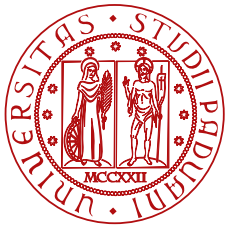

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Computer Engineering

Single-frame multi-camera to robot pose estimation

Master Candidate

Alberto Cappelletto

Student ID 2091905

Supervisor

Prof. Stefano Ghidoni

Co-supervisor

Davide Allegro, Ph.D Student

Academic Year
2023/2024

To all the ones that believed in me

Abstract

Human-robot collaboration relies on multi-camera systems to robustly monitor human operators in a robotic workcell. In this scenario, precise localization of the person in the robot coordinate system is essential, making the calibration of the camera network critical. In this work, we propose an innovative method to calibrate a network of cameras using a robot present in the cameras' field of view as a calibration model, based on a single image per camera, and to estimate the pose between the cameras and the robot. This approach is innovative because previous methods have primarily focused on calibrating individual cameras.

We demonstrate the effectiveness of our model by comparing it with the single-camera case, highlighting improvements in robustness and accuracy. We analyze how our method performs with 3, 4, and 5 cameras, and how the distance of the cameras to the robot affects our estimations. The experiments show that our method is more accurate and robust than the single-camera method, achieving an increase of about 38% in rotation estimation and 40% in translation estimation for some cameras. Our findings indicate that our model successfully handles variations in the number of cameras and is robust to changes in the setup configuration.

Sommario

La collaborazione uomo-robot si basa su sistemi multi-camera per monitorare in modo robusto gli operatori umani in una cella di lavoro robotica. In questo scenario, la localizzazione precisa della persona nel sistema di coordinate del robot è essenziale, rendendo critica la calibrazione della rete di telecamere. In questo lavoro, proponiamo un metodo innovativo per calibrare una rete di telecamere utilizzando un robot presente nel campo visivo delle telecamere come modello di calibrazione, basato su una singola immagine per telecamera, e per stimare la posa tra le telecamere e il robot. Questo approccio è innovativo perché i metodi precedenti si sono concentrati principalmente sulla calibrazione di singole telecamere.

Dimostriamo l'efficacia del nostro modello confrontandolo con il caso di una singola telecamera, evidenziando miglioramenti in robustezza e accuratezza. Analizziamo come il nostro metodo si comporta con 3, 4 e 5 telecamere, e come la distanza delle telecamere dal robot influisce sulle nostre stime. Gli esperimenti mostrano che il nostro metodo è più accurato e robusto rispetto a quello con una singola telecamera, raggiungendo un aumento di circa il 38% nella stima della rotazione e del 40% nella stima della traslazione per alcune telecamere. I nostri risultati indicano che il nostro modello gestisce con successo le variazioni nel numero di telecamere ed è robusto ai cambiamenti nella configurazione dell'installazione.

Contents

List of Figures

List of Tables

List of Acronyms

HRC Human Robot Collaboration

SVD Singular Value Decomposition

PnP Perspective-n-Point

RANSAC RANdom SAmple Consensus

DLT Direct Linear Transform

BA Bundle Adjustment

SfM Structure form Motion

SLAM Simultaneous Localization And Mapping

AR Augmented Reality

CNN Convolutional Neural Network

SIFT Scale-Invariant Feature Transform

SURF Speeded Up Robust Features

AKAZE Accelerated-KAZE

ORB Oriented FAST and Rotated BRIEF

BRIEF Binary Robust Independent Elementary Features

FAST Features from Accelerated Segment Test

DoF Degrees of Freedom

CPN Cuboid Proposal Network

PRN Pose Regression Network

SMPL Skinned Multi-Person Linear (model)

ROS Robot Operating System

IQR InterQuartile Range

1 Introduction

In the realm of computer vision, calibrating cameras is crucial for ensuring accurate and reliable measurements, especially in human-robot collaboration (HRC) tasks. In HRC scenarios, accurate camera calibration is indispensable for robots to accurately perceive and interact with humans within a shared environment, which is vital for safety and efficiency. Calibrated cameras provide precise spatial data, allowing robots to navigate and respond appropriately without collisions or errors. Furthermore, calibration of multiple cameras allows for the integration of sensor data from different viewpoints, creating a coherent and comprehensive view of the workspace. This holistic view is critical for tasks that require an understanding of the entire environment, enhancing the overall effectiveness of human-robot interactions. Therefore, having well-calibrated cameras and collecting comprehensive scene information are fundamental to the success of HRC applications.

Camera calibration is the process of determining the intrinsic and extrinsic parameters of a camera system to ensure accurate interpretation of the geometric properties of images. Intrinsic parameters describe the internal characteristics of the camera, such as focal length, principal point, and lens distortion coefficients. These parameters affect how light rays are refracted and focused onto the camera sensor, thereby influencing the geometry of captured images. Extrinsic parameters, on the other hand, define the position and orientation of the camera relative to the scene being observed. These parameters specify the spatial transformation between the camera's coordinate system and the world coordinate system, enabling accurate mapping of image coordinates to real-world coordinates.

The process of camera calibration typically involves capturing images of calibration patterns, such as checkerboards or grids, from different viewpoints and orientations [78, 69, 58]. These images are then processed to extract feature points, which serve as reference points for estimating both intrinsic and extrinsic parameters. Various

mathematical techniques, including geometric algorithms and optimization methods, are employed to compute these parameters based on the observed feature points.

In addition to calibrating cameras individually, it is also crucial to calibrate the cameras with respect to the robot. This ensures perfect synchronization between the robot's movements and the camera's observations, which is essential for tasks requiring close interaction between the human operator and the robot. Classical calibration methods for this setup typically involve placing a calibration pattern on the robot's end effector and capturing images from different camera angles [15, 62].

Classical camera calibration methods, like the one proposed by Zhang [78], while effective, are associated with several challenges:

- **Manual Intervention**: They often require manual intervention, such as capturing images of calibration patterns from multiple viewpoints and orientations. This process can be time-consuming, not extremely accurate, because if the pattern is moved manually there can be cases where the images are not clear due to the human holding the pattern not being completely still, and labor-intensive, particularly for systems with multiple cameras or in dynamic environments where frequent recalibration is necessary.

- **Sensitivity to Environmental Conditions**: These calibration techniques may be sensitive to changes in environmental conditions, such as variations in lighting, camera positions, or scene complexity. These changes can impact the accuracy of calibration results and require adjustments to be made manually, limiting the robustness of the calibration process.

- **Limited Adaptability**: In dynamic environments, where cameras may undergo changes in focal length, lens distortion, or position, these calibration techniques may fail to maintain accuracy without continuous recalibration. These methods are often not well-suited for real-time applications where rapid adaptation to changing conditions is required. The iterative nature of optimization-based approaches and the manual intervention involved in feature-based methods make them impractical for scenarios that demand continuous recalibration in real-time.

- **Complexity and Scalability**: The mathematical models and optimization algorithms used in classical calibration methods can be complex and computationally demanding, particularly for large-scale camera networks or systems with non-linear distortion. Scaling traditional calibration techniques to handle complex scenarios can be challenging and may require significant computational resources.

- **Dependency on Calibration Patterns**: Many of these calibration methods rely on specific calibration patterns, such as checkerboards or grids [78, 19], for fea-

ture extraction and parameter estimation. While these patterns are effective in controlled environments, they may not be suitable for complex or unstructured scenes, limiting the applicability of traditional calibration techniques in diverse scenarios.

1.1 Contributions

In recent years, advancements in machine learning and deep learning have catalyzed an innovative approach: using objects of known geometry crucial to subsequent tasks for calibration. This new method diverges from traditional practices reliant on human intervention and specific calibration markers. By leveraging deep learning capabilities, this approach not only enhances robustness against environmental variations but also enables real-time calibration solutions.

One notable advancement in these techniques involves directly employing a robot for calibration tasks [31, 76]. This method is distinguished by its ability to incorporate the robot's precise information into the calibration process, ensuring optimal alignment between camera observations and robot actions.

This thesis introduces a novel system designed to estimate the extrinsic parameters of a network of N cameras (where $N > 2$) using a robot as a known geometry object, and to determine the robot's pose with respect to the camera network. To our knowledge, this is the first method to address the challenge of estimating the pose of an uncalibrated camera network relative to a manipulating robot. Previous research primarily focuses on single-camera setups [31, 74, 76, 75, 22], concentrating on estimating the relative pose between the camera and robot.

In single-camera setups, the majority of approaches utilize neural networks to recover relative rotation and translation, either directly or as an initial step before solving the Perspective-n-Point (PnP) problem [32, 29, 72]. However, using a single camera can lead to less precise results due to occlusions or configuration discrepancies not represented in the training dataset. One of the main weaknesses of such approaches is their heavy reliance on the initial estimation by the deep learning model. Poor results may arise due to factors such as specific robot configurations, changes in lighting, or variations in the robot-camera distance compared to the training data, resulting in significantly erroneous pose estimates. Another critical challenge is occlusion, which can occur when objects or people obstruct the view or due to self-occlusion caused by robot configuration, all of which can degrade accuracy.

While calibrating multiple cameras introduces complexity compared to a single-camera setup, it also enhances estimation accuracy by providing richer scene information.

This work aims to advance existing research by improving accuracy and robustness,

particularly in multi-camera scenarios. Our proposed approach begins with estimating inter-camera poses using feature matching [35], followed by recovering the rotation and translation components [61]. We then utilize epipolar geometry for multi-view information fusion [79] to address occlusion challenges. Subsequently, we refine these results through constrained triangulation [8] and a final bundle adjustment step. A detailed description of the method is provided in Chapter 4.

The efficacy of our method is evaluated through experiments where we compare it with a single-camera approach repeated across all the cameras. We assess our model's performance with varying numbers of cameras (3, 4, and 5) and explore the impact of different camera-to-robot distances. Further discussion and numerical results will be presented in Chapter 5.

1.2 Thesis Outline

The thesis follows a structured organization: Chapter 2 explores essential preliminary notions to facilitate understanding of the work undertaken. Chapter 3 critically reviews the literature on the subject, emphasizing key methodologies employed, their strengths and weaknesses. Chapter 4 introduces the overarching framework of the proposed method, providing detailed insights into its constituent steps. Chapter 5 outlines the experimental procedures conducted and presents the corresponding results. Finally, Chapter 6 offers a comprehensive conclusion, summarizing contributions made and elucidating potential avenues for future research.

2

Preliminary Notions

In this chapter, the groundwork for a comprehensive understanding of the forthcoming sections is presented by delving into pivotal preliminary notions. These fundamental concepts serve as the bedrock upon which subsequent discussions and explorations will be built.

2.1 Pose

The pose of an object, as shown in Figure 2.1 refers to its specific position and orientation in space relative to a defined coordinate system or reference frame. It encompasses two fundamental aspects: translation, which describes: the object's location or displacement in three-dimensional space along the x, y, and z axes and rotation, indicating the orientation of the object concerning these axes. It defines how the object is turned or tilted relative to the coordinate system.

The translation component of a pose, which describes the displacement or location of an object in three-dimensional space, can be represented in various ways:

- **Cartesian Coordinates**: The most common representation involves using Cartesian coordinates (x, y, z) to denote the position of an object in a three-dimensional space. Each coordinate represents the displacement along the x, y, and z axes, respectively, from a reference point or origin.
- **Homogeneous Coordinates**: Often used in computer graphics and transformations, homogeneous coordinates add a fourth coordinate (usually 1) to the Cartesian coordinates, enabling efficient representation of translation along with rotations and perspective transformations within matrices.

The rotation component of a pose can be represented in various ways, each with its advantages and specific use cases, such as:

- **Euler Angles**: Euler angles describe rotations using three angles, typically representing rotations about the x, y, and z axes. While intuitive, they can suffer from issues like gimbal lock and ambiguity in certain orientations.
- **Rotation Matrix**: A rotation matrix is a 3x3 matrix that directly encodes the orientation of an object in three-dimensional space. Each column represents the direction of the axes of the rotated frame in the reference frame.
- **Quaternion**: Quaternions are four-dimensional mathematical entities used to represent rotations. They are compact, computationally efficient, and free from some of the issues that Euler angles face, like gimbal lock. Quaternions consist of a scalar part and a vector part

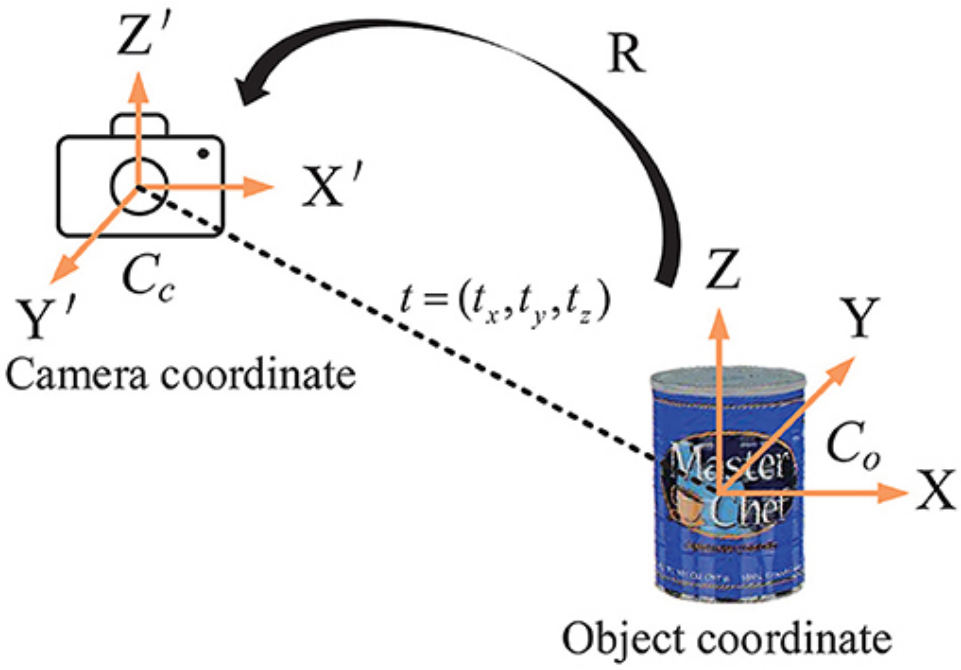

Figure 2.1: Pose of a rigid object with respect to the camera coordinate frame. R represent the rotation and t the translation.

2.2 Projective Geometry

Projective geometry is a branch of mathematics that studies geometric properties and relationships that are invariant under projective transformations.
By extending the principles of Euclidean geometry, projective geometry allows for the representation and manipulation of images in a way that accurately reflects the projection of a three-dimensional world onto a two-dimensional image plane, as shown in Figure 2.2.
In Euclidean geometry, the parallel axiom states that parallel lines never intersect.

However, projective geometry changes this by ensuring that all lines intersect, even parallel lines. This is achieved by introducing so called points at infinity, where each pair of parallel lines meets, corresponding to their direction. The collection of all these points at infinity forms what is known as the line at infinity.
A point (x,y) in the Euclidean plane can be represented as (x,y,1) in homogeneous coordinates. A point at infinity can be represented as (x,y,0). Similarly, a point (X,Y,Z) in the 3D world is represented as (X,Y,Z,1). Homogeneous coordinates allow for the inclusion of points at infinity, which are essential in projective geometry.

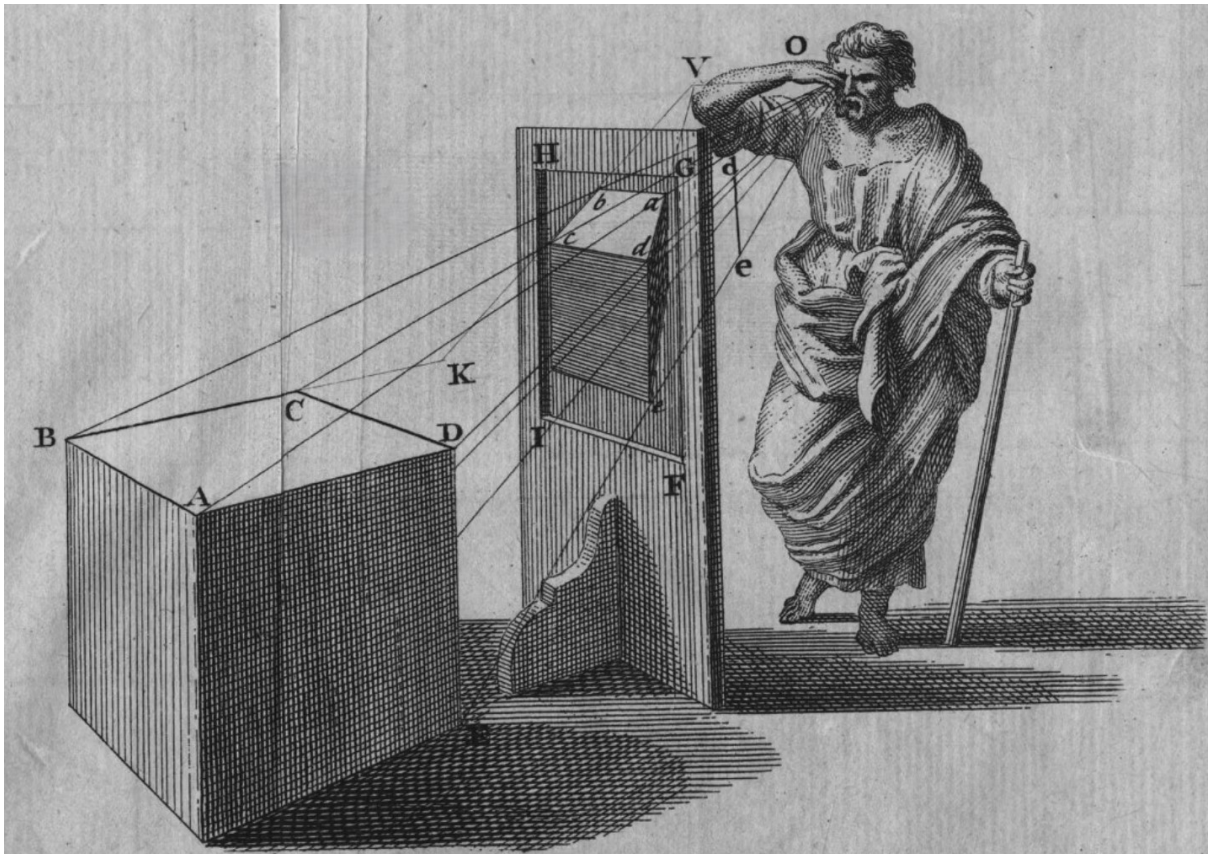

Figure 2.2: Illustration of projective geometry principles showing how a three-dimensional object is projected onto a plane, with lines converging at the viewer's eye.

2.2.1 Pinhole Camera Model

The pinhole camera model is used to describe how a three-dimensional scene is projected onto a two-dimensional image plane. This model simplifies the behavior of a camera to its essential geometric principles, excluding effects like lens distortion and focusing. The model is named after the pinhole camera, a simple camera without a lens but with a tiny aperture, or pinhole, which allows light to enter and form an image on the opposite side.

In the pinhole camera model, light from a scene passes through a single point (the pinhole) and projects an inverted image onto an image plane. This can be understood through the following components:

- **Camera Coordinate System**: Defined with the camera's optical center at the origin $(0, 0, 0)$ of the 3D space.
- **Image Plane:** A 2D plane where the projected image is formed, located at a distance $f = (f_x, f_y)$ (focal length) from the pinhole.
- **Pinhole (Optical Center)**: The point through which all light rays pass, situated at the origin of the camera coordinate system.
- **Projection**: A mapping from 3D points in the scene to 2D points on the image plane.

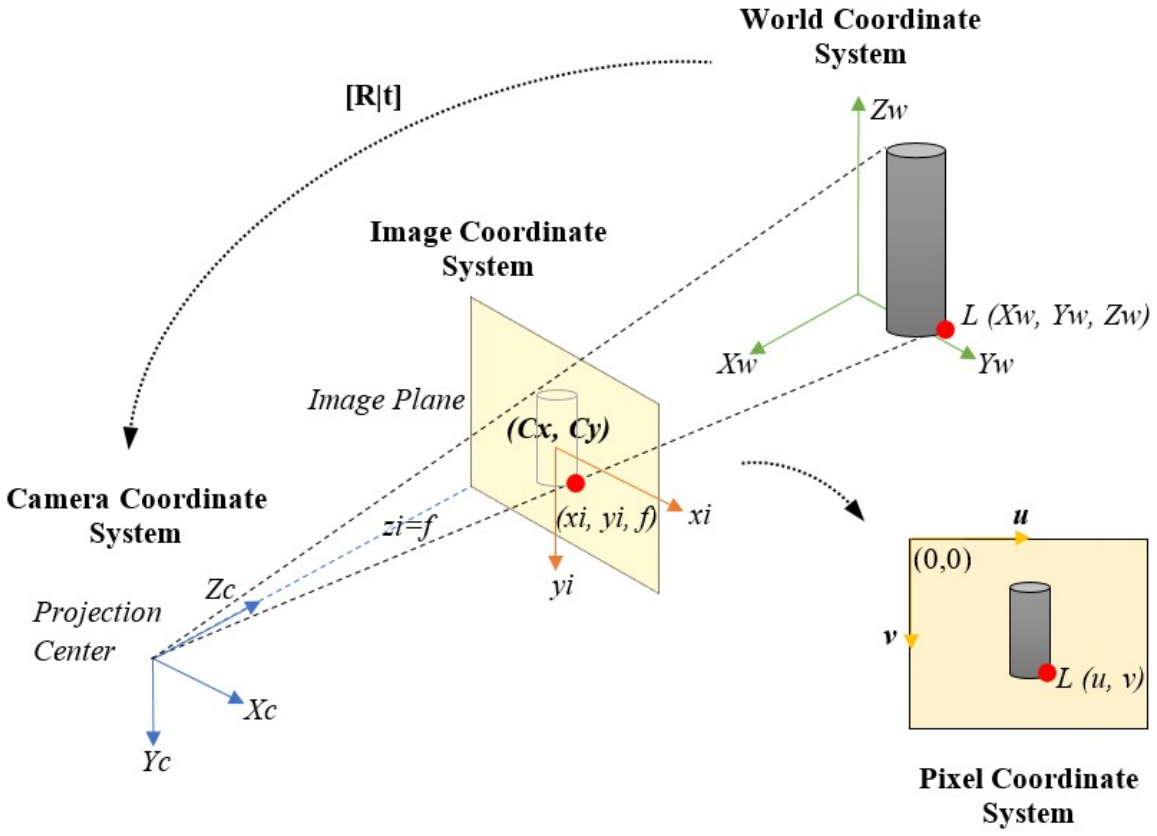

Figure 2.3: Illustration of the pinhole camera model demonstrating the relationship between the world coordinate system, camera coordinate system, image coordinate system, and pixel coordinate system.

