "Learning camera miscalibration detection," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4997–5003, IEEE, 2020.
- [24] A. Hagemann, M. Knorr, and C. Stiller, "Deep geometry-aware camera self-calibration from video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3438–3448, 2023.
- [25] O. Bogdan, V. Eckstein, F. Rameau, and J.-C. Bazin, "Deepcalib: A deep learning approach for automatic intrinsic calibration of wide field-of-view cameras," in *Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production*, pp. 1–10, 2018.
- [26] T. Svoboda, D. Martinec, and T. Pajdla, "A convenient multicamera self-calibration for virtual environments," *Presence: Teleoperators & virtual environments*, vol. 14, no. 4, pp. 407–422, 2005.

- [27] J. Heikkilä and O. Silvén, “A four-step camera calibration procedure with implicit image correction,” in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pp. 1106–1112, IEEE, 1997.
- [28] O. D. Faugeras, Q. T. Luong, and S. J. Maybank, “Camera self-calibration: Theory and experiments,” in *Computer Vision—ECCV’92: Second European Conference on Computer Vision Santa Margherita Ligure, Italy, May 19–22, 1992 Proceedings 2*, pp. 321–334, Springer, 1992.
- [29] C. S. Fraser, “Digital camera self-calibration,” *ISPRS Journal of Photogrammetry and Remote sensing*, vol. 52, no. 4, pp. 149–159, 1997.
- [30] R. I. Hartley, “Self-calibration from multiple views with a rotating camera,” in *Computer Vision—ECCV’94: Third European Conference on Computer Vision Stockholm, Sweden, May 2–6, 1994 Proceedings, Volume I 3*, pp. 471–478, Springer, 1994.
- [31] Y. Hold-Geoffroy, K. Sunkavalli, J. Eisenmann, M. Fisher, E. Gambaretto, S. Hadap, and J.-F. Lalonde, “A perceptual measure for deep single image camera calibration,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2354–2363, 2018.
- [32] T. H. Butt and M. Taj, “Camera calibration through camera projection loss,” in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2649–2653, IEEE, 2022.
- [33] M. Lopez, R. Mari, P. Gargallo, Y. Kuang, J. Gonzalez-Jimenez, and G. Haro, “Deep single image camera calibration with radial distortion,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11817–11825, 2019.
- [34] N. Wakai and T. Yamashita, “Deep single fisheye image camera calibration for over 180-degree projection of field of view,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1174–1183, 2021.