To mathematically describe the projection, consider a point $\mathbf{P} = (X, Y, Z)$ in the 3D world. The corresponding point $\mathbf{p} = (x, y)$ on the 2D image plane can be found using

similar triangles. The projection equations are derived as follows:

$$x = f\frac{X}{Z} \tag{2.1}$$

$$y = f\frac{Y}{Z} \tag{2.2}$$

Here, (X, Y, Z) are the coordinates of the 3D point in the camera coordinate system, f is the focal length, and (x, y) are the coordinates of the projected point on the image plane.
Using homogeneous coordinates simplifies the representation of projection equations, especially for more complex transformations. A 3D point $\mathbf{P} = (X, Y, Z)$ in homogeneous coordinates is represented as $\mathbf{P} = (X, Y, Z, 1)$. The projection of this point onto the image plane can be written using a projection matrix $\mathbf{P}$.
The projection matrix $\mathbf{P}$ for the pinhole camera model is:

$$\mathbf{P} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The transformation from the 3D world point $\mathbf{P'}$ to the 2D image point $\mathbf{p}$ in homogeneous coordinates is given by:

$$\mathbf{p} = \mathbf{PP'}$$

In practical applications, the pinhole camera model involves intrinsic and extrinsic parameters.
Intrinsic parameters describe the internal characteristics of the camera, such as focal length and principal point (the point where the optical axis intersects the image plane). These parameters are typically represented by the intrinsic matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

where f_x and f_y are the focal lengths in the x and y directions, and (c_x, c_y) is the principal point.
Extrinsic parameters describe the position and orientation of the camera in the world coordinate system. These parameters are represented by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$:

$$\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

Combining intrinsic and extrinsic parameters, the full projection matrix $\mathbf{P}$ can be written as:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

where $[\mathbf{R}|\mathbf{t}]$ is a 3×4 matrix combining rotation and translation.
The projection of a 3D point $\mathbf{P}$ in homogeneous coordinates $(X, Y, Z, 1)$ to a 2D image point $\mathbf{p}$ in homogeneous coordinates $(x, y, 1)$ is:

$$\mathbf{p} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{P}$$

The pinhole camera model is an extremely useful tool to understand projective geometry thanks to its simplicity, indeed it's strength lies in the ability to reduce the problem to it's basic components.
However, the model has limitations. It assumes an ideal pinhole camera without lens distortion, which is not realistic for actual cameras with lenses. Real-world cameras often require additional modeling to account for these distortions.

2.3 Epipolar Geometry

Epipolar geometry deals with the geometric relationship between two views of the same scene. It provides the mathematical framework to understand the constraints between the image points captured by two cameras.
When two cameras capture images of the same scene from different viewpoints, the line connecting the two camera centers is called the baseline. The points where this line intersects the image planes of the two cameras are called the epipoles. Each epipole is the image of the other camera center.
An epipolar plane is any plane that contains the baseline. Each 3D point in space defines a unique epipolar plane. The intersection of the epipolar plane with the image planes defines the epipolar lines (these notions are exemplified in the figure 2.4).
Epipolar lines are the projections of the epipolar plane intersections on the image planes. For a given point in one image, the corresponding point in the other image must lie on the corresponding epipolar line. This constraint reduces the search space for matching points to one dimension.

2.3.1 Fundamental Matrix

The epipolar constraint provides a relationship between corresponding points in stereo images. Given a point $\mathbf{p} = (x, y, 1)^T$ in the first image and its corresponding

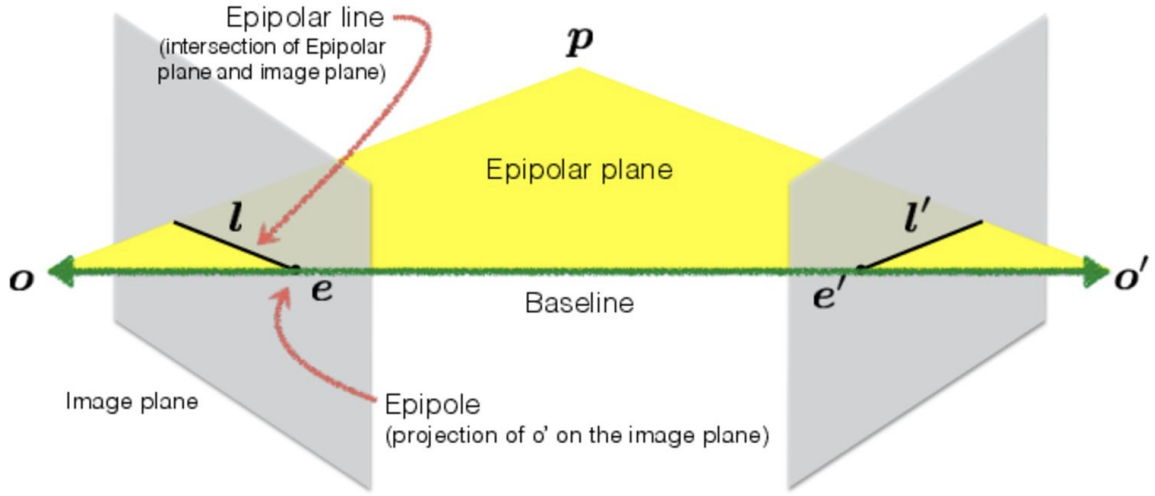

Figure 2.4: Epipolar geometry basic concepts.

point $\mathbf{p}' = (x', y', 1)^T$ in the second image, the epipolar constraint can be expressed as:

$$\mathbf{p}'^T \mathbf{F} \mathbf{p} = 0$$

Here, **F** is the fundamental matrix, which encapsulates the intrinsic and extrinsic parameters of the camera pair.
The fundamental matrix **F** is a 3×3 matrix that relates the corresponding points between two images. It is defined up to a scale factor and has rank 2.

2.3.2 Essential Matrix

The essential matrix **E** is a 3×3 matrix that relates corresponding points in stereo images when the intrinsic parameters are known. The essential matrix encodes the relative rotation and translation between the two camera coordinate systems. For corresponding points **p** and **p'** in normalized image coordinates, the epipolar constraint is:

$$\mathbf{p}'^T \mathbf{E} \mathbf{p} = 0$$

The essential matrix **E** can be related to the fundamental matrix **F** through the intrinsic camera matrices **K** and **K'** of the two cameras:

$$\mathbf{E} = \mathbf{K}'^T \mathbf{F} \mathbf{K}$$

Here, **K** and **K'** are the intrinsic parameter matrices of the two cameras.
Given the relative rotation **R** and translation **t** between the cameras, the essential matrix

is:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$$

where $[\mathbf{t}]_\times$ is the skew-symmetric matrix of the translation vector $\mathbf{t}$:

$$[\mathbf{t}]_\times = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}$$

The essential matrix $\mathbf{E}$ can be decomposed to retrieve the relative rotation $\mathbf{R}$ and translation $\mathbf{t}$ between the cameras. One of the most used methods to achieve it is through Singular Value Decomposition (SVD).
SVD decomposes a matrix into three component matrices: an orthogonal matrix $\mathbf{U}$, a diagonal matrix $\mathbf{\Sigma}$ containing singular values, and another orthogonal matrix $\mathbf{V}^T$, such that $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. SVD is instrumental in decomposing the essential matrix, allowing for the extraction of camera motion parameters (rotation and translation) between two views. The method's robustness and ability to handle noise make it particularly useful in applications requiring precise geometric transformations and feature matching across images.

2.4 Estimation Background

2.4.1 RANSAC

RANSAC, which stands for Random Sample Consensus, is an iterative method used to estimate parameters of a mathematical model from a dataset that contains outliers. It was introduced by Fischler and Bolles [16] in 1981 and has since become a staple in the fields of computer vision and robotics for tasks such as model fitting, motion estimation, and object recognition.
The primary strength of RANSAC lies in its robustness to outliers. Unlike traditional least-squares fitting methods, RANSAC can handle datasets where a significant fraction of the data points are outliers. The algorithm works by repeatedly selecting random subsets of the data, fitting a model to these subsets, and then determining how many data points from the entire set fit the estimated model within a predefined tolerance.
The RANSAC algorithm can be summarized in the following steps:

1. **Random Sampling**: Randomly select a subset of the original data points.
2. **Model Fitting**: Fit a model to the selected subset.
3. **Consensus Set**: Determine the set of all data points that are within a certain distance from the model.

4. **Model Evaluation**: Evaluate the model based on the number of inliers (points in the consensus set).

5. **Iterate**: Repeat the above steps for a fixed number of iterations or until a model with a sufficiently large consensus set is found.

The final output is the model that has the largest consensus set.
Altough really powerful, it has also some disadvantages:

- **Computational Cost**: The iterative nature of RANSAC can be computationally expensive, especially for large datasets or complex models.

- **Parameter Sensitivity**: The performance of RANSAC depends on parameters such as the number of iterations, tolerance threshold, and sample size, which need to be carefully tuned.

- **No Guarantee of Optimality**: RANSAC may not always find the optimal model, especially if the number of iterations is insufficient.

Implementing RANSAC involves several key considerations: The type of model to be fitted depends on the specific application (the model defines the mathematical relationship between the data points). The error tolerance threshold determines how close a data point must be to the model to be considered an inlier. This threshold should be set based on the expected noise level in the data. The number of iterations should be chosen to balance computational cost and the likelihood of finding a good model. More iterations increase the chances of finding the correct model but also increase computation time.

2.4.2 Perspective-n-Points Problem

The Perspective-n-Point (PnP) problem [29, 32, 43, 72] is a fundamental problem in computer vision and robotics. It involves determining the pose of a camera given a set of n 3D points in the world coordinate system and their corresponding 2D projections in the image plane.
The PnP problem can be mathematically formulated as follows: Given a set of n 3D points $\mathbf{X}_i = [X_i, Y_i, Z_i]^T$ in the world coordinate system and their corresponding 2D projections $\mathbf{x}_i = [u_i, v_i]^T$ in the image plane, determine the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ that relate the 3D points to their 2D projections through the following equation:

$$s\mathbf{x}_i = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X}_i$$

where $\mathbf{K}$ is the camera intrinsic matrix, and s is a scale factor.
Various algorithms have been developed to solve the PnP problem. These methods can be broadly categorized into linear and non-linear approaches. Linear methods aim to solve the PnP problem by formulating it as a linear system of equations. These methods are generally faster but may be less accurate due to their sensitivity to noise. Direct Linear Transformation (DLT) is a popular linear method that estimates the camera pose by solving a set of linear equations derived from the projection equation. However, DLT requires a minimum of six points and is sensitive to noise and outliers.
Non-linear methods refine the initial pose estimate obtained from linear methods through iterative optimization techniques. These methods are generally more accurate but computationally expensive. Levenberg-Marquardt is a widely used non-linear optimization technique that iteratively minimizes the reprojection error, which is the difference between the observed and projected 2D points.
While the PnP problem is well-studied, several challenges remain: Real-world data is often noisy and contains outliers, which can affect the accuracy of the estimated pose. Robust algorithms like RANSAC are often combined with PnP to handle outliers. For real-time applications, the computational efficiency of PnP algorithms is crucial. Efficient methods like EPnP [32] and optimizations in non-linear solvers are essential to meet these demands.

2.4.3 Bundle Adjustment

Bundle Adjustment (BA) is a crucial optimization technique in computer vision and photogrammetry used to refine the 3D coordinates of a scene's structure and the camera parameters simultaneously. It plays a vital role in applications such as Structure from Motion (SfM), Simultaneous Localization and Mapping (SLAM), and 3D reconstruction.
Bundle adjustment optimizes the 3D structure of a scene and the camera poses by minimizing the reprojection error. Given a set of n 3D points $\mathbf{X}_i$ and m camera poses $(\mathbf{R}_j, \mathbf{t}_j)$, along with the corresponding 2D image points $\mathbf{x}_{ij}$, bundle adjustment aims to minimize the following objective function:

$$\min_{\mathbf{R}_j, \mathbf{t}_j, \mathbf{X}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} \left\| \mathbf{x}_{ij} - \pi(\mathbf{K}[\mathbf{R}_j | \mathbf{t}_j] \mathbf{X}_i) \right\|^2$$

Here, π denotes the projection function, $\mathbf{K}$ is the camera intrinsic matrix, and $\|\cdot\|$ represents the Euclidean distance.
The optimization in bundle adjustment is typically performed using iterative techniques such as:

- **Levenberg-Marquardt Algorithm**: A popular choice for non-linear least squares

problems that combines the advantages of gradient descent and Gauss-Newton methods.

- **Gauss-Newton Algorithm**: An iterative method that approximates the non-linear optimization problem using a second-order Taylor series expansion.
- **Dogleg Method**: A trust-region method that combines the steepest descent direction and the Gauss-Newton direction.

Recent advancements in bundle adjustment focus on improving its efficiency, scalability, and robustness. A few examples are:

- **Sparse bundle adjustment**: Exploiting the sparsity of the Jacobian matrix in the optimization problem can significantly reduce the computational complexity. Sparse bundle adjustment techniques leverage this sparsity to improve the efficiency of the optimization process.
- **Robust bundle adjustment**: To handle outliers in the data, robust estimation techniques such as RANSAC and M-estimators are integrated with bundle adjustment. These techniques improve the robustness and accuracy of the optimization.
- **Incremental bundle adjustment**: Incremental BA methods update the 3D structure and camera parameters incrementally as new images are added. This approach is particularly useful in real-time applications like SLAM, where new data is continuously acquired.

Despite its effectiveness, bundle adjustment faces several challenges. Bundle adjustment is computationally intensive, particularly for large-scale problems. Developing more efficient algorithms and leveraging hardware acceleration are ongoing research areas. Real-time bundle adjustment is essential for applications like SLAM and AR. Balancing accuracy and computational efficiency in real-time scenarios remains a significant challenge. Traditional bundle adjustment assumes a static scene. Extending bundle adjustment techniques to handle dynamic scenes, where objects and cameras move independently, is an area of active research. Improving the robustness of bundle adjustment to outliers and noise in the data is crucial for reliable 3D reconstruction and pose estimation.

3

Related Works

The task of calibrating a camera network using a robot as a calibration pattern and estimating its relative position and orientation with respect to the cameras consists of two major components: camera-to-camera pose estimation and camera-to-robot pose estimation. In this chapter, we review the state of the art of methods focused on estimating the relative pose of the camera and then the most significant works that aim to determine the pose of the robot with respect to the camera network.

3.1 Relative Camera Pose Estimation

Relative camera pose estimation is fundamental in computer vision and robotics, involving determining the position and orientation of one camera relative to another. The main paradigm used is to detect and match features between the images of the cameras and then use them to recover the relative rotation and translation, either using the 8-point algorithm [37], 5-point algorithm [45], or other methods.

3.1.1 Feature-Based Approaches

Methods to detect and match features have been continuously researched because of their importance in relative camera pose estimation. These methods are mainly divided into two categories: classic and deep learning-based approaches. Classic methods such as SIFT [38] (Scale-Invariant Feature Transform) and SURF [4] (Speeded-Up Robust Features) detect and describe local features that are invariant to scale, rotation, and illumination changes, making them effective across various applications. ORB [53] (Oriented FAST and Rotated BRIEF) provides a more efficient alternative by combining the FAST [52] keypoint detector with the BRIEF [7] descriptor, maintaining robustness while significantly reducing computational cost.

However, these methods often require human intervention for parameter fine-tuning and post-processing, prompting recent advancements in feature detectors leveraging deep learning. For example, SuperPoint [12] utilizes a self-supervised neural network to detect and describe key points, demonstrating superior performance in challenging scenarios. SuperGlue [54] goes further by employing graph neural networks to learn context-aware feature matching, improving accuracy and robustness against noise and outliers. LightGlue [35] continues this trend with a focus on speed and efficiency without compromising accuracy, making it suitable for real-time applications.
These approaches handle the first step of feature detection and matching. For the second step of actual pose estimation, various methods have been proposed to use these points to obtain viable results. For example, in [45], an efficient solution is introduced for the classic five-point algorithm, to solve the relative pose problem. This problem involves determining the possible relative camera poses between two calibrated views using five corresponding points. The algorithm operates by calculating the coefficients of a tenth-degree polynomial in closed form and then finding its roots. Additionally, Kukelova et al. [30] present new, fast, and straightforward solutions to the five-point relative pose problem and the six-point focal length problem. They demonstrate that these problems can be easily formulated as polynomial eigenvalue problems of degree three and two, respectively, and can be solved using standard efficient numerical algorithms, which are an improvement over [45]. Hongdong et al. [33] introduce a simplified algorithm that utilizes the hidden variable resultant technique. Unlike the method in [45], which eliminates unknown variables sequentially using Gauss-Elimination, their algorithm simultaneously eliminates multiple unknowns. Additionally, during the equation-solving stage, rather than back-substituting to solve all unknowns one by one, they estimate all the unknown parameters at once by computing the minimal singular vector of the coefficient matrix.
One case that requires estimating the pose between cameras, and that has received a lot of attention, is Structure from Motion (SfM) [10, 13, 44, 11, 70]. Several pipelines have been developed to solve this task, such as those discussed in [57] by Schönberger and Frahm. They introduced more efficient methods for feature extraction, matching, and geometric verification. Their approach also includes a robust bundle adjustment technique that refines camera poses and 3D structure by minimizing re-projection errors. Another significant paper is [48], where authors solve the problem through direct alignment of low-level image information from multiple views. Initially refining keypoint locations before geometric estimation, they follow with post-processing refinement of points and camera poses. This method proves robust against significant detection noise and appearance changes, optimizing a feature metric error using dense features predicted by neural networks. Jianyuan et al. [24] first predict dense correspondences between frames using an optical flow estimation network. Then, a

normalized pose estimation module derives relative camera poses from these correspondences, followed by a scale-invariant depth estimation network using epipolar geometry to refine dense correspondences and estimate relative depth maps. Additionally, Schmidt et al. propose in [56] a differentiable SfM framework leveraging video sequences to enhance depth estimation and camera pose recovery. Their approach uses deep neural networks to predict depth and camera motion directly from video frames, facilitating end-to-end training and optimization. DeepV2D demonstrates superior performance in challenging environments with varying lighting and texture conditions, highlighting the potential of deep learning to augment traditional SfM methods.

3.1.2 Deep Learning-Based Methods

In recent years, a new approach has gained traction: directly estimating the relative pose between a pair of cameras from just a pair of images in an end-to-end manner. Yuheng Li et al. [34] introduce a method that directly regresses the 6-DoF (Degrees of Freedom) relative pose between two cameras. They employ supervised learning with 6D pose supervision to enhance pose estimation accuracy under challenging conditions. GRelPose [28] proposes a deep learning-based approach for relative pose regression using the pre-trained LoFTR network [23] for 2d feature extraction. A convolutional network then predicts the relative rotation and translation, demonstrating high generalization across different scenes and datasets. RelMobNet, introduced by Rajendran et al. in [50], employs an end-to-end Siamese network architecture for relative camera pose estimation. Trained on the Cambridge Landmarks dataset, it utilizes a two-stage training process to significantly improve the accuracy of translation vector estimation compared to other CNN-based methods.
All the previous methods recover the pose only up to a scale factor. MicKey [2] addresses the challenge of uniquely recovering depth information, distinguishing itself from methods that typically recover relative pose up to scale, such as those based on the essential matrix. This method introduces a novel approach to depth estimation from camera poses based on a new type of metric image-to-image correspondences.
Additionally, Rockwell et al. [51] propose a pipeline integrating feature matching, fundamental matrix resolution, and direct pose prediction using neural networks. Their model incorporates a Transformer [66] that balances between traditional and learned pose estimations, offering a prior to guide the solver effectively.
In our work, we employ LightGlue for feature point detection and matching due to its versatility across various configurations and scenes without requiring fine-tuning. It exhibits greater robustness compared to classical detectors like ORB and operates with fast inference times. Subsequently, we utilize C2P to recover both the rotation and

translation components. Our approach differs by leveraging known robot dimensions to accurately recover the scale factor. This distinguishes our method from others that only recover up to scale, such as [34, 28, 50].

3.2 Pose Estimation From a Single-Image

Single-image pose estimation involves determining the position and orientation of a camera with respect to a given coordinates frame from a single image, a task that presents significant challenges due to the limited amount of visual information available. Traditional approaches require multiple images to work and rely on predefined patterns, such as ArUco[55] and AprilTag[60] markers, placed within the environment to assist in extracting positional data and are heavily reliant on human intervention. However, recent advancements have led to more sophisticated methods that do not require external aids and work with only a single image per camera.

3.2.1 Single-Camera Robot Pose Estimation

Estimating the pose of a camera with respect to a robot[39, 82, 14] can be divided into three main approaches: keypoint-based, render-and-compare, and depth-based methods.
A notable keypoint-based model is DREAM [31], a deep neural network that, given an image, outputs a set of 2d points corresponding to the robot's joints. It then uses the Perspective-n-Point (PnP) algorithm [32] exploiting the 3D locations of the joints provided by the forward kinematics of the manipulator to accurately predict robot poses, demonstrating robustness to variations in appearance. Similarly, [22] refines keypoint detection through shape segmentation, significantly improving single-shot pose estimation precision. Lu et al.[26] discuss iterative keypoint optimization using simulation data for real-world applications, while Tian et al.[75] enhance pose estimation by incorporating temporal attention mechanisms guided by the robot's structure, ensuring more accurate and consistent pose predictions over image sequences.
Robopose[76] is an example of a render-and-compare based model. It requires the CAD model of the robot to function and works by iteratively estimating pose and joint angles by comparing rendered images of the robot model to actual images, providing a practical solution for real-world applications.
Bohg et al.[21] adopt a pixel-wise classification approach to identify different parts of the robot arm, leading to precise pose estimation. Simoni et al.[1] present a depth-based method using 'semi-perspective decoupled heatmaps' for estimating 3D poses from depth maps, offering a robust solution for pose estimation tasks.
RoboKeyGen[74] estimates both pose and angles of the robot with a keypoint-based

approach. This framework divides the high-dimensional prediction task into two subtasks: detecting 2D keypoints and converting these into 3D keypoints. Traditional deterministic regression methods can struggle with uncertainties from 2D detection errors or self-occlusions. By leveraging the robust modeling capabilities of diffusion models, this framework reformulates the challenge as a conditional 3D keypoints generation task. Each proposed solution has its own set of advantages and disadvantages. One significant disadvantage is the heavy reliance on deep learning models. This reliance means that in scenarios where the model encounters configurations not seen during training, the results may deteriorate. Another drawback is the use of a single camera, which can lead to issues, especially in situations involving occlusions. In our approach, we mitigate these issues by employing multiple cameras, thereby reducing the impact of occlusions. Additionally, we integrate information from multiple views without relying on a neural network architecture. This approach ensures that even if one camera's deep learning model performs poorly due to encountering an unfamiliar scenario, the overall results benefit from the information fused across other views.

3.2.2 Multi-Camera Robot Pose Estimation

As previously mentioned, to the best of our knowledge, there currently exists no established model for estimating robot poses using camera networks. This gap may be attributed to the relatively new nature of this research area. In our view, the primary reason for this gap lies in the scarcity of multi-view datasets available for training deep learning models. Presently, datasets are predominantly limited to single-camera setups, like, for example, the one proposed in [31].
Consequently, we directed our focus towards human pose estimation in a multi-camera environment. This field exhibits several similarities to our research. Both areas involve utilizing an object (the human body in their case) either as a calibration pattern or to directly estimate its pose relative to cameras, often employing a keypoint-based approach for pose estimation. The primary distinction lies in the fact that while human pose estimation focuses on body joints, such as knees or head, our research pertains to robot joints and their corresponding keypoints.

3.2.3 Multi-Camera Human Pose Estimation

This topic has been extensively researched [67, 71, 5, 65, 77, 73, 59] in comparison to methods involving robots. Several significant models have been proposed, with the ones resembling more our case being: [36] introduces an innovative automatic calibration technique for multi-camera systems using human joints. The approach utilizes human joints as reference points across different camera views to achieve calibration. Unlike traditional methods, this technique only needs a person to move within the

calibration area. This significantly reduces the overall cost and complexity of the calibration process.

VoxelPose, proposed in [18], is a method designed to estimate 3D poses of multiple individuals from multiple camera views. Unlike previous approaches that rely on noisy and incomplete 2D pose estimates to establish cross-view correspondence, VoxelPose operates directly in the 3D space, thereby avoiding errors in individual camera views. This is achieved by aggregating features from all camera views into the 3D voxel space, which are then processed by the Cuboid Proposal Network (CPN) to locate all individuals. Following this, the Pose Regression Network (PRN) estimates a detailed 3D pose for each detected individual.

SmartMocap[46] presents a framework for simultaneously estimating human motion and camera movements using uncalibrated RGB cameras. Their method incorporates several key components. First, they use the ground plane as a common reference to represent both body and camera motions. Second, they learn a probability distribution of short human motion sequences (approximately one second) relative to the ground plane, using this to differentiate between camera and human motion. Third, this distribution is used as a motion prior in a multi-stage optimization approach to fit the SMPL[41] human body model and the camera poses to the human body keypoints in the images.

Jiang et. al.[6] introduces a probabilistic triangulation method for estimating 3D human poses from multiple uncalibrated camera views. The core concept is to utilize a probability distribution to model the camera pose and iteratively refine this distribution based on 2D features, rather than directly using a fixed camera pose. Specifically, the method involves maintaining a camera pose distribution and updating it iteratively by calculating the posterior probability of the camera pose through Monte Carlo sampling. This approach allows for gradients to be directly back-propagated from the 3D pose estimation to the 2D heatmap, facilitating end-to-end training.

Qiu et. al[17] introduce a method to recover absolute 3D human poses from multi-view images by integrating multi-view geometric priors into their model. The approach involves two distinct steps: first they estimate the 2D poses in multi-view images, and second they reconstruct the 3D poses from these multi-view 2D poses. The authors employ a cross-view fusion mechanism within a CNN to jointly estimate 2D poses across multiple views, ensuring that the 2D pose estimation for each view is enhanced by the information from other views. Subsequently, they utilize a recursive Pictorial Structure Model to reconstruct the 3D pose from the multi-view 2D poses.

Moliner et. al. [47] build upon prior works and introduce several novel concepts to enhance the accuracy of human-pose-based extrinsic calibration. They formulate a robust reprojection loss, developed from a deeper understanding of the sources of pose estimation error and construct a 3D human pose likelihood model, which is learned

from motion capture data.
[3] proposes an online, marker-free method for extrinsic camera calibration using person keypoint detections. The approach eliminates the need for physical markers and allows for real-time calibration adjustments, making it highly practical for dynamic environments. Gyeongsik et. al.[42] introduce a fully learning-based, camera distance-aware top-down method for estimating 3D multi-person poses from a single RGB image. The proposed system comprises three main modules: human detection, absolute 3D human root localization, and root-relative 3D single-person pose estimation. This pipeline ensures a comprehensive approach to accurately determining the 3D positions of multiple individuals within a single image.
[20] introduces two end-to-end differentiable solutions for multi-view 3D human pose estimation utilizing techniques that integrate 3D information from multiple 2D perspectives. The first solution, serving as a baseline, employs a basic differentiable algebraic triangulation augmented with confidence weights derived from the input images. The second, more advanced solution, introduces a volumetric aggregation method that combines intermediate 2D backbone feature maps. This aggregated volume is subsequently refined using 3D convolutions to generate final 3D joint heatmaps, effectively modeling a human pose prior.
AdaFuse is an adaptive multiview fusion technique proposed in [79], designed to enhance features in occluded views by utilizing those in visible views. The central concept is to establish point-to-point correspondences between views using epipolar geometry, effectively addressed by exploiting the sparsity of the heatmap representation. Additionally, AdaFuse learns an adaptive fusion weight for each camera view to reflect its feature quality, thereby minimizing the risk of valuable features being compromised by inferior views.
We draw inspiration from AdaFuse for our model, particularly their method of fusion using epipolar geometry. Unlike other models such as [20], [6], and [18] that rely on deep learning for fusion, AdaFuse can operate without the need to train a neural network.

The main distinction between AdaFuse and other models, like the one proposed in [17], lies in their approach to fusion. In [17], the deep learning architecture is essential to the model, specifically designed for fusion purposes. In contrast, AdaFuse uses Sampson distances, heatmaps, and confidence scores as inputs to a neural network that outputs weights for the views, assigning higher importance to more reliable views. This means that AdaFuse can also function without a trained model for fusion by either assigning equal weights to all views or employing alternative methods to compute view weights.

However, our approach differs in several aspects. Post-fusion, we incorporate the robot's structure to refine triangulation, whereas AdaFuse employs standard triangu-

lation without structural guidance. Furthermore, we enhance our results through a bundle adjustment step, which AdaFuse does not include.

4

Proposed Method

In this chapter, the general pipeline of the proposed method is described in detail. Each step of the process will be thoroughly dissected to provide a comprehensive understanding of the methodology. This includes an in-depth exploration of the techniques and algorithms employed at each stage, the rationale behind their selection, and their roles within the overall framework. By the end of this chapter, the reader will have a clear and detailed picture of how the proposed method operates from start to finish.

4.1 Pipeline

Our aim is to estimate the extrinsics parameters of a multi-camera setup by using a robot, in the cameras' field of view, as a calibration pattern and to estimate the pose of said robot with respect to the camera network. The proposed technique, exemplified in Figure 4.1, requires to know the intrinsics of the cameras, a single image per camera and information about the robot's kinematics to return a valid pose. The process is divided into five main steps:

1. **Estimating camera-to-camera poses**: Determine the relative positions and orientations of the cameras using feature matching.

2. **Generating heatmaps and keypoints**: Apply a deep learning model to generate keypoints denoting the robot's joints and their relative heatmaps for each camera from the input images.

3. **Fusing heatmaps**: Combine the heatmaps using epipolar geometry to leverage multi-view information and enhance the original keypoints.

4. **Keypoints structural triangulation**: Use the robot's kinematic information to constrain and guide the triangulation process.

5. **Camera-to-robot pose estimation**: Estimate the rotation and translation between the cameras and the robot using the triangulated points and the 3D points provided by the kinematics.

6. **Bundle Adjustment**: Perform optimization in two stages:

   - First, optimize the camera-to-robot poses using the 3D points provided by the kinematics as ground truth.
   - Next, refine the relative poses between cameras by minimizing the reprojection error between the optimized keypoints and the reprojected points.

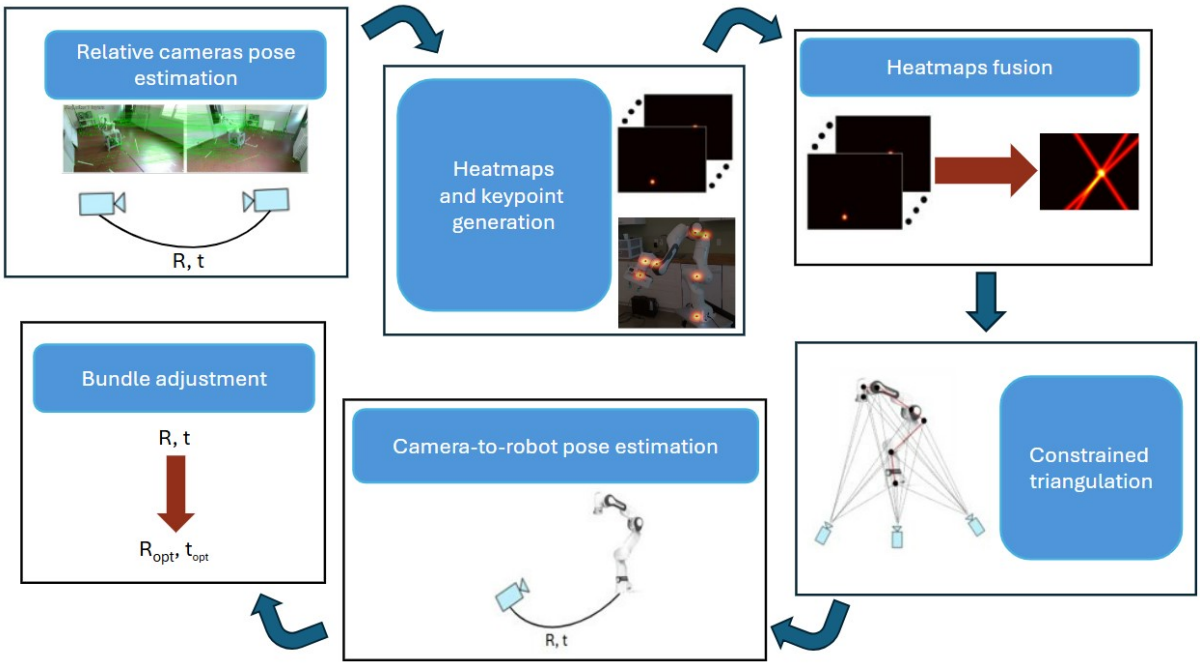

Figure 4.1: Pipeline of our proposed method.

We deliberately avoided using deep learning architectures except for the models used in relative camera calibration and heatmap/keypoint generation. This decision stemmed from the known challenge that deep learning models often struggle to generalize effectively to unseen configurations. Many approaches in human multi-pose estimation rely on neural network architectures for fusion[20, 18, 17], which are susceptible to this limitation. In contrast, our model operates without requiring a specially trained neural network, enabling it to handle unseen camera configurations with the same robustness as familiar ones. This capability will be further demonstrated and discussed in Chapter 5.

4.2 Relative Cameras Pose Estimation

The first step is to recover the structure of the cameras. To this end, similar to procedures in Structure from Motion, we start by detecting and matching features between pairs of images, using these matches to obtain the relative rotation matrix and translation vector between the two cameras.
To estimate poses, we start with feature detection and matching. Given our lack of prior knowledge regarding camera configurations or their proximity to the robot and each other, we avoided classical methods, like ORB [53] or SIFT [38], because they require meticulous fine-tuning to achieve reliable performance. Consequently, we opt for deep learning approaches due to their demonstrated capability to perform effectively across a wide range of configurations and lighting conditions.
In particular our process starts with using LightGlue [35] with SuperPoint [12] as a feature extractor. We chose LightGlue over SuperGlue due to its faster inference time while maintaining almost the same accuracy. Specifically, where SuperGlue takes 70 ms, LightGlue requires only 31.4 ms to detect and match features. The decision to use this detector and matcher is because we verified empirically that they usually provide the highest number of matches. In Figures 4.2, 4.3, 4.4, and 4.5, examples of feature matching using different descriptors, specifically SuperPoint, SIFT, DISK [63], and ALIKED [80, 81], are shown. Table 4.1 presents the numerical values of features detected in both images for the various methods (the maximum value was set to 2048) and the number of matches. We can see that the one with the highest number of matches is the one we chose, while the others are worse, even if the number of detected features is higher with other detectors like SIFT and DISK. This specific case is particularly problematic for SIFT because it returns 0 matches, and maybe with some fine-tuning, it would outperform even SuperPoint. However, what this shows is that SuperPoint works decently even without human intervention, which is exactly what we need.

Method	**No. of Features (Left)**	**No. of Features (Right)**	**No. of Matches**
SuperPoint	1565	1502	**348**
SIFT	1719	1767	0
ALIKED	1232	672	34
DISK	**2048**	**2048**	213

Table 4.1: Comparison of the number of features found by using different detectors and matched using LightGlue.

Since we do not know the initial arrangement of the cameras, we need to determine the best pair of cameras to estimate the relative pose with the highest possible precision.

To achieve this, we construct what we call a 'chain of poses'. In this chain, cameras are represented as nodes and relative poses as edges in a graph.

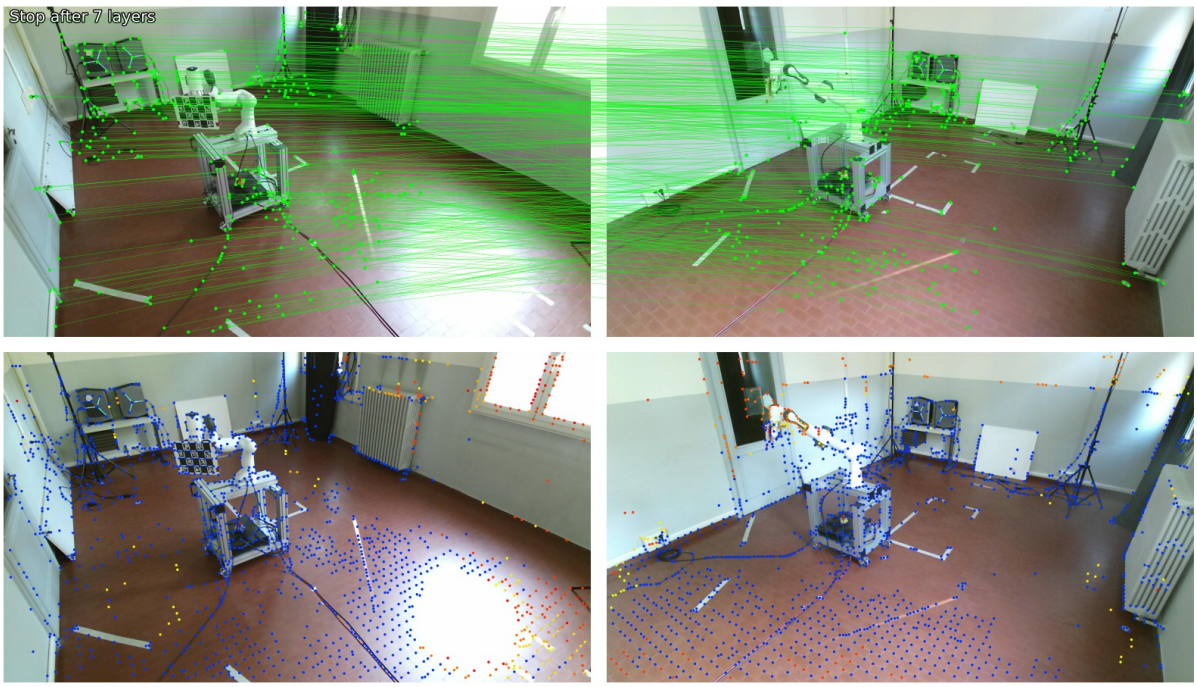

Figure 4.2: Example of feature detection and matching using LightGlue + SuperPoint. The top images show the matches, while the bottom images show the features discarded by LightGlue due to being too dissimilar.

Figure 4.3: Example of feature detection and matching using LightGlue + SIFT. The top images show the matches, while the bottom images show the features discarded by LightGlue due to being too dissimilar.

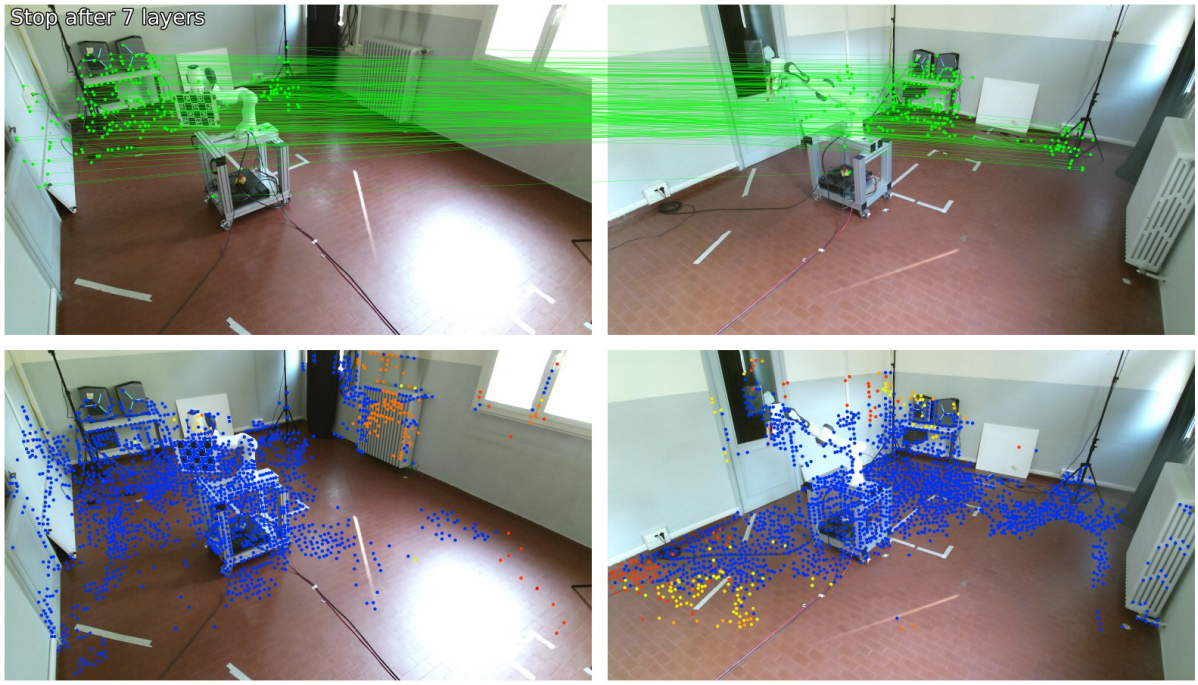

Figure 4.4: Example of feature detection and matching using LightGlue + DISK. The top images show the matches, while the bottom images show the features discarded by LightGlue due to being too dissimilar.

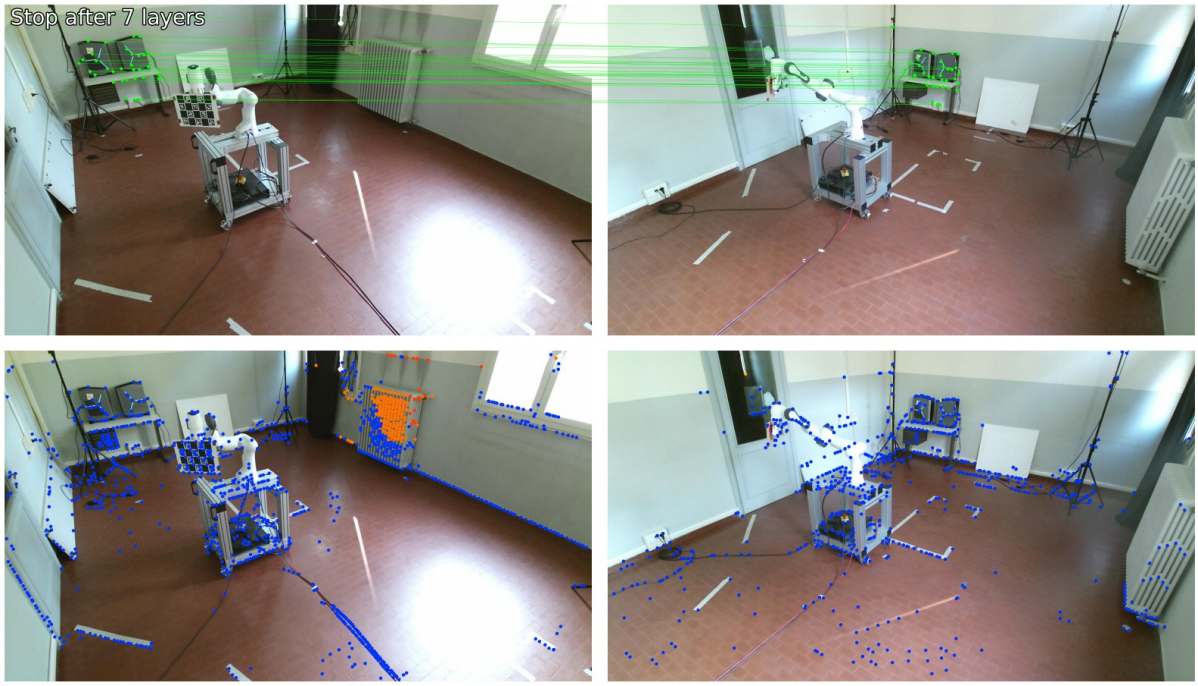

Figure 4.5: Example of feature detection and matching using LightGlue + ALIKED. The top images show the matches, while the bottom images show the features discarded by LightGlue due to being too dissimilar.

The problem then becomes similar to finding the maximum spanning tree, where the weights on the edges are the number of feature matches between image pairs. We

modify the Prim algorithm [49] for our use case. We compute feature matching for each pair of images and select the pair with the highest number of matches. This selection is based on the empirical observation that a higher number of matches correlates with a greater number of inliers, and consequently, a higher likelihood that the estimated pose is accurate or close to accurate. This process is shown in Figure 4.6 with an example.

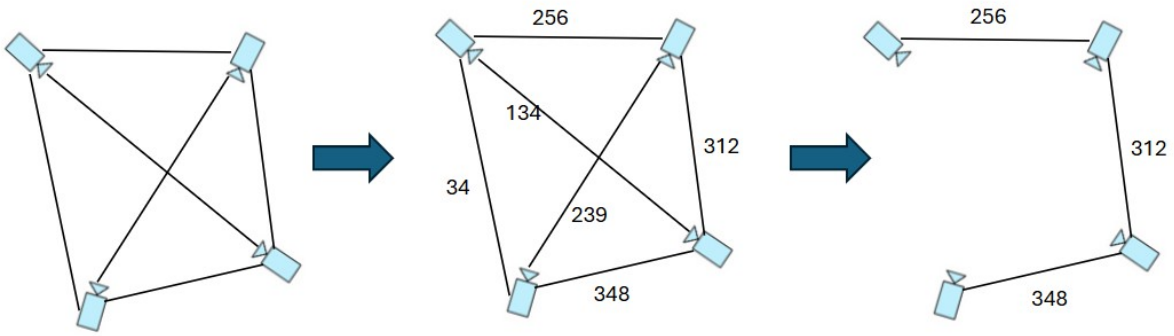

Figure 4.6: Example of the step-by-step construction of what we call the 'chain of poses'. Here, the cameras are nodes, and the relative poses between cameras are edges. The weights on the edges correspond to the number of matched features between the images of the respective cameras.

Once we have matched the features for the two images, we use C2P [61], a solver that obtains the relative pose directly from 2D correspondences. This contrasts with previous methods that solved the problem in two steps: estimating the essential matrix between the views and disambiguating among the four candidate relative poses that satisfy epipolar geometry (called the cheirality check). Using C2P, we obtain a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ such that $\|\mathbf{t}\| = 1$. The reason for this unit norm constraint lies in how the essential matrix $\mathbf{E}$ is computed from corresponding points in normalized image coordinates.
Since $\mathbf{E}$ is derived from normalized coordinates (which typically scales the coordinates to unit magnitude), the resulting translation vector $\mathbf{t}$ extracted from $\mathbf{E}$ also reflects this normalization. Therefore, after solving for $\mathbf{t}$ from $\mathbf{E}$, it inherently maintains a unit norm unless further scaled by an external factor. To correct this, we use an object of known dimensions in the scene: the robot. Knowing the size of the robot in 3D space, we can use the distance between two joints of the robot and triangulate the same two points in the image planes. Once we have the triangulated points, we divide the known distance between the joints by the distance obtained from the triangulated points to obtain a scaling factor. Multiplying this factor by the translation vector gives us the true translations along the axes between the cameras.
While this method works perfectly in theory, in practice, due to outliers, small errors in joint detection, or matching errors, the pose obtained is not perfect. Therefore, we

use bundle adjustment at a later stage to refine the results (Section 4.7). The size of the errors and their impact on the final result will be analyzed in Chapter 5.

4.3 Heatmaps Generation and Keypoint Detection

For each camera, we use a neural network architecture to extract information from the image. We employ DREAM [31], which, given an input RGB image, returns a list of keypoints corresponding to the robot's joints and a heatmap for each detected keypoint, where an heatmap is a 2D belief map where pixel values represent the likelihood that the keypoint is projected onto that pixel. This provides both the keypoints and their corresponding heatmaps for each view.
It is not necessary to use the same model for obtaining heatmaps and keypoints; we choose DREAM because it is currently one of the best models available, known for its fast inference time. However, a strong aspect of our proposed method is its flexibility in not relying on a specific architecture. DREAM can be substituted with any model that outputs heatmaps and keypoints without altering the core structure. This adaptability ensures that future advancements in models can be seamlessly integrated by simply replacing DREAM with a newer and better network.

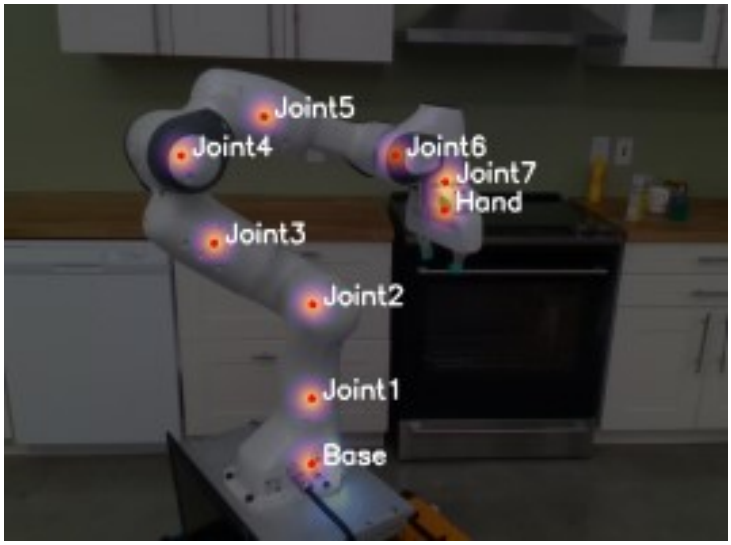

Figure 4.7: Example of keypoints detected by the DREAM model

DREAM also returns the pose using keypoints to solve the PnP problem, but we do not utilize this feature because it processes each image independently, without leveraging the multi-view information and constraints relevant to our scenario.
Therefore, we run DREAM in parallel for each camera in the network. As a result, we obtain keypoints and heatmaps for each robot joint obtained from each camera viewpoint, which will be used for the next step described in Section 4.4.

4.4 Heatmaps Fusion

After obtaining the heatmaps and keypoints, we leverage the multi-view setup to incorporate more information than what is available to single-view models like DREAM. The challenge is how to effectively fuse the multi-view information to improve the results of our deep learning model before estimating the transformation from camera to robot.
Given that there are currently no works focusing on multi-view robot pose estimation, we drew inspiration from multi-view human pose estimation. Specifically, we found Adafuse[79] to be particularly useful. Adafuse uses epipolar geometry constraints to fuse heatmaps, enhancing the results of single-camera human pose estimation, especially in cases of occlusion.

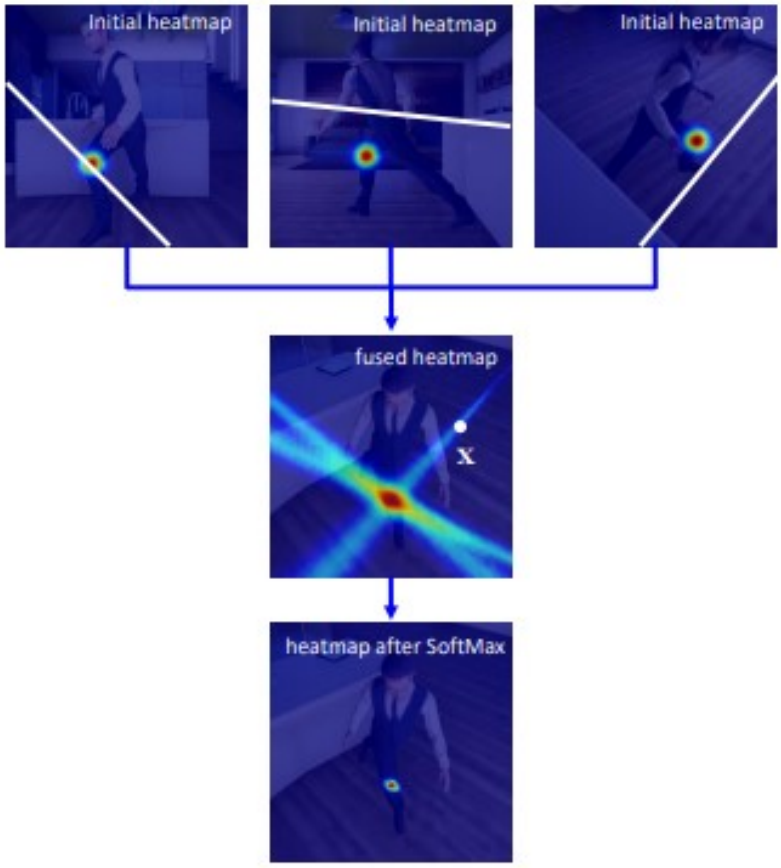

Figure 4.8: Example of heatmap fusion performed by the Adafuse model, which we used as a basis for the fusion step in our model (taken from [79]).

Our model similarly exploits the principles of epipolar geometry. In each view i, every point corresponds to lines in the other views. Heatmaps are grid-like structures that generate per-pixel likelihoods for joint locations, typically represented as two-dimensional Gaussian distributions centered at the joint coordinates. Let us denote the heatmap in view i as H_i. The value at location x of the heatmap is denoted as $H_i(x)$.

The corresponding epipolar line of x in view j is denoted as $I_j(x)$, which comprises discrete locations on the heatmap H_j. For each point x in the heatmap of view i, we select the point with the maximum value along its epipolar line in the other views. Using the following formula, we fuse the heatmaps:

$$\hat{H}_i(x) = \frac{H_i(x) + \sum_{\substack{j=1 \\ j \neq i}}^{N} \max_{x_0 \in I_j(x)} H_j(x_0)}{N},$$

where $\hat{H}_i$ denotes the fused heatmap, and N is the number of camera views contributing to the fusion of the current view.
We repeat this process for every view to create new, fused, heatmaps. To achieve sub-pixel accuracy, we perform a weighted average of the maximum value point and its eight surrounding points.
This fusion process relies on accurate epipolar lines, which depict how a 3D point projects onto the image plane of another camera. This in turn requires precise rotation and translation, because, by definition, epipolar lines are determined by the rotation and translation between the cameras. Since our setup is uncalibrated, the rotation matrices and translation vectors are not perfect, introducing errors.

4.5 Constrained Triangulation

We perform triangulation using the refined points obtained by the previous step simultaneously in all views rather than triangulating each point one at a time, like its done in standard triangulation (Figure 4.9). Doing the latter is not optimal as it disregards the robot's kinematic information specific to our case.

This situation is not unprecedented; other works have explored similar scenarios, such as [8] and [68]. These works discuss how standard triangulation treats each point independently, but propose methods that triangulate all points concurrently, which improves the results. Specifically, [8] introduces a structural constraint based on the human skeleton, and proposes a closed-form solution for triangulation with bone lengths as constraints.

Similarly, our approach involves triangulating all the points simultaneously and utilizing the robot's kinematics to optimize the process. The kinematics provide us with the coordinates of the joints relative to the base joint, allowing us to calculate the distances between the joints (which were already used in the camera relative pose estimation step) like so:

$$d_i = \|\mathbf{v}_i\|, \quad \text{for } i = 1, 2, \ldots, n-1,$$

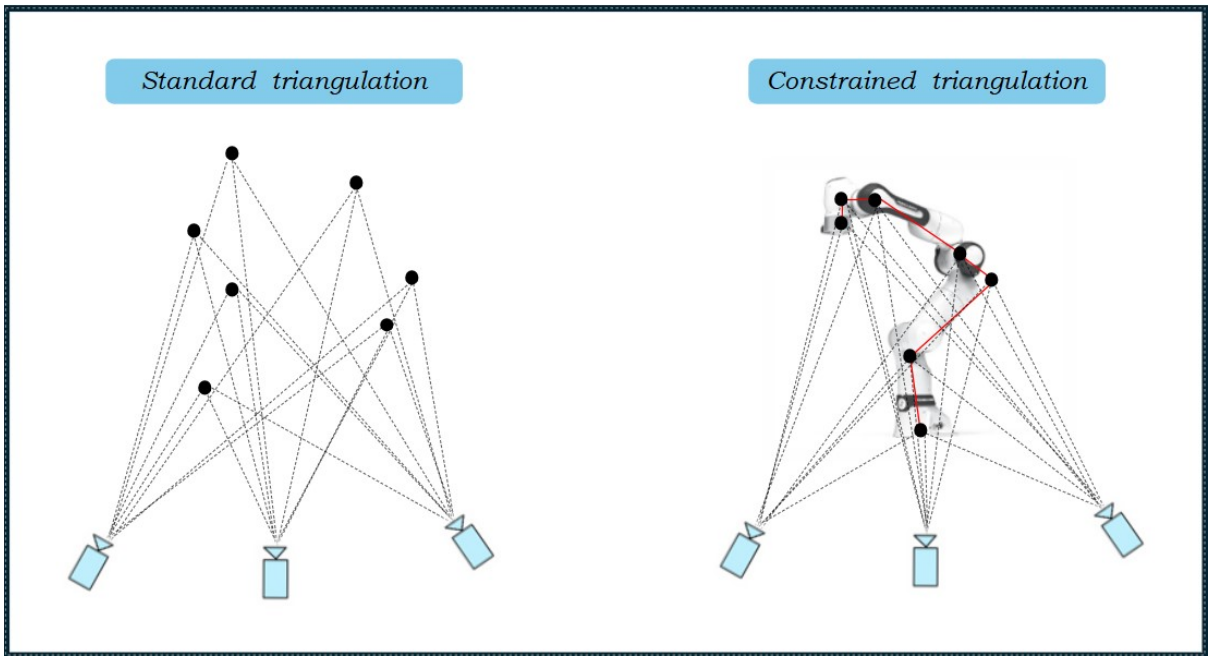

Figure 4.9: Comparison of triangulation methods: on the left, standard triangulation, which triangulates each point individually and on the right, our proposed constrained triangulation method, which incorporates structural constraints of the robot into the process.

where d_i represents the distance between joint $\mathbf{p}_i$ and joint $\mathbf{p}_{i+1}$. Here, $\mathbf{v}_i = \mathbf{p}_{i+1} - \mathbf{p}_i$ denotes the vector from joint $\mathbf{p}_i$ to joint $\mathbf{p}_{i+1}$.

Additionally, knowing the relative positions of the joints, we can construct vectors from one joint to the next, forming a "linked chain" from the base joint to the final joint. This structure allows us to compute the angles between a vector and its successor, yielding $n-2$ angles for n joints, like so:

$$\theta_i = \cos^{-1}\left(\frac{\mathbf{v}_i \cdot \mathbf{v}_{i+1}}{\|\mathbf{v}_i\|\|\mathbf{v}_{i+1}\|}\right), \quad \text{for } i = 1, 2, \ldots, n-2,$$

where θ_i represents the angle between vector $\mathbf{v}_i$ and vector $\mathbf{v}_{i+1}$. Here, $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ are consecutive vectors in the linked chain.

We incorporate this information into our triangulation process as constraints that must not be violated, as the triangulated points should adhere to these constraints, reflecting the robot's structure. Thus, by adding these constraints, the optimization problem that we need to solve becomes:

$$\begin{aligned} \min_{b} \quad & f(b) = \frac{1}{2} b^\top A b - \beta^\top b + d \\ \text{s.t.} \quad & \|b_i\| = L_i, \quad i = 1, 2, \ldots, n-1, \\ & \theta_{j,j+1} = \cos^{-1}\left(\frac{b_j \cdot b_{j+1}}{\|b_j\|\|b_{j+1}\|}\right) = \theta_{gt}, \quad j = 1, 2, \ldots, n-2. \end{aligned} \tag{4.1}$$

where $A \in \mathbb{R}^{3n \times 3n}$ is a symmetric positive semi-definite constant matrix, $\beta \in \mathbb{R}^{3n}$ and $d \in \mathbb{R}$ are constants. A is singular if and only if $\exists i = 0, 1, \ldots, n$, there holds $\forall k_1, k_2 = 1, 2, \ldots, c$, $l_{i,k_1} \parallel l_{i,k_2}$. A, β, d are obtained through the same process proposed in [8] and further discussed in [9]. This triangulation process is repeated for each camera, resulting in a set of triangulated points for each view in the respective camera coordinate system. Each of these set of points will then be used to estimate the pose of the robot with respect to their relative camera frame.

4.6 Camera-to-Robot Pose Estimation

After the triangulation phase, we have a set of triangulated points in each of the cameras' respective coordinate systems. We also have the coordinates of the joints with respect to the base of the robot frame. These points are the same, just in different coordinate systems, or, in other words, they are the same points observed from different perspectives. The idea is that because these sets of points are the same, the difference between them corresponds to the transform between the camera and the robot.
Finding the relative rotation and translation between two sets of corresponding points is a well-known problem and can be solved using a straightforward algorithm (the algorithm described below requires at least three points to work). The problem we want to solve is:

$$RA + t = B$$

where A and B are the two sets of corresponding points, R is a 3×3 rotation matrix, and t is the translation vector. The algorithm can be divided into four main steps:

1. Finding the centroid of both sets
2. Shifting both sets so the center of both is the origin
3. Finding the optimal rotation
4. Finding the optimal translation

For the first step, we compute the centroid coordinates:

$$\text{centroid}_A = \frac{1}{N} \sum_{i=1}^{N} A^i$$

$$\text{centroid}_B = \frac{1}{N} \sum_{i=1}^{N} B^i$$

where A^i and B^i are 3×1 vectors corresponding to the i-th point.
After this, we center the two sets to the origin of the coordinate system, removing the translation component and leaving only the rotation. We chose to use the Kabsch-Umeyama [25, 27, 64] algorithm and Singular Value Decomposition (SVD) to compute the rotation matrix. We compute a matrix H and use SVD as follows:

$$H = (A - \text{centroid}_A)(B - \text{centroid}_B)^T$$

$$[U, S, V] = \text{SVD}(H)$$

$$R = VU^T$$

Theres a special case when finding the rotation matrix that needs to be addressed. Sometimes the SVD will return a reflection matrix, which is numerically correct but physically incorrect. This is addressed by checking the determinant of R (from SVD above) and seeing if its negative. If it is, then the third column of V is multiplied by -1. After this, all that remains is to find the translation:

$$R \times A + t = B$$

$$R \times \text{centroid}_A + t = \text{centroid}_B$$

$$t = \text{centroid}_B - R \times \text{centroid}_A$$

This works perfectly if the points are exactly the same. In practice, due to errors, the triangulated points are not perfect, so the SVD actually minimizes the least squares error:

$$\text{err} = \sum_{i=1}^{N} \left\| RA^i + t - B^i \right\|^2$$

4.7 Bundle Adjustment

The final step of the pipeline is to refine the pose obtained from all the previous steps. This process is divided into two phases: first, optimizing the poses between the robot and the cameras, and then optimizing the relative poses between the cameras.
In the first phase, we consider the robot's joints and their relative coordinates obtained from the kinematics as the ground truth. The goal is to formulate an optimization

problem aimed at refining parameters such as the 2D points, rotation matrices, and translation vectors by minimizing the reprojection error. This involves iteratively optimizing the 2D points alongside the rotation matrices and translation vectors. The objective is to ensure that the 2D points align as closely as possible with their observed reprojected counterparts across multiple views.
Let:

- $\mathbf{R}_{cr}$ and $\mathbf{t}_{cr}$ be the rotation matrix and translation vector between camera c and the robot,
- $\mathbf{P}_r$ be the 3D points obtained from the robot's kinematics,
- $\mathbf{p}_c$ be the 2D reprojected points onto the camera c image plane.

The reprojection error e can be defined as:

$$e = \sum_{i=1}^{N} \left\| \mathbf{p}_c^i - \pi(\mathbf{R}_{cr}\mathbf{P}_r^i + \mathbf{t}_{cr}) \right\|^2$$

where $\pi(\cdot)$ denotes the projection function from 3D to 2D. The optimization problem aims to minimize this error:

$$\min_{\mathbf{R}_{cr},\mathbf{t}_{cr}} e = \sum_{i=1}^{N} \left\| \mathbf{p}_c^i - \pi(\mathbf{R}_{cr}\mathbf{P}_r^i + \mathbf{t}_{cr}) \right\|^2$$

In the second phase, we use one of the newly optimized poses to transform the coordinate system of the points obtained from the robot's kinematics into the respective camera coordinate system. Then, we iteratively optimize the 2D keypoints, rotation matrices, and translation vectors for each camera, as well as the 3D points, by minimizing the reprojection error to refine the relative poses.

Let:

- $\mathbf{R}_{c1c2}$ and $\mathbf{t}_{c1c2}$ be the relative rotation matrix and translation vector between cameras $c1$ and $c2$,
- $\mathbf{P}_{c1}$ be the 3D points in camera $c1$'s coordinate system,
- $\mathbf{p}_{c2}$ be the 2D reprojected points in camera $c2$.

The reprojection error e in this context can be defined as:

$$e = \sum_{i=1}^{N} \left\| \mathbf{p}_{c2}^i - \pi(\mathbf{R}_{c1c2}\mathbf{P}_{c1}^i + \mathbf{t}_{c1c2}) \right\|^2$$

The optimization problem for this phase aims to minimize the reprojection error across all cameras:

$$\min_{\mathbf{R}_{c1c2},\mathbf{t}_{c1c2}} e = \sum_{i=1}^{N} \left\| \mathbf{p}_{c2}^{i} - \pi(\mathbf{R}_{c1c2}\mathbf{P}_{c1}^{i} + \mathbf{t}_{c1c2}) \right\|^2$$

After this process, we obtain the refined poses of the cameras with respect to the robot and the refined relative poses between them.

5 Experiments and Results

To verify the effectiveness of our approach, we conducted several experiments. All the experiments were performed in a simulated environment due to the simplicity and speed of setting up the environment, as well as the greater precision in obtaining ground truth information compared to real-world environments. In this chapter, we will test our method against several criteria. We will demonstrate that by leveraging multi-view information, it can achieve better results than its single-camera counterpart. We will test the model with different numbers of cameras and various configurations, such as cameras positioned close to or distant from the robot.
Additionally, we will conduct an ablation study on the various components of our model and report the results.

5.1 Testing Setup

The software developed is primarily based on the Robot Operating System (ROS)[1], a flexible framework designed to simplify the development of complex and robust robot software. ROS is middleware that provides essential services such as hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.
ROS is particularly advantageous for our project as it facilitates rapid and straightforward testing of our model. This is especially true due to its convenient setup for simulated testing environments. ROS integrates a rich set of tools for simulation, visualization, and debugging, enabling quick adjustments to the number and configuration of cameras, the robot's placement, and its configuration.

[1] https://www.ros.org/

One of the tools we extensively utilized is Gazebo[2], a powerful open-source simulation tool designed to accurately and efficiently simulate robots in complex environments. Its integration with ROS enables seamless communication between the simulation and ROS nodes, making it an essential tool for the development and testing of our model. Using Gazebo, we mitigate risks typically associated with real-world scenarios and minimize errors caused by sensor noise or other issues, thanks to the controlled environment of the simulation. By providing a realistic and flexible simulation environment, Gazebo helps reduce development time while enhancing the reliability and performance of robotic systems.

Figure 5.1: Franka Emika Panda robot arm

In our setup, we used the Franka Emika Panda[3] (Figure 5.1), a collaborative robot arm designed for precision, safety, and ease of use. The Panda is equipped with seven degrees of freedom, allowing it to perform a wide range of tasks with high flexibility and dexterity. Its advanced control features and intuitive programming interface make it suitable for both industrial applications and research purposes.
For capturing color information, we used the Kinect v2 sensor (Figure 5.2). It is equipped with a high-definition (HD) color camera, an infrared (IR) emitter, and an array of IR sensors, which together enable the capture of detailed color images.

The method was tested in a simulated environment using Gazebo, with the Franka Emika Panda robot and the Kinect v2 sensors. To ensure consistency and reliability in our experiments, we first set up the environment and registered a ROS bag of the robot moving in various configurations for each type of experiment. Each ROS bag was then

[2]https://gazebosim.org/home
[3]https://franka.de/

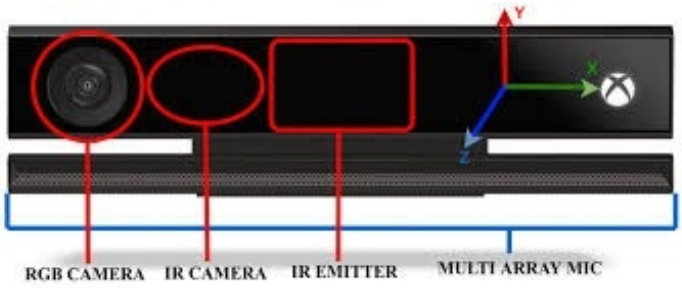

Figure 5.2: Kinect v2 camera

replayed 3 times for its relative experiment and we selected 100 random frames from the bag each run for testing, ensuring that the results were consistent across different runs. To make the scenario more realistic, random objects were added to the simulated environment in addition to the robot, as illustrated in Figure 5.3 and 5.4, so that features could also be extracted by LightGlue[35] from the external environment surrounding the robot.
However, this is not an issue for real-world applications, as it is highly unlikely that only the robot would be present in the scene. Even in real-world scenarios, the floor and other environmental features provide sufficient points for feature matching.
The code used to process the images was developed primarily using OpenCV[4], a powerful open-source computer vision library. It provides a wide range of tools and functions for image processing, feature detection, and image analysis. The code was developed in Python, on a machine running Ubuntu 20.04, equipped with a 16GB RAM and 6GB GPU. This setup ensured that we had the necessary computational power and software environment to effectively develop and test our methods.

5.2 Comparison with Single-Camera Model

Firstly, we began by comparing our proposed method with a single-camera model like DREAM [31], which was applied independently across multiple cameras. This comparison allowed us to assess whether our model enhances the results obtained from a single-view perspective by integrating multi-view information.
Specifically, we evaluated our method against the pose estimated by DREAM using its detected keypoints and the robot's joints to solve the PnP problem across all cameras. Figures 5.5 and 5.6 illustrate the rotation and translation errors, respectively.
To estimate the goodness of the results, we utilize two metrics: one for the rotation component and one for the translation component. We refer to them as the rotation error and the translation error.

[4]https://opencv.org/

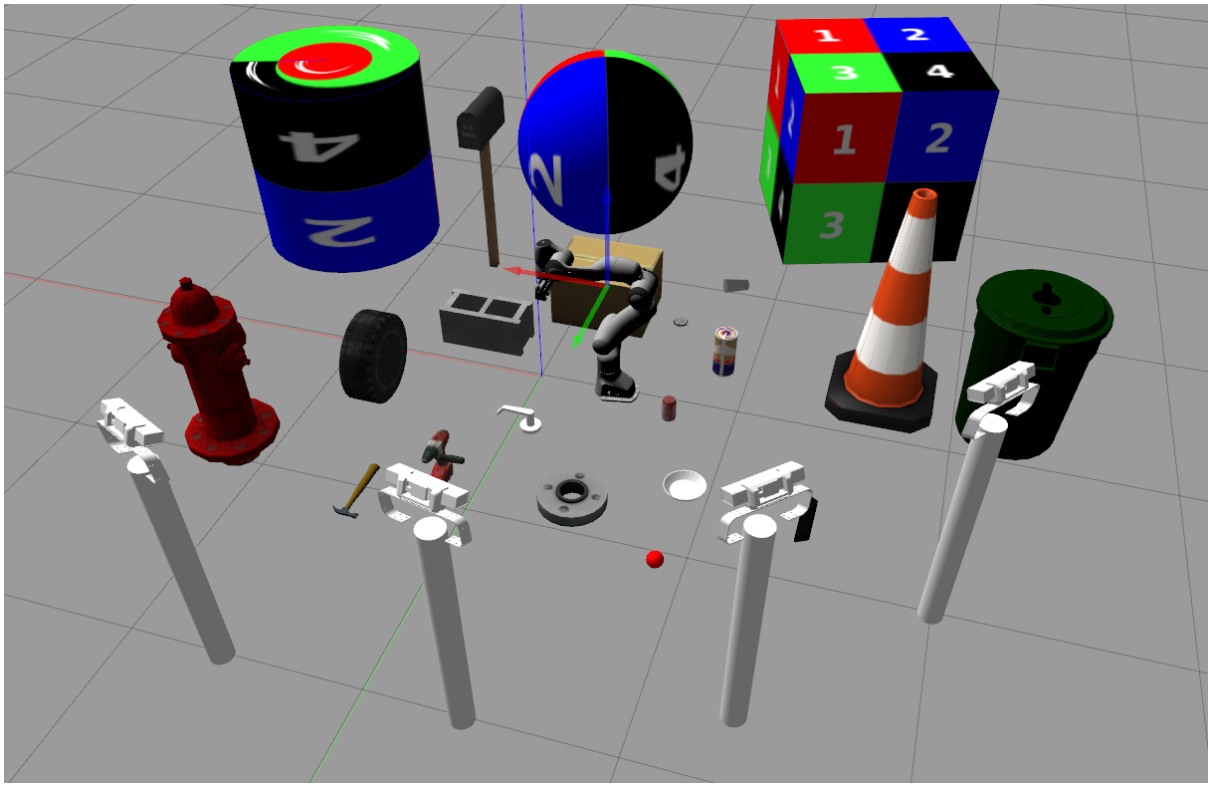

Figure 5.3: Setup used to evaluate our model against the single-camera model.

They are computed as follows:

Rotation Error:

1. Compute Displacement Matrix:

$$R_{\text{error}} = R_{\text{model}} \cdot R_{\text{ground truth}}^{-1}$$

where:

- R_{model} is the rotation matrix returned by the model,
- $R_{\text{ground truth}}$ is the ground truth rotation matrix.

2. Convert Displacement Matrix to Euler Angles:

$$(\theta_{\text{error, x}}, \theta_{\text{error, y}}, \theta_{\text{error, z}}) = \text{Euler}(R_{\text{error}})$$

Here, $\text{Euler}(R_{\text{error}})$ denotes the conversion of the rotation matrix R_{error} to Euler angles (e.g., using ZYX convention).

3. Average Euler Angles:

$$\theta_{\text{error}} = \frac{\theta_{\text{error, x}} + \theta_{\text{error, y}} + \theta_{\text{error, z}}}{3}$$

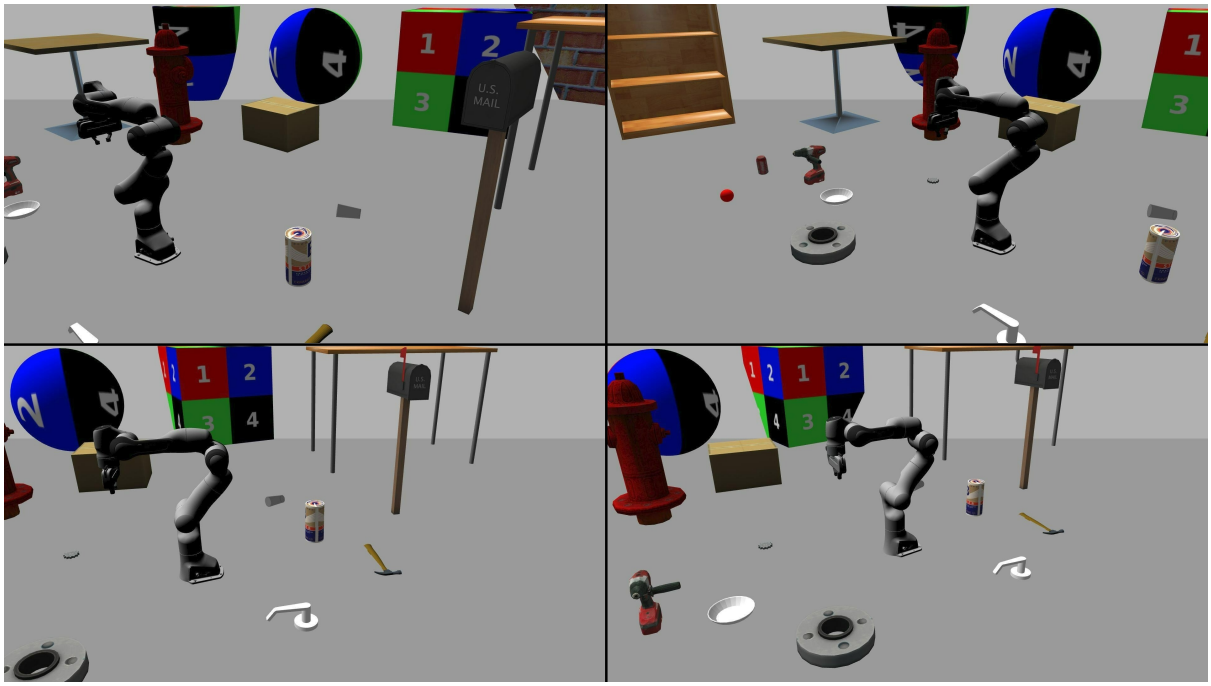

Figure 5.4: Scene viewed by the 4 cameras with the setup of Figure 5.3.

This yields a single measure of rotation error θ_{error}.

Translation Error:

- Compute Euclidean Distance:

$$\text{translation\_error} = \|\mathbf{t}_{\text{model}} - \mathbf{t}_{\text{ground truth}}\|$$

where:

  - $\mathbf{t}_{\text{model}}$ is the translation vector calculated by our method,
  - $\mathbf{t}_{\text{ground truth}}$ is the ground truth translation vector.

The rotation error, expressed in degrees, and the translation error, in meters, were evaluated across four cameras using our proposed method and the DREAM model. The results were obtained from a recorded rosbag that captured all the necessary topics for the system to run. From Figures 5.5 and 5.6, the following key observations emerge:

- **Rotation error comparison:** Across all four cameras, our method consistently achieves lower median rotation errors compared to the DREAM model. Specifically, the median rotation error for our method ranges from approximately 2.9° to 3.2°, whereas for DREAM, it spans from 3.1° to 4.7°. This translates to a notable improvement of 7% to 38% in rotation accuracy with our method compared to DREAM. Moreover, our method exhibits a smaller interquartile range (IQR) for rotation errors, ranging from approximately 4.97° to 5.05°, compared to

DREAM's wider IQRs ranging from approximately 5.58° to 8.22°. The reduced variability indicates higher stability and reliability in our method's rotation estimations, crucial for applications requiring precise spatial orientation.

- **Translation error comparison:** Similarly, in terms of translation errors, our method consistently outperforms DREAM across all cameras. The median translation error for our method ranges from approximately 0.22 to 0.25 meters, whereas DREAM shows higher median errors ranging from about 0.24 to 0.37 meters. This signifies a significant improvement of 5% to 40% in translation accuracy with our method. The IQR for translation errors also favors our method, ranging from approximately 0.27 to 0.32 meters, compared to DREAM's wider IQRs ranging from approximately 0.35 to 0.96 meters. These findings underscore the robustness and consistency of our method in accurately determining positional shifts in robotic environments.

- **Distribution spread (IQR):** The smaller IQR for both rotation and translation errors with our method compared to DREAM across all cameras highlights the superior consistency and reliability of our approach. This narrower spread in error values indicates that our method not only achieves lower median errors but also maintains higher precision in its predictions, crucial for applications demanding high accuracy.

- **Maximum and minimum errors:** Our method generally demonstrates lower maximum errors in rotation across all cameras compared to DREAM, indicating superior robustness in minimizing extreme error cases. However, for translation errors, while our method exhibits lower maximum errors overall, Camera 1 shows a slightly higher maximum error compared to DREAM. This specific case highlights the impact of varying data quality among cameras, where our method's performance might be affected by poorer quality inputs from other views.

From Figure 5.5, we observe that the rotation error for our method is consistently lower than that of the DREAM model across all cameras. For example, Camera 3 shows a median rotation error of 3.2° with our method compared to 4.3° for DREAM, reflecting an improvement of around 25%. The smaller IQR of our method, such as 4.97° for Camera 2, compared to DREAM's 7.83°, signifies that our method's predictions are more stable and exhibit less variation. This consistency is crucial for high-precision applications as it suggests fewer unexpected deviations in the estimated rotations.

Additionally, the presence of fewer extreme results in our method compared to DREAM implies greater robustness to anomalies and extreme cases. For instance, the maximum rotation error for Camera 4 is 12.32° with our method, significantly lower

than DREAM's 22.61°, highlighting the reduced occurrence of high errors with our approach. The DREAM method's longer whiskers in the box plots reveal a higher likelihood of producing extreme error values, which can be detrimental in critical scenarios requiring utmost accuracy.

In Figure 5.6, illustrating translation errors, our method demonstrates a lower median translation error across all cameras. For example, Camera 2 exhibits a median translation error of 0.23 meters with our method versus 0.37 meters for DREAM, indicating a 38% improvement. The reduced IQR for translation errors, such as 0.27 meters for Camera 3 with our method versus 0.51 meters for DREAM, further highlights our method's reliability and consistency.

An interesting observation is that for Camera 1, the maximum translation error for our method is slightly higher than that of the DREAM method (0.82 meters versus 0.74 meters). This could be attributed to poorer quality information from other cameras affecting the estimation process for Camera 1. This phenomenon suggests that while multi-view information generally improves performance, the quality of the contributing views must be managed effectively.

Unlike Adafuse [79], our model assigns equal weights to all views in the estimation process. Future work could involve a weighted approach, assigning different weights to each view based on their quality to enhance performance, especially when some cameras provide lower quality data. This approach could mitigate the negative impact of poorer quality views on the overall estimation process.

Overall, the errors of our model are quite consistent between cameras, indicating that the system effectively refines poses by leveraging multi-view information. In contrast, the single-camera model (DREAM) struggles, likely due to difficulties in extracting information from the scene, such as occlusions.

Table 5.1: Rotation error results of our method and the DREAM model for each of the 4 cameras, expressed in degrees.

Camera	Method	Mean	Median	IQR	Maximum	Minimum
Camera 1	DREAM	11.9762	3.12837	6.03347	15.8176	**0.00253463**
	Ours	**6.39162**	**2.90266**	**5.05149**	**13.6725**	0.0162856
Camera 2	DREAM	13.8073	4.72184	7.82561	21.4264	0.0435054
	Ours	**6.44195**	**3.14077**	**4.96794**	**12.9928**	**0.0162856**
Camera 3	DREAM	13.6011	4.26559	5.57691	15.7272	0.0363425
	Ours	**6.57177**	**3.2469**	**4.99643**	**13.6725**	**0.0162856**
Camera 4	DREAM	13.5596	4.51157	8.21908	22.611	**0.00675646**
	Ours	**6.46327**	**3.01727**	**4.97681**	**12.3224**	0.0162884

Table 5.2: Translation error results of our method and the DREAM model for each of the 4 cameras, expressed in meters.

Camera	Method	Mean	Median	IQR	Maximum	Minimum
Camera 1	DREAM	0.761274	**0.24194**	0.352065	**0.738411**	**0.0219761**
	Ours	**0.447935**	0.246548	**0.27498**	0.814737	0.0296835
Camera 2	DREAM	0.988929	0.369559	0.96176	2.56828	0.0410574
	Ours	**0.447053**	**0.225923**	**0.318978**	**0.895663**	**0.024438**
Camera 3	DREAM	0.945339	0.349948	0.51416	1.06865	0.0340042
	Ours	**0.458141**	**0.224357**	**0.291363**	**0.698099**	**0.0290697**
Camera 4	DREAM	0.942284	0.331758	0.52876	1.44742	0.0276484
	Ours	**0.48591**	**0.245465**	**0.25358**	**0.744495**	**0.0228487**

5.3 Impact of the Amount of Cameras

We tested our model with different numbers of cameras to verify if the number of cameras influences the results, as we expect, and to see if the model correctly utilizes the additional information obtained from multiple cameras. This section provides a detailed analysis of the translation and rotation error distributions for the proposed multi-view pose estimation model across different numbers of cameras. We tested our model with 3, 4, and 5 cameras configuration, as shown in Figure 5.7.

5.3.1 Camera-to-Robot Errors

We start by considering the camera-to-robot errors. As can be seen from Figures 5.8 and 5.9, the following detailed observations can be made:

- **Rotation error comparison:** The median rotation error values are quite similar across different numbers of cameras. For the 3-camera setup, the median rotation error is approximately 3.39°, for the 4-camera setup, it is about 3.31°, and for the 5-camera setup, it is around 3.25°. This slight improvement with more cameras indicates that additional views marginally enhance the rotation accuracy. The percentage decrease in median rotation error from 3 cameras to 5 cameras is about 4%.

- **Translation error comparison:** The median translation error also shows minor variations across different numbers of cameras. For 3 cameras, the median translation error is approximately 0.245 meters, for 4 cameras, it is around 0.239 meters, and for 5 cameras, it decreases to about 0.233 meters. This suggests a small improvement in translation accuracy with more cameras, with a percentage

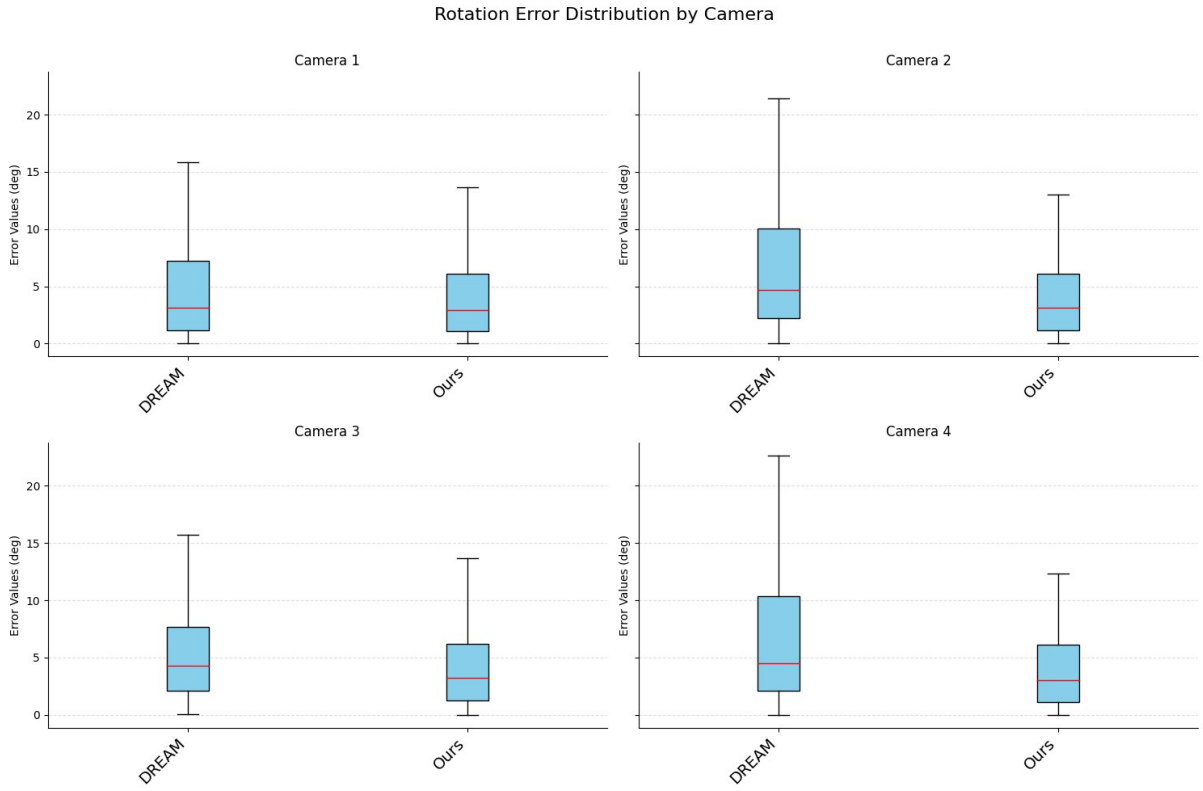

Figure 5.5: Rotation error of the proposed method compared with the DREAM model

decrease in median translation error from 3 cameras to 5 cameras being about 5%.

- **Distribution spread (IQR):** The IQR for rotation errors remains consistent, with values such as 4.07° for 3 cameras, 3.99° for 4 cameras, and 3.91° for 5 cameras. For translation errors, the IQR is 0.264 meters for 3 cameras, 0.257 meters for 4 cameras, and 0.251 meters for 5 cameras. This stability in IQR values across different numbers of cameras indicates that the error distribution's spread does not significantly change, suggesting robustness in the model's performance.

- **Maximum and minimum errors:** The maximum rotation error decreases slightly with more cameras, from 12.25° for 3 cameras to 11.77° for 5 cameras. Similarly, the maximum translation error decreases from 0.743 meters for 3 cameras to 0.705 meters for 5 cameras. The minimum error remains stable and low across all configurations, indicating that the model's lower bound of performance is consistent.

From Figures 5.8 and 5.9, we observe that the translation and rotation error distributions are relatively stable across different numbers of cameras. The minor improvements in median errors and the slight reduction in maximum errors suggest that adding more cameras helps in mitigating high-error outliers, enhancing the model's

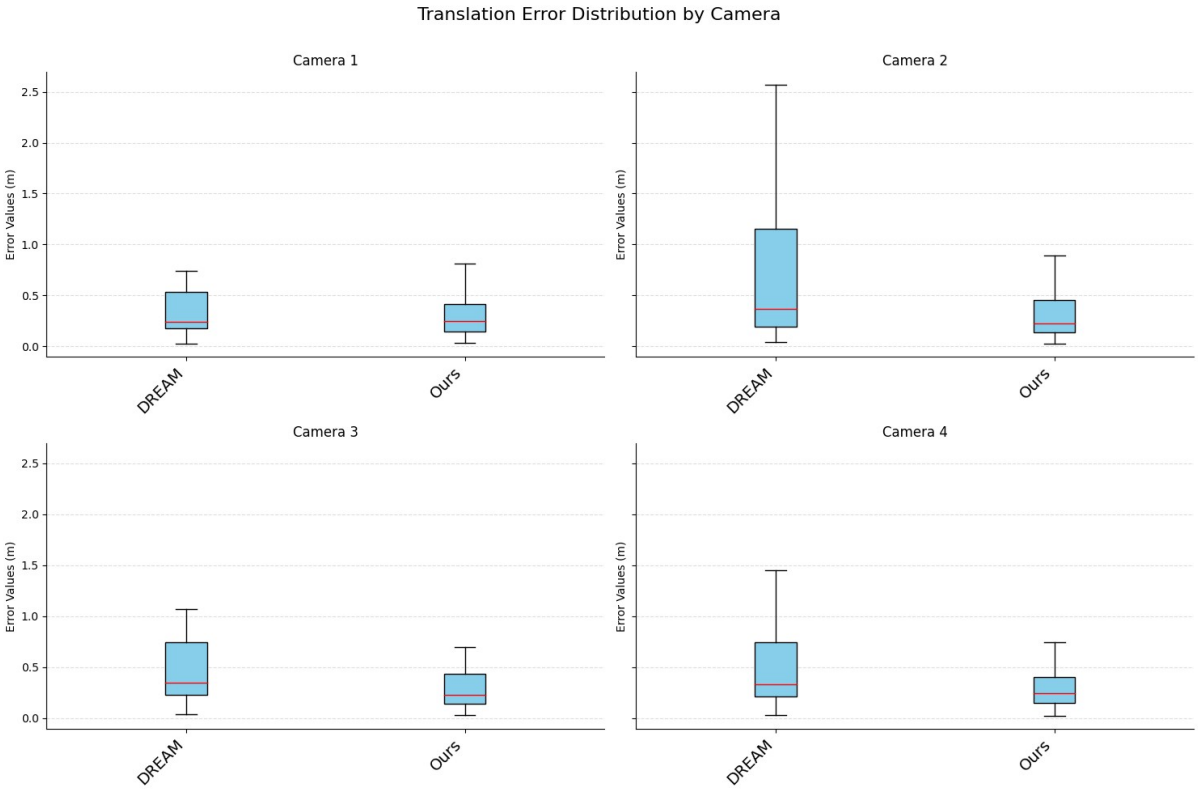

Figure 5.6: Translation error of the proposed method compared with the DREAM model

robustness.

The consistent IQR values across different camera setups indicate that the model performs reliably regardless of the number of cameras used. This suggests that the model can effectively utilize the information from multiple views, maintaining stable performance even as the number of cameras increases.

An interesting point is the slight reduction in maximum errors with more cameras. For example, the maximum rotation error decreases by about 4% when increasing from 3 to 5 cameras, and the maximum translation error decreases by about 5%. This indicates that additional views help in better handling critical cases, which is crucial for ensuring reliability in real-world applications.

Table 5.3: Rotation error results of the camera-to-robot pose estimation when using 3, 4 and 5 cameras, expressed in degrees.

Method	Mean	Median	IQR	Maximum	Minimum
3 cams	6.60943	3.38553	4.07412	12.252	0.47983
4 cams	6.46715	3.31265	3.98642	11.9882	0.469501
5 cams	**6.35074**	**3.25302**	**3.91467**	**11.7724**	**0.46105**

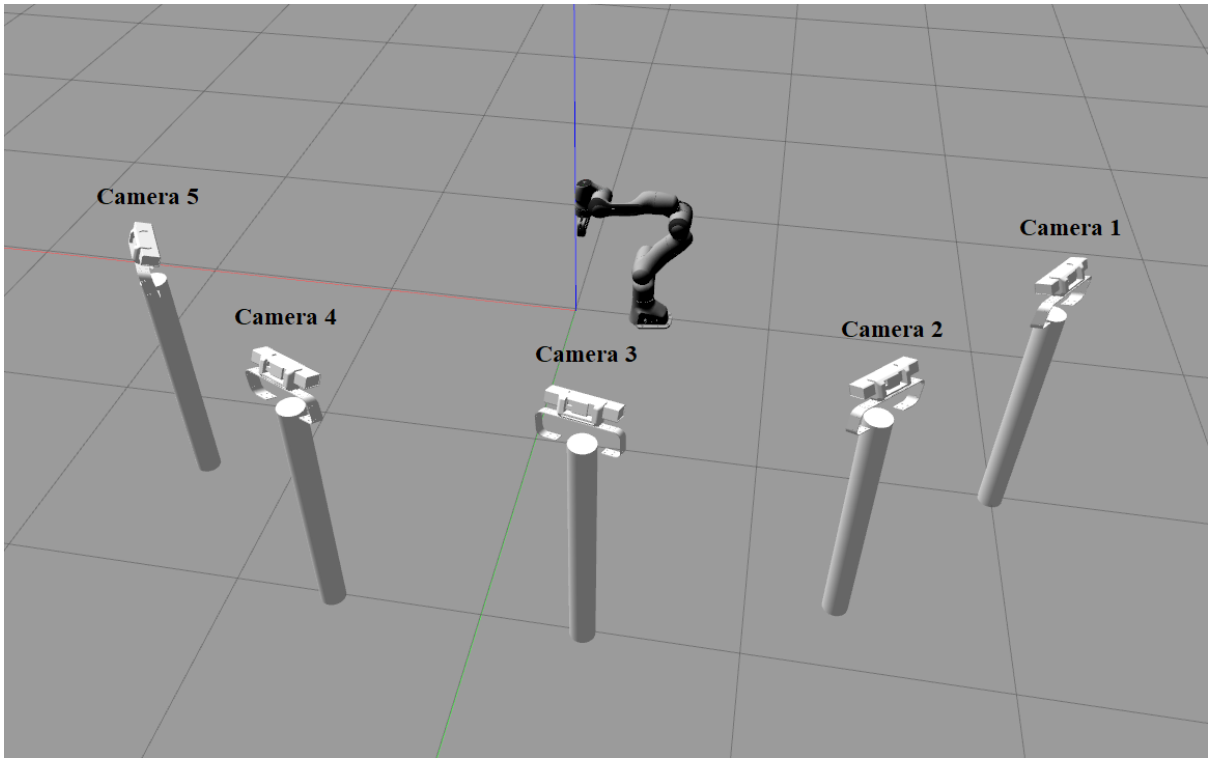

Figure 5.7: Cameras placement for the test on the number of cameras. The model was evaluated considering only camera 1 through 3 for the 3 cams, considering camera 1 through 4 for 4 cams, and considering all the cameras for 5 cams

Table 5.4: Translation error results of the camera-to-robot pose estimation when using 3, 4 and 5 cameras, expressed in meters.

Method	Mean	Median	IQR	Maximum	Minimum
3 cams	0.476586	0.245414	0.264281	0.742502	0.0380742
4 cams	0.464057	0.238962	0.257333	0.722981	0.0370732
5 cams	**0.452455**	**0.232988**	**0.2509**	**0.704907**	**0.0361464**

5.3.2 Camera-to-Camera Errors

We also analyze the camera-to-camera error distributions. Referring to Figures 5.10 and 5.11, we can see that:

- **Rotation error comparison:** The median rotation error values are consistent across different numbers of cameras. For the 3-camera setup, the median rotation error is approximately 1.63°, for the 4-camera setup, it is about 1.61°, and for the 5-camera setup, it is around 1.59°. The slight improvement in rotation accuracy with more cameras is indicated by a 3% decrease in median rotation error from 3 cameras to 5 cameras.

- **Translation error comparison:** The median translation error is also consistent across different numbers of cameras. For 3 cameras, the median translation error

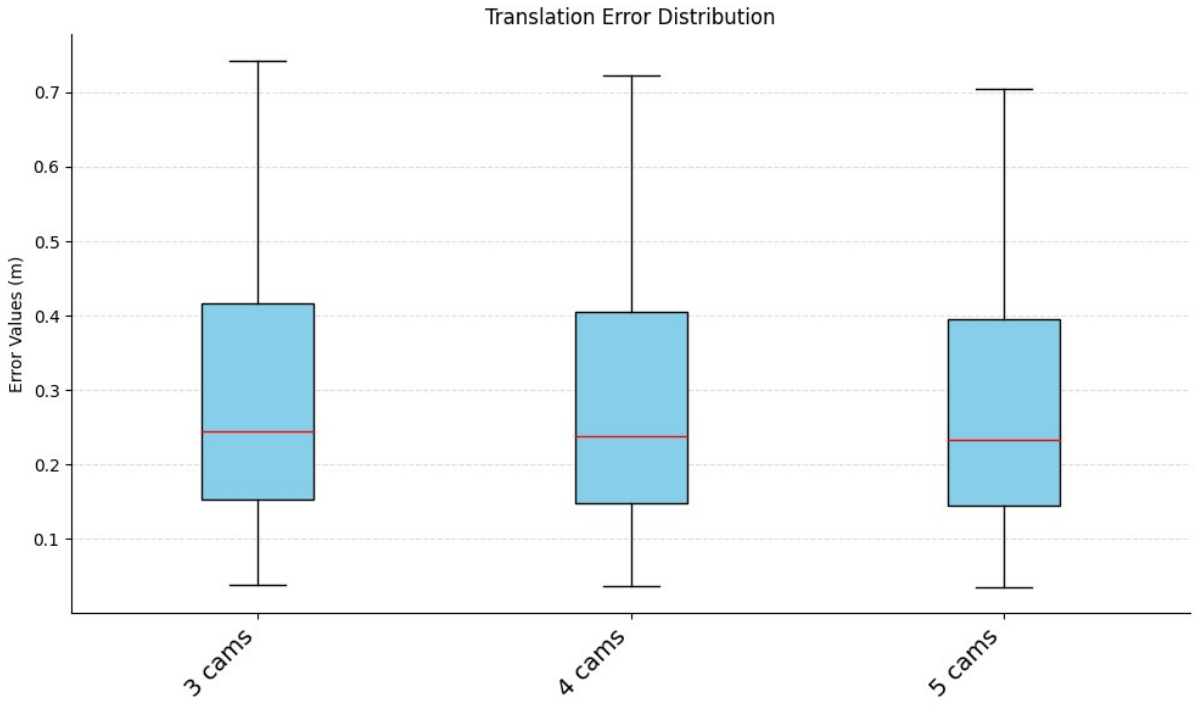

Figure 5.8: Translation error distribution for 3, 4, and 5 cameras of the camera-to-robot pose using the proposed method

is approximately 0.223 meters, for 4 cameras, it is around 0.219 meters, and for 5 cameras, it decreases to about 0.216 meters. This minor improvement in translation accuracy is highlighted by a 3% decrease in median translation error from 3 cameras to 5 cameras.

- **Distribution spread (IQR):** The IQR for rotation errors is similar across all configurations, with values such as 1.97° for 3 cameras, 1.94° for 4 cameras, and 1.91° for 5 cameras. For translation errors, the IQR is 0.240 meters for 3 cameras, 0.236 meters for 4 cameras, and 0.233 meters for 5 cameras. This consistency in IQR values indicates that the spread of the middle 50% of error values does not vary significantly with the number of cameras.

- **Maximum and minimum errors:** The maximum rotation error decreases slightly with more cameras, from 5.91° for 3 cameras to 5.74° for 5 cameras. Similarly, the maximum translation error decreases from 0.674 meters for 3 cameras to 0.654 meters for 5 cameras. The minimum error remains stable and low across all configurations, suggesting consistent lower bounds of performance.

Both the translation and rotation error distributions indicate that the proposed multi-view pose estimation model performs consistently well regardless of the number of cameras used. The median errors remain stable, with rotation errors around 1.6° and translation errors approximately 0.22 meters. The variability of errors, as shown

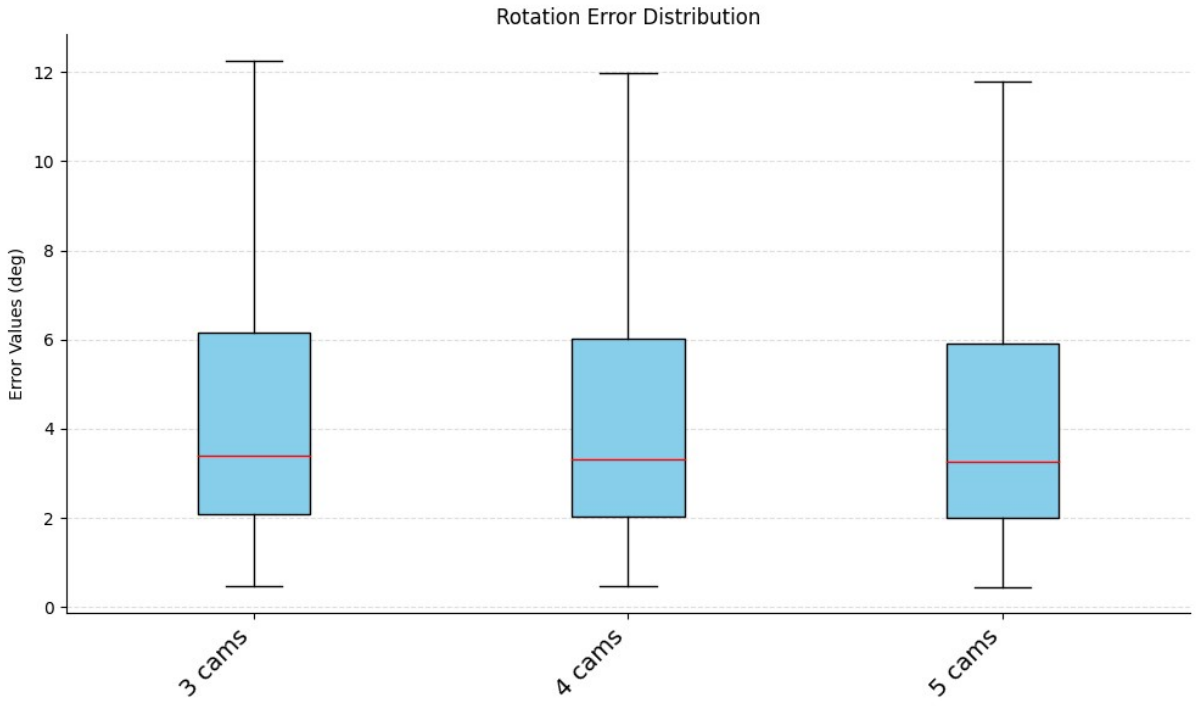

Figure 5.9: Rotation error distribution for 3, 4, and 5 cameras of the camera-to-robot pose using the proposed method

by the IQR, also remains consistent, demonstrating the model's robustness.

The slight reduction in maximum errors with more cameras, such as the decrease in maximum rotation error from 5.91° to 5.74° and in maximum translation error from 0.674 meters to 0.654 meters, suggests that additional views can help reduce high-error outliers, enhancing the model's overall reliability.

This outcome aligns with our expectations because, for camera-to-camera pose estimation, the only step that utilizes multi-camera information is the final bundle adjustment phase. Therefore, we anticipated that adding more cameras would not significantly impact the results. There is a slight improvement, possibly due to obtaining a better camera-to-robot pose that is then used for the bundle adjustment. However, the marginal improvement in camera-to-robot accuracy with the addition of more cameras was less significant than expected, highlighting the model's robust performance with even fewer cameras.

These findings underscore the model's robustness and dependable performance, making it a suitable choice for applications requiring precise and stable pose estimation from multiple camera perspectives.

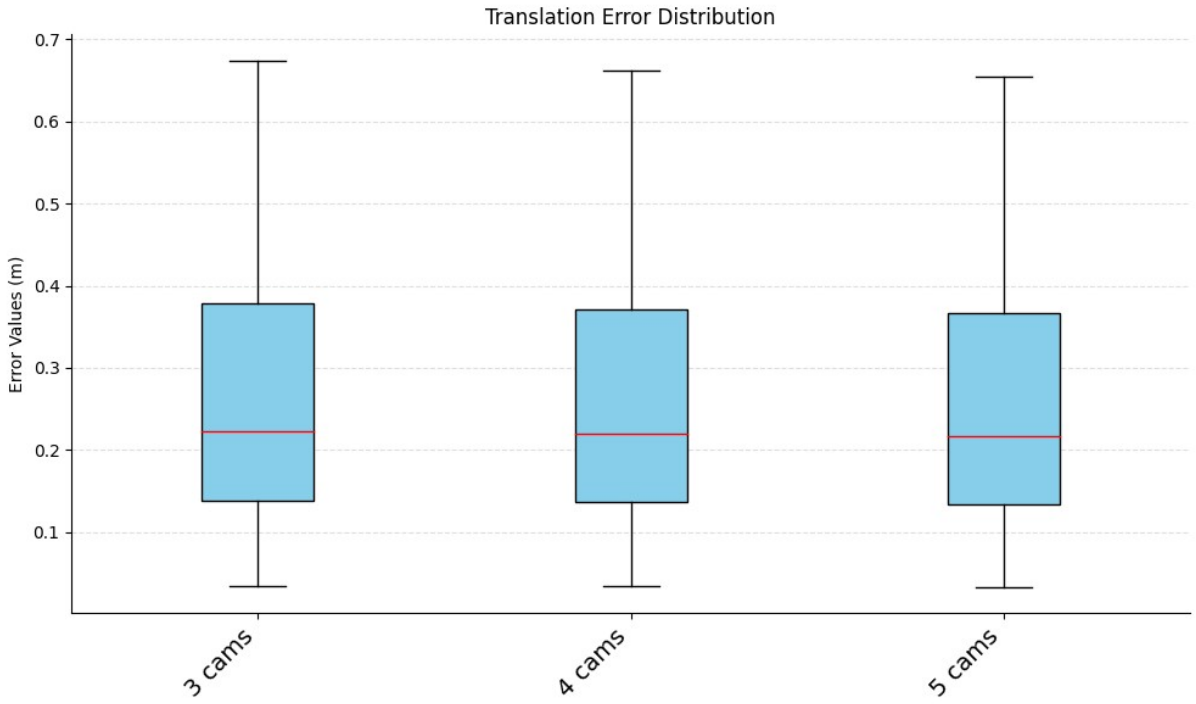

Figure 5.10: Translation error distribution for 3, 4, and 5 cameras of the camera-to-camera pose using the proposed method

Table 5.5: Rotation error results of the camera-to-camera pose estimation when using 3, 4 and 5 cameras, expressed in degrees.

Method	Mean	Median	IQR	Maximum	Minimum
3 cams	3.19006	1.63403	1.96639	5.91345	0.231591
4 cams	3.14292	1.60989	1.93733	5.82605	0.228169
5 cams	**3.09891**	**1.58735**	**1.9102**	**5.74449**	**0.224974**

5.4 Analysis of Camera-to-Robot Distance

Another aspect we wanted to test was the resilience of our model to changes in the disposition of the cameras in the space. In particular, we wanted to know how the distance to the robot affected the results. This is because the DREAM model was trained only with images where the robot was at a distance between 75 to 120 centimeters from the camera. This might negatively influence its output, thereby degrading the final result because it relies on the DREAM output. Therefore, we compared the results when the cameras are at a distance of a maximum of 1.5 meters from the robot (close) and when they are at a distance greater than 2 meters from the robot (distant). Both cameras configurations are shown in Figure 5.12 and 5.13 respectively.

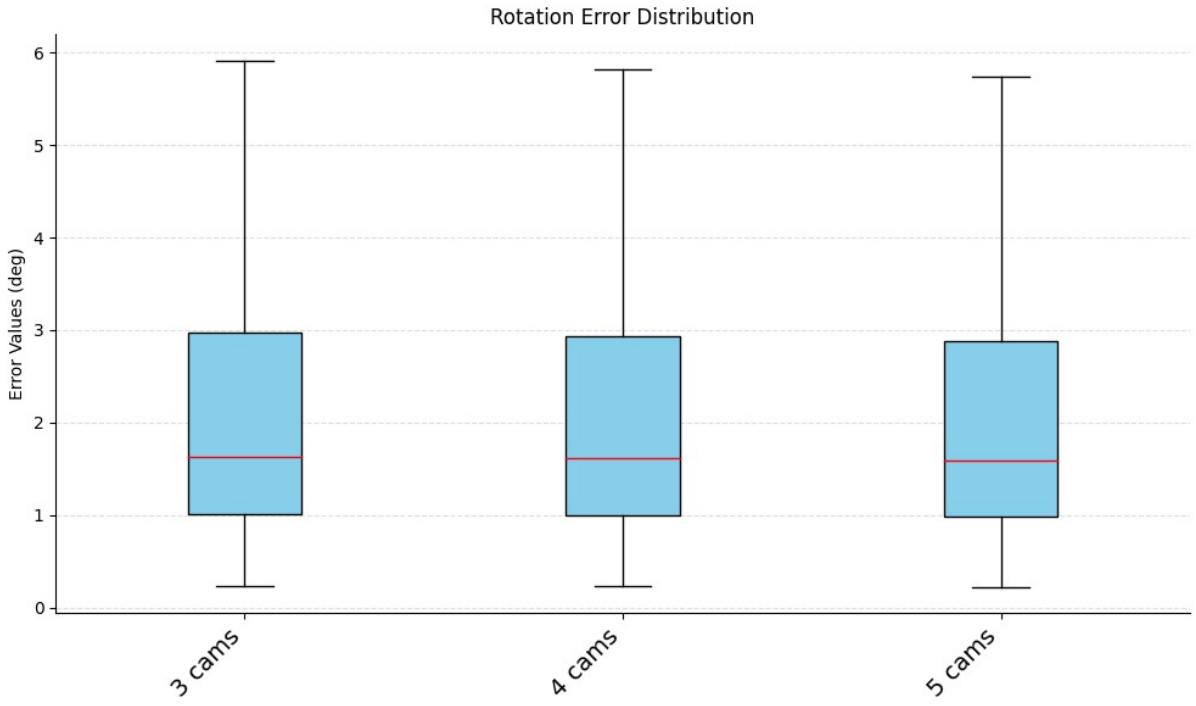

Figure 5.11: Rotation error distribution for 3, 4, and 5 cameras of the camera-to-camera pose using the proposed method

Table 5.6: Translation error results of the camera-to-camera pose estimation when using 3, 4 and 5 cameras, expressed in meters.

Method	Mean	Median	IQR	Maximum	Minimum
3 cams	0.432617	0.222773	0.239899	0.673999	0.0345615
4 cams	0.425385	0.219049	0.235889	0.662733	0.0339838
5 cams	**0.419855**	**0.216201**	**0.232822**	**0.654117**	**0.033542**

5.4.1 Camera-to-Robot Pose Error

We start by testing how the distance from the robot affects the camera-to-robot pose estimation process.
From Figures 5.15 and 5.14, we can draw the following observations:

- **Rotation error comparison:** The median rotation error is approximately 3.25° for close proximity cameras and 3.67° for distant cameras. This indicates a 12.9% increase in the median rotation error when using distant cameras, suggesting that rotation accuracy is slightly compromised with increased distance.

- **Translation error comparison:** The median translation error is about 0.237 meters for close proximity cameras and 0.258 meters for distant cameras. This represents an 8.9% increase in median translation error with distant cameras, implying a modest degradation in translation accuracy as the distance increases.

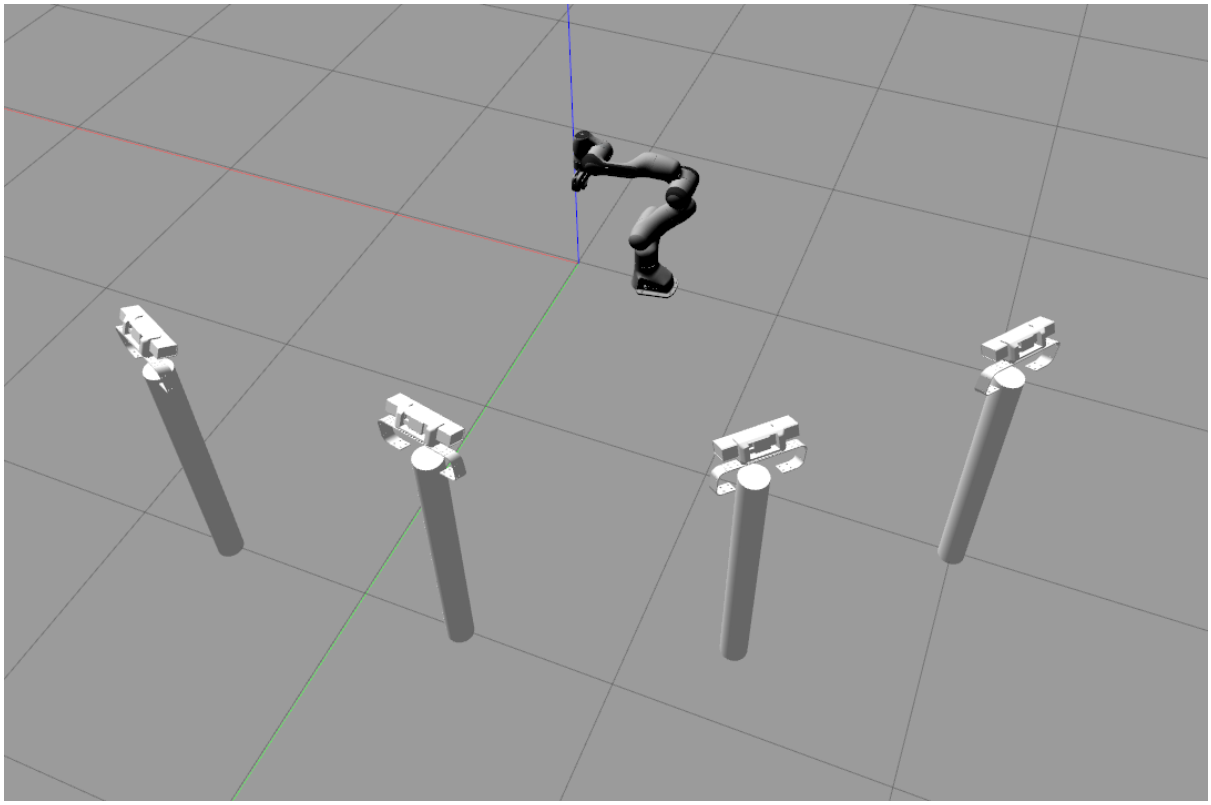

Figure 5.12: Cameras configuration for the 'Close' case

- **Distribution spread (IQR):** The IQR for rotation errors is 3.91° for close proximity cameras and 4.42° for distant cameras, indicating a wider spread in errors with increased distance. For translation errors, the IQR is 0.255 meters for close proximity cameras and 0.278 meters for distant cameras, suggesting a similar increase in error variability with distance.
- **Maximum and Minimum Errors:** The maximum rotation error is higher for distant cameras (13.29°) compared to close proximity cameras (11.76°). The maximum translation error also increases from 0.716 meters for close proximity cameras to 0.781 meters for distant cameras. The minimum errors remain low and similar in both cases, indicating consistent lower bounds of performance.

Both the translation and rotation error distributions show that the proposed pose estimation model performs differently across different camera proximities. The median errors remain somewhat stable, with rotation errors around 3.25° for close proximity and 3.67° for distant cameras, and translation errors approximately 0.237 meters for close proximity and 0.258 meters for distant cameras. The variability of errors (as indicated by the IQR) is comparable for both configurations, but close proximity cameras have an edge in performance.

The broader range of errors for distant cameras suggests that the model's performance in handling robot configurations could be improved with closer camera

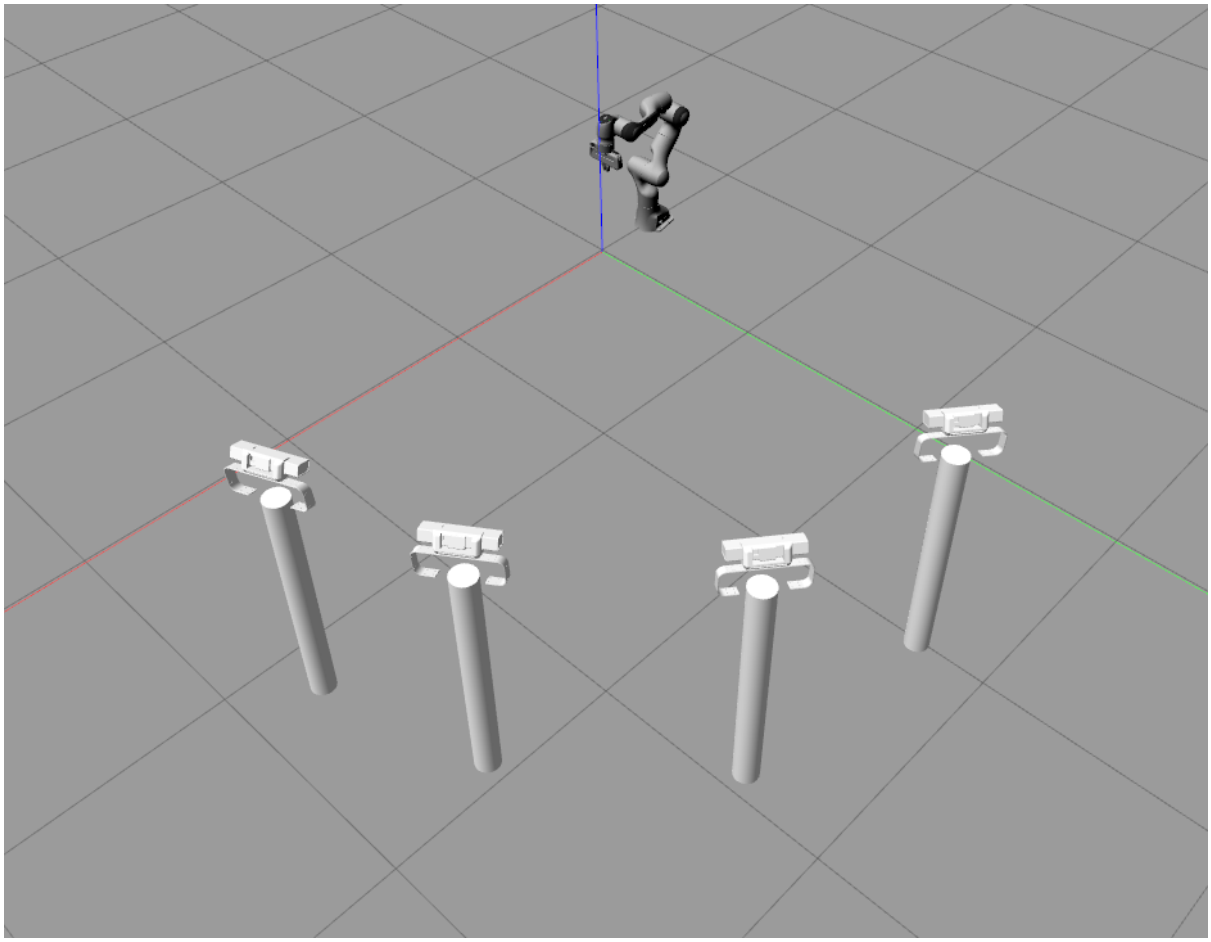

Figure 5.13: Cameras configuration for the 'Distant' case

placement. For instance, the maximum rotation error increases from 11.76° for close proximity cameras to 13.29° for distant cameras, and the maximum translation error rises from 0.716 meters to 0.781 meters. This trend indicates that while the model remains robust, the increased distance introduces more variability and higher outliers in error.

This is likely due to the DREAM model that performs the 2D keypoint detection, which has worse results the farther it is from the robot. These findings highlight the importance of camera placement in maintaining high accuracy and reliability in pose estimation.

Table 5.7: Rotation error results of the camera-to-robot pose estimation when the distance to the robot changes, expressed in degrees.

Method	Mean	Median	IQR	Maximum	Minimum
Close	**6.34627**	**3.25073**	**3.91191**	**11.7641**	**0.460725**
Distant	7.17129	3.67333	4.42046	13.2935	0.52062

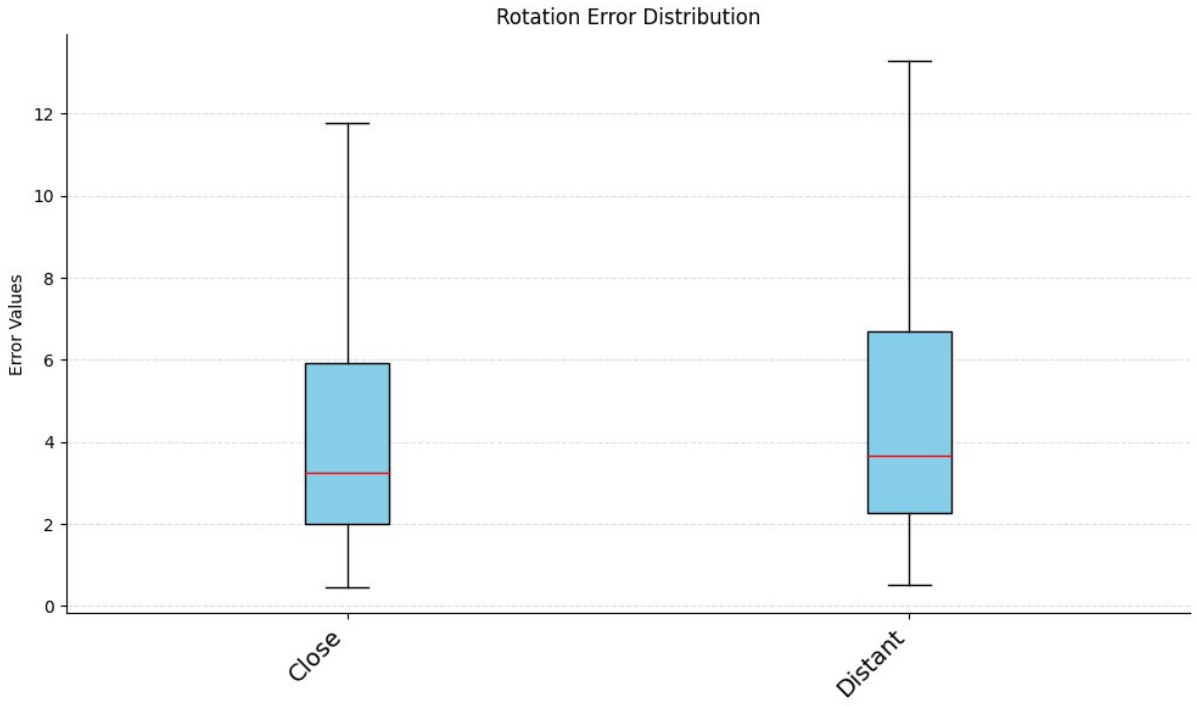

Figure 5.14: Distribution of rotation errors in the camera-to-robot poses using the proposed method. 'Close' cameras refer to those closer to the robot, while 'Distant' cameras are farther away.

Table 5.8: Translation error results of the camera-to-robot pose estimation when the distance to the robot changes, expressed in meters.

Method	Mean	Median	IQR	Maximum	Minimum
Close	**0.45976**	**0.23675**	**0.25495**	**0.716287**	**0.0367299**
Distant	0.501138	0.258057	0.277896	0.780753	0.0400356

5.4.2 Camera-to-Camera Pose Error

We also wanted to analyze how the distance from the robot impacts the relative camera poses. From Figures 5.16 and 5.17, we derive the following observations:

- **Rotation Error Comparison:** The median rotation error is approximately 1.55° for close proximity cameras and 1.46° for distant cameras. This shows a slight decrease of about 5.8% in the median rotation error when using distant cameras, indicating slightly improved rotation accuracy with increased camera distance.

- **Translation Error Comparison:** The median translation error is about 0.223 meters for close proximity cameras and 0.219 meters for distant cameras. This represents a minor reduction of approximately 1.8% in median translation error with distant cameras, suggesting similar translation accuracy across different camera proximities.

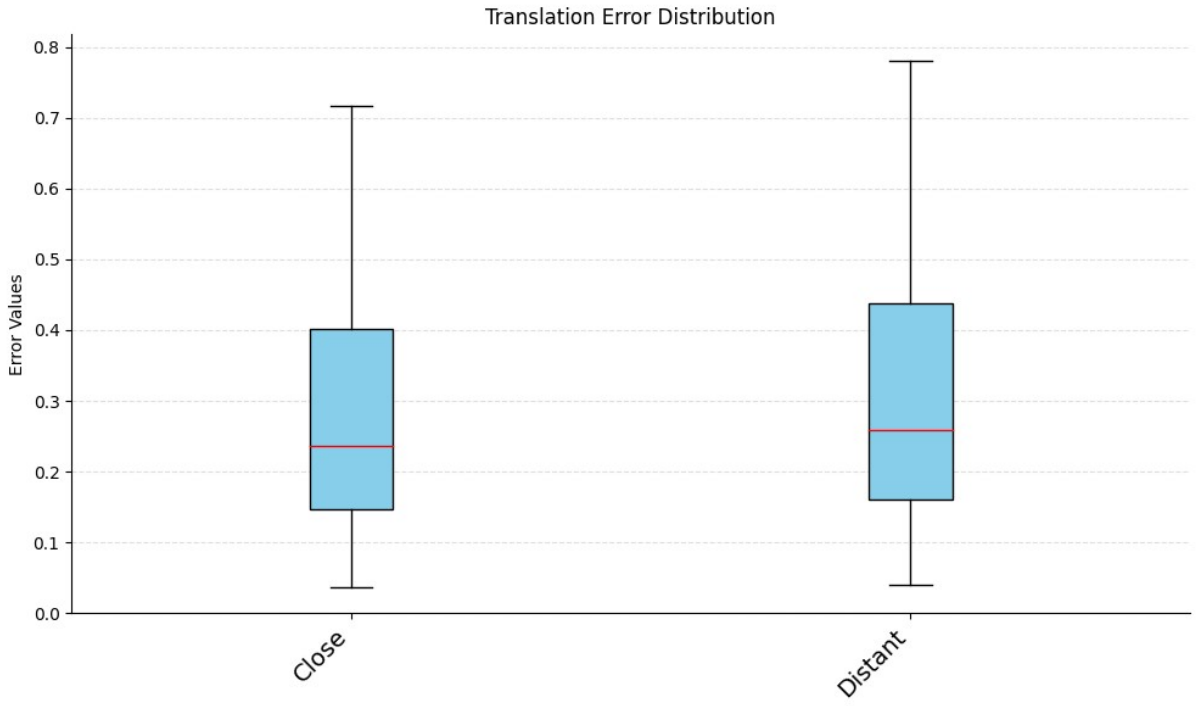

Figure 5.15: Distribution of translation errors in the camera-to-robot poses using the proposed method. 'Close' cameras refer to those closer to the robot, while 'Distant' cameras are farther away.

- **Distribution Spread (IQR):** The IQR for rotation errors is 1.86° for close proximity cameras and 1.75° for distant cameras, indicating a slightly narrower spread in errors with increased distance between cameras. For translation errors, the IQR is 0.241 meters for close proximity cameras and 0.235 meters for distant cameras, suggesting comparable error variability across different camera distances.
- **Maximum and Minimum Errors:** The maximum rotation error is 5.60° for close proximity cameras and 5.27° for distant cameras. The maximum translation error is 0.676 meters for close proximity cameras and 0.661 meters for distant cameras. Both rotation and translation errors show slightly lower maximum errors with distant cameras compared to close proximity cameras, indicating fewer extreme outliers with increased camera distance.

Both the translation and rotation error distributions indicate that the proposed pose estimation model performs comparably across different camera proximities. The median errors remain stable, with rotation errors around 1.55° for close proximity and 1.46° for distant cameras, and translation errors approximately 0.223 meters for close proximity and 0.219 meters for distant cameras. The variability of errors (as indicated by the IQR) is also comparable for both configurations.

The slight reduction in median rotation and translation errors with distant cameras suggests that increased camera distance may provide a small advantage in improving

accuracy. This advantage likely stems from Lightglues ability to detect and match more features in the larger field of view provided by distant cameras. Empirical verification during the initial relative camera pose estimation phase of our approach demonstrates that increased feature matches lead to more inliers and improved performance of the C2P solver[61]. This explains why the difference in performance between near and distant cameras is more pronounced in the rotation error distribution. In contrast, the translation error difference is less evident, possibly because we compensate for scale variation in the translation vector obtained from the solver using keypoints from DREAM, which degrade as cameras move farther away from the robot, as inferred from our earlier analysis.

Testing different camera distances was not conducted because it was anticipated that the model's performance might decrease with greater distances between cameras. This is due to reduced scene similarity between cameras capturing the scenes, as detailed in prior research [35, 54]. Given that our model relies on scene similarity in the initial processing steps, any degradation in similarity would likely result in errors throughout the pose estimation process.

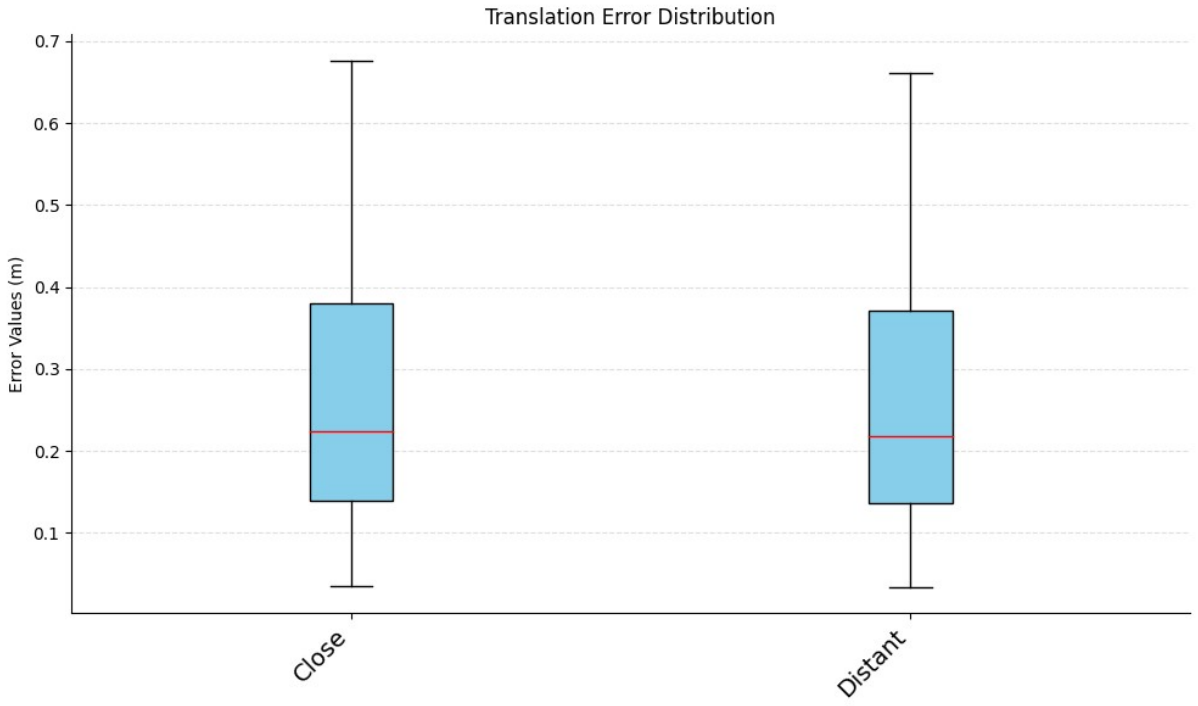

Figure 5.16: Distribution of translation errors in the camera-to-camera poses using the proposed method. 'Close' cameras refer to those closer together, while 'Distant' cameras are farther apart.

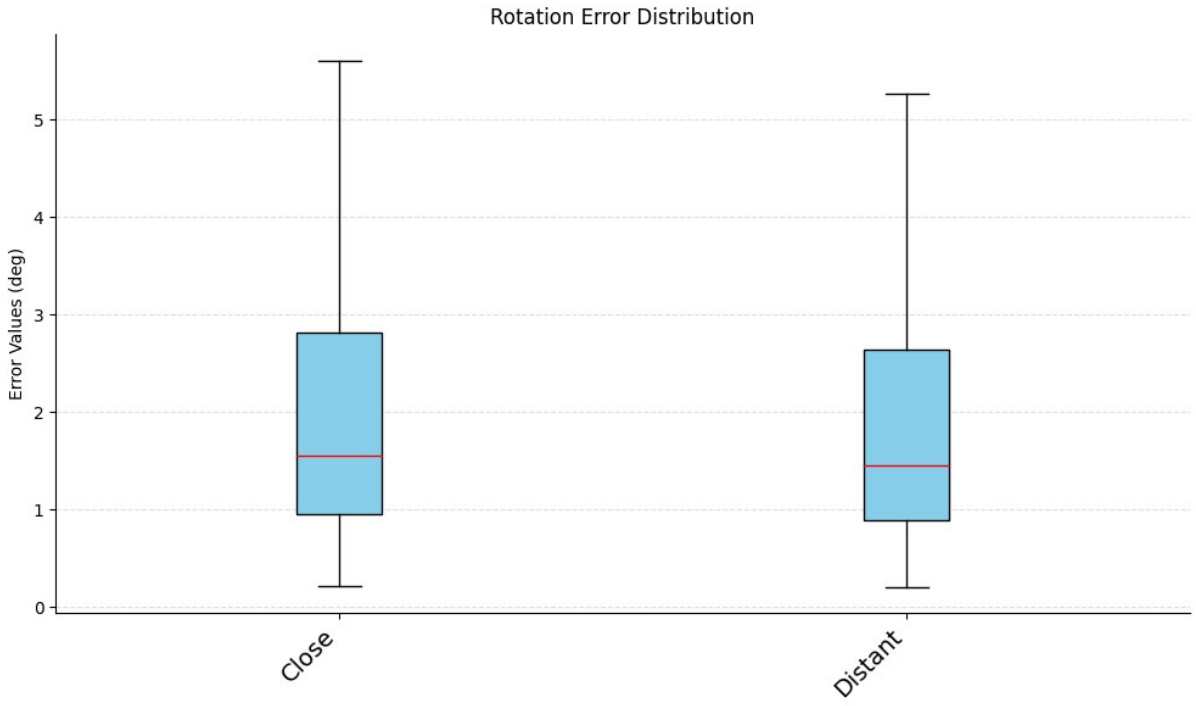

Figure 5.17: Distribution of rotation errors in the camera-to-camera poses using the proposed method. 'Close' cameras refer to those closer together, while 'Distant' cameras are farther apart.

Table 5.9: Rotation error results of the camera-to-camera pose estimation when the distance to the robot changes, expressed in degrees.

Method	Mean	Median	IQR	Maximum	Minimum
Close	3.02203	1.54797	1.86281	5.60198	0.219393
Distant	**2.84071**	**1.45509**	**1.75105**	**5.26586**	**0.206229**

5.5 Ablation Study

As the final test, we conducted an ablation study on our model to systematically analyze the impact of each step in our process on the final outcome. We compared our complete method against variations where one specific step was removed while keeping all other steps unchanged. This approach allowed us to understand the individual contribution of each step to the overall performance of our model.

5.5.1 Camera-to-Robot Pose Analysis

We analyze the performance of different methods for estimating the camera-to-robot pose by comparing their rotation and translation errors. Our method includes all steps of the process, and the results are presented in Figure 5.19 and Figure 5.18. 'CC' refers to the method using calibrated cameras, omitting the initial step that employs

Table 5.10: Translation error results of the camera-to-camera pose estimation when the distance to the robot changes, expressed in meters.

Method	Mean	Median	IQR	Maximum	Minimum
Close	0.433979	0.223474	0.240654	0.676122	0.0346703
Distant	**0.424431**	**0.218558**	**0.23536**	**0.661247**	**0.0339076**

feature matching to estimate the relative poses between cameras. 'ST' uses normal triangulation instead of the constrained triangulation explained in Section 4.5. 'NF' refers to the model without the heatmap fusion step. Finally, 'WBA' is the model without the last two-stage refinement step.

The ablation study reveals significant insights into the performance of each method:

- **Calibrated Cameras (CC):** Calibrated Cameras consistently demonstrate the lowest median errors across all tested cameras. For instance, using Camera 2, this method shows a mean rotation error of 6.02°, a median of 3.32°, and an IQR of 3.60°. The mean translation error is 0.42 meters, with a median of 0.21 meters and an IQR of 0.30 meters. These results underscore the high accuracy achieved by using fully calibrated cameras.

- **Proposed Method (Ours):** Our Proposed Method shows competitive performance compared to Calibrated Cameras, particularly evident in Camera 3. It exhibits a mean rotation error of 6.57°, a median of 3.47°, and an IQR of 4.20°. The mean translation error is 0.46 meters, with a median of 0.22 meters and an IQR of 0.29 meters. This highlights the effectiveness of our approach even without full calibration.

- **Standard Triangulation (ST):** Using standard triangulation methods results in higher errors compared to constrained methods. For example, with Camera 4, it shows a mean rotation error of 6.66°, a median of 3.34°, and an IQR of 4.16°. The mean translation error is 0.50 meters, with a median of 0.25 meters and an IQR of 0.26 meters. This emphasizes the importance of using constraints to guide the triangulation process.

- **No Fusion (NF):** Omitting the heatmap fusion step leads to increased rotation and translation errors. For Camera 4, it results in a mean rotation error of 6.58°, a median of 3.30°, and an IQR of 4.12°. The mean translation error is 0.50 meters, with a median of 0.25 meters and an IQR of 0.26 meters. This underscores the critical role of fusing heatmaps to improve keypoint accuracy across multiple views.

- **Without Bundle Adjustment (WBA):** Excluding bundle adjustment also results in considerable increases in errors. For Camera 4, it shows a mean rotation error of 6.62°, a median of 3.32°, and an IQR of 4.14°. The mean translation error is 0.50 meters, with a median of 0.25 meters and an IQR of 0.26 meters. This demonstrates that bundle adjustment has a crucial role in refining poses and minimizing errors.

The translation and rotation error distributions from the ablation study provide insights into the performance of different methods for camera-to-robot pose estimation.

Calibrated Cameras consistently demonstrate the lowest median errors, with rotation errors averaging approximately 6.04° and translation errors around 0.429 meters. This highlights the effectiveness of using fully calibrated cameras to achieve high accuracy.

Our Proposed Method, despite not relying on full calibration, exhibits competitive performance. For instance, in the case of Camera 3, rotation errors average 6.57° and translation errors around 0.458 meters, showcasing the robustness of our approach.

Standard Triangulation methods result in noticeable increases in both rotation and translation errors. For example, using this method for Camera 3, rotation errors average 6.76° and translation errors around 0.472 meters, emphasizing the importance of using constrained methods to guide triangulation.

Omitting the heatmap fusion step leads to significant deteriorations in accuracy. For Camera 2, rotation errors average 6.558° and translation errors around 0.455 meters, underscoring the critical role of heatmap fusion in improving keypoint accuracy across multiple views.

Excluding bundle adjustment also results in considerable increases in errors. For Camera 1, rotation errors average 6.54° and translation errors around 0.457 meters, demonstrating the essential role of bundle adjustment in refining poses and reducing errors.

Removing these steps results in a marked increase in both rotation and translation errors, emphasizing the necessity of all these techniques for achieving high precision and reliability in camera-to-robot pose estimation.

5.5.2 Camera-to-Camera Pose Analysis

We perform an ablation study also on the estimation of poses between relative cameras. In this case, the only step of the model that actually works to refine the relative camera-to-camera poses is the final phase of the bundle adjustment step. The results are plotted in Figure 5.21 and Figure 5.20 for the rotation and translation error respectively. The study indicates the following insights:

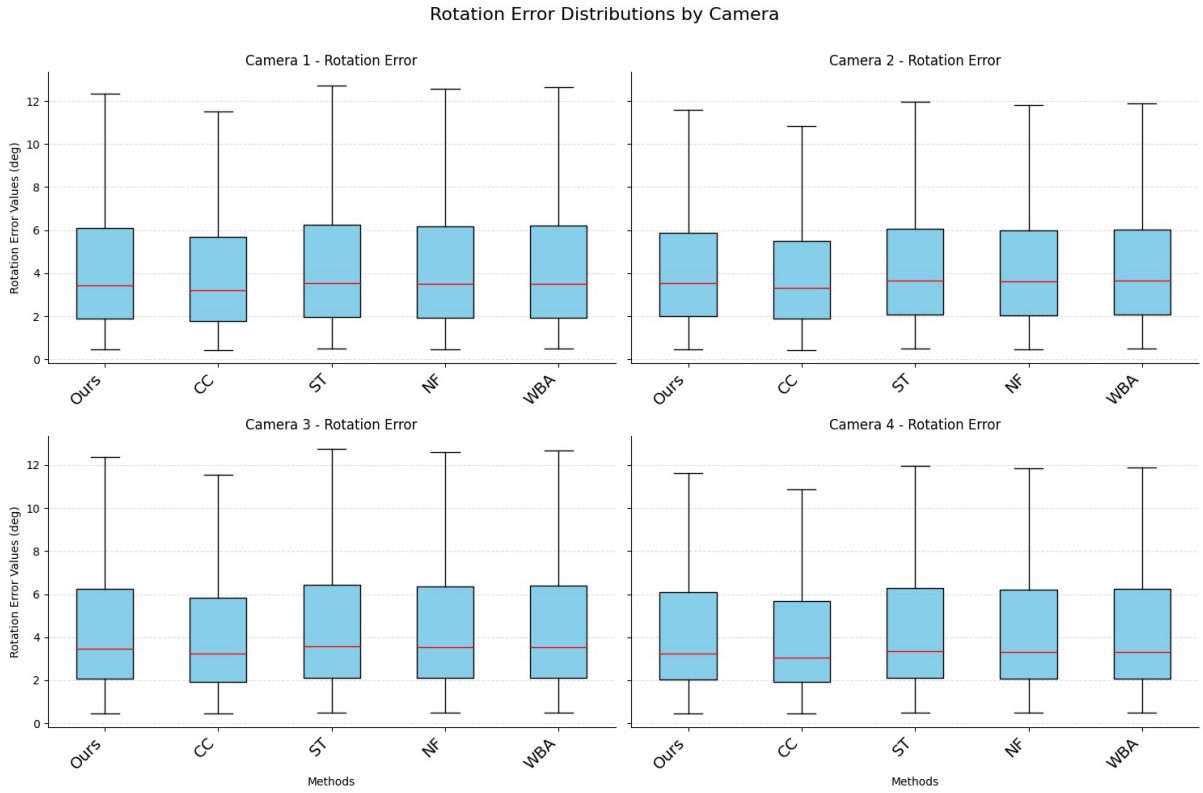

Figure 5.18: Ablation study results demonstrating the impact of the model's steps on the rotation error in camera-to-robot pose estimation.

- **With Bundle Adjustment (Ours):** Including the bundle adjustment step consistently reduces both rotation and translation errors. For instance, with Camera 1, rotation errors average 2.98°, with a median of 1.60° and an IQR of 1.95°. Translation errors average 0.397 meters, with a median of 0.218 meters and an IQR of 0.244 meters.

- **Without Bundle Adjustment (WBA):** Excluding the bundle adjustment step leads to higher rotation and translation errors across all cameras. For example, with Camera 1, rotation errors average 3.07°, with a median of 1.65° and an IQR of 2.01°. Translation errors average 0.413 meters, with a median of 0.227 meters and an IQR of 0.253 meters.

Detailed examination of rotation and translation error distributions provides further insights. Cameras consistently exhibit lower median rotation errors and narrower IQRs when bundle adjustment is included. For instance, in Camera 2, rotation errors average 3.01°, with a median of 1.66° and an IQR of 1.80°. Translation errors average 0.396 meters, with a median of 0.20 meters and an IQR of 0.283 meters.

Without bundle adjustment, cameras show higher variability and increased errors. For example, in Camera 2, rotation errors average 3.10°, with a median of 1.71° and an IQR of 1.85°. Translation errors average 0.413 meters, with a median of 0.20 meters

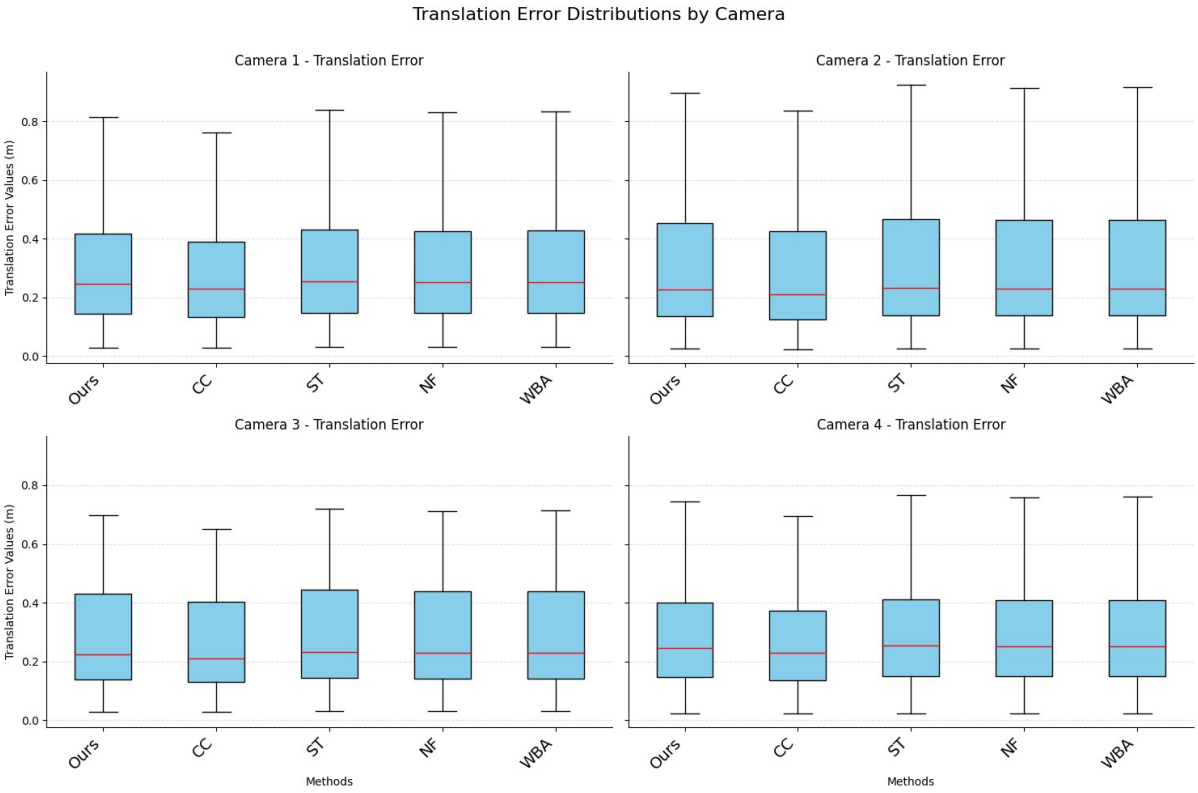

Figure 5.19: Ablation study results demonstrating the impact of the model's steps on the translation error in camera-to-robot pose estimation.

and an IQR of 0.294 meters.

These observations highlight the necessity of bundle adjustment in achieving precise and consistent camera-to-camera pose estimates, essential for robust performance in multi-camera systems.

The method that incorporates bundle adjustment consistently outperforms the one that does not, showcasing lower median errors and diminished variability. This underscores the critical importance of the refinement step in optimizing the performance of our pose estimation model.

This means that integrating the estimated camera-to-robot poses within the bundle adjustment process contributes significantly to refining relative poses. This integration not only enhances robustness against noise and errors but also strengthens the overall stability and reliability of the pose estimation framework. By treating the estimated cameras-to-robot poses as part of the refinement phase, we capitalize on their role in achieving more precise and consistent results.

Table 5.11: Rotation error results of the camera-to-robot pose estimation for the ablation study, expressed in degrees. 'Ours' refers to the proposed method with all the steps 'CC' refers to the method using calibrated cameras, 'ST' uses normal triangulation instead of the constrained triangulation, 'NF' refers to the model without the heatmap fusion step and 'WBA' is the model without the last two-stage bundle adjustment.

Camera	Method	Mean	Median	IQR	Maximum	Minimum
Camera 1	Ours	6.39162	3.42818	4.18635	12.3351	0.4695
	CC	**5.97347**	**3.20391**	**3.91248**	**11.5281**	**0.438785**
	ST	6.58336	3.53103	4.31195	12.7052	0.483585
	NF	6.50666	3.48989	4.26171	12.5571	0.477951
	WBA	6.54501	3.51046	4.28683	12.6311	0.480768
Camera 2	Ours	6.44195	3.55508	3.85672	11.6077	0.4695
	CC	**6.02052**	**3.3225**	**3.60441**	**10.8483**	**0.438785**
	ST	6.63521	3.66173	3.97242	11.9559	0.483585
	NF	6.55791	3.61907	3.92614	11.8166	0.477951
	WBA	6.59656	3.6404	3.94928	11.8862	0.480768
Camera 3	Ours	6.57177	3.47016	4.19942	12.3706	0.4695
	CC	**6.14184**	**3.24314**	**3.92469**	**11.5613**	**0.438785**
	ST	6.76892	3.57427	4.3254	12.7418	0.483585
	NF	6.69006	3.53262	4.27501	12.5933	0.477951
	WBA	6.72949	3.55344	4.3002	12.6675	0.480768
Camera 4	Ours	6.46327	3.24636	4.04283	11.6237	0.469504
	CC	**6.04044**	**3.03398**	**3.77835**	**10.8633**	**0.438789**
	ST	6.65717	3.34375	4.16412	11.9724	0.483589
	NF	6.57961	3.30479	4.1156	11.8329	0.477955
	WBA	6.61839	3.32427	4.13986	11.9027	0.480772

Table 5.12: Translation error results of the camera-to-robot pose estimation for the ablation study, expressed in meters. 'Ours' refers to the proposed method with all the steps, 'CC' refers to the method using calibrated cameras, 'ST' uses normal triangulation instead of the constrained triangulation, 'NF' refers to the model without the heatmap fusion step and 'WBA' is the model without the last two-stage bundle adjustment.

Camera	Method	Mean	Median	IQR	Maximum	Minimum
Camera 1	Ours	0.447935	0.246548	0.27498	0.814737	0.0296835
	CC	**0.418631**	**0.230419**	**0.25699**	**0.761437**	**0.0277416**
	ST	0.461821	0.254191	0.283504	0.839994	0.0306037
	NF	0.456893	0.251479	0.280479	0.831032	0.0302772
	WBA	0.457789	0.251972	0.281029	0.832662	0.0303365
Camera 2	Ours	0.447053	0.225923	0.318978	0.895663	0.024438
	CC	**0.417807**	**0.211143**	**0.298111**	**0.837068**	**0.0228392**
	ST	0.460912	0.232927	0.328867	0.923428	0.0251955
	NF	0.455994	0.230442	0.325358	0.913576	0.0249267
	WBA	0.456888	0.230894	0.325996	0.915367	0.0249756
Camera 3	Ours	0.458141	0.224357	0.291363	0.698099	0.0290697
	CC	**0.428169**	**0.209679**	**0.272302**	**0.652429**	**0.027168**
	ST	0.472344	0.231312	0.300395	0.719741	0.0299709
	NF	0.467304	0.228844	0.29719	0.712061	0.0296511
	WBA	0.46822	0.229293	0.297773	0.713458	0.0297093
Camera 4	Ours	0.48591	0.245465	0.25358	0.744495	0.0228487
	CC	**0.454122**	**0.229406**	**0.23699**	**0.69579**	**0.021354**
	ST	0.500974	0.253074	0.261441	0.767574	0.023557
	NF	0.495629	0.250374	0.258651	0.759385	0.0233057
	WBA	0.4966	0.250865	0.259158	0.760874	0.0233514

Table 5.13: Rotation error results of the camera-to-camera pose estimation for the ablation study, expressed in meters. 'Ours' refers to the proposed method with all the steps and 'WBA' is the model without the last two-stage bundle adjustment.

Camera	Method	Mean	Median	IQR	Maximum	Minimum
Camera 1	Ours	**2.98674**	**1.60195**	**1.95624**	**5.76407**	**0.219393**
	WBA	3.07634	1.65001	2.01493	5.93699	0.225974
Camera 2	Ours	**3.01026**	**1.66125**	**1.8022**	**5.42414**	**0.219393**
	WBA	3.10057	1.71109	1.85627	5.58686	0.225974
Camera 3	Ours	**3.07092**	**1.62157**	**1.96234**	**5.78067**	**0.219393**
	WBA	3.16305	1.67022	2.02121	5.95409	0.225974
Camera 4	Ours	**3.02022**	**1.51699**	**1.88917**	**5.43164**	**0.219394**
	WBA	3.11083	1.5625	1.94585	5.59458	0.225976

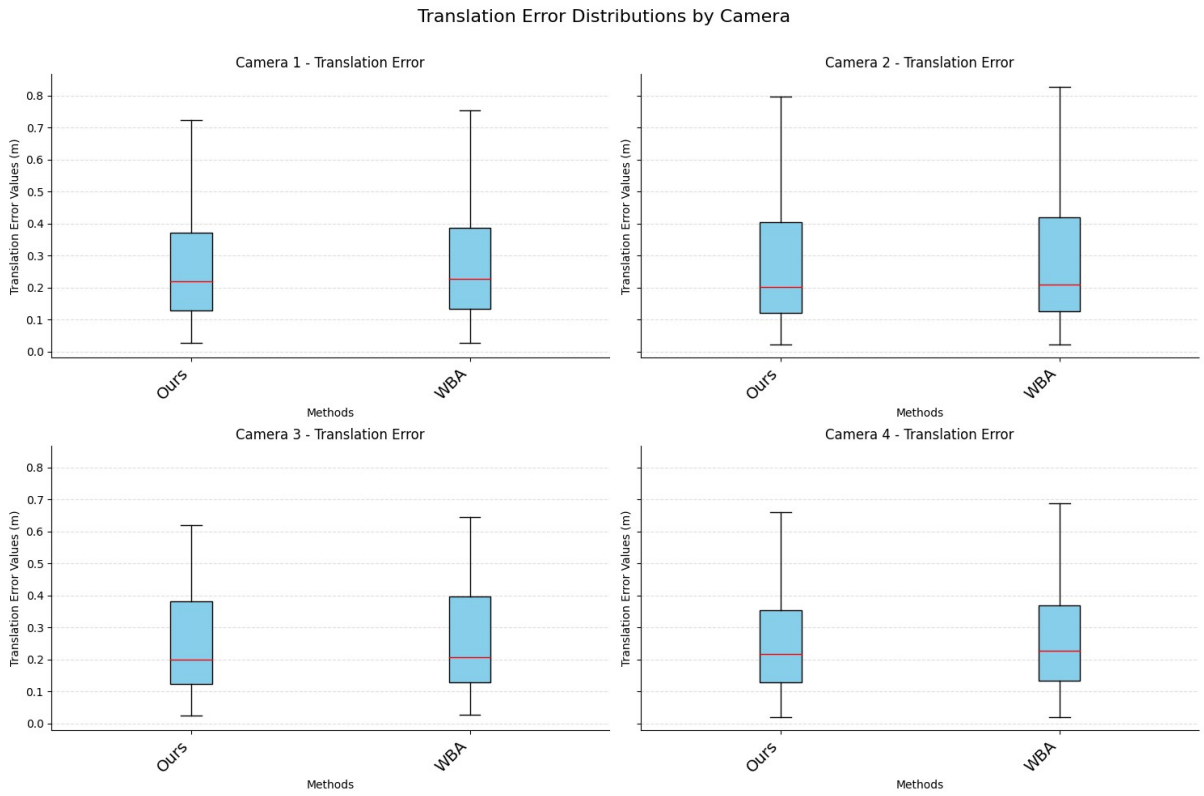

Figure 5.20: Ablation study results demonstrating the impact of the model's steps on the translation error in camera-to-camera pose estimation.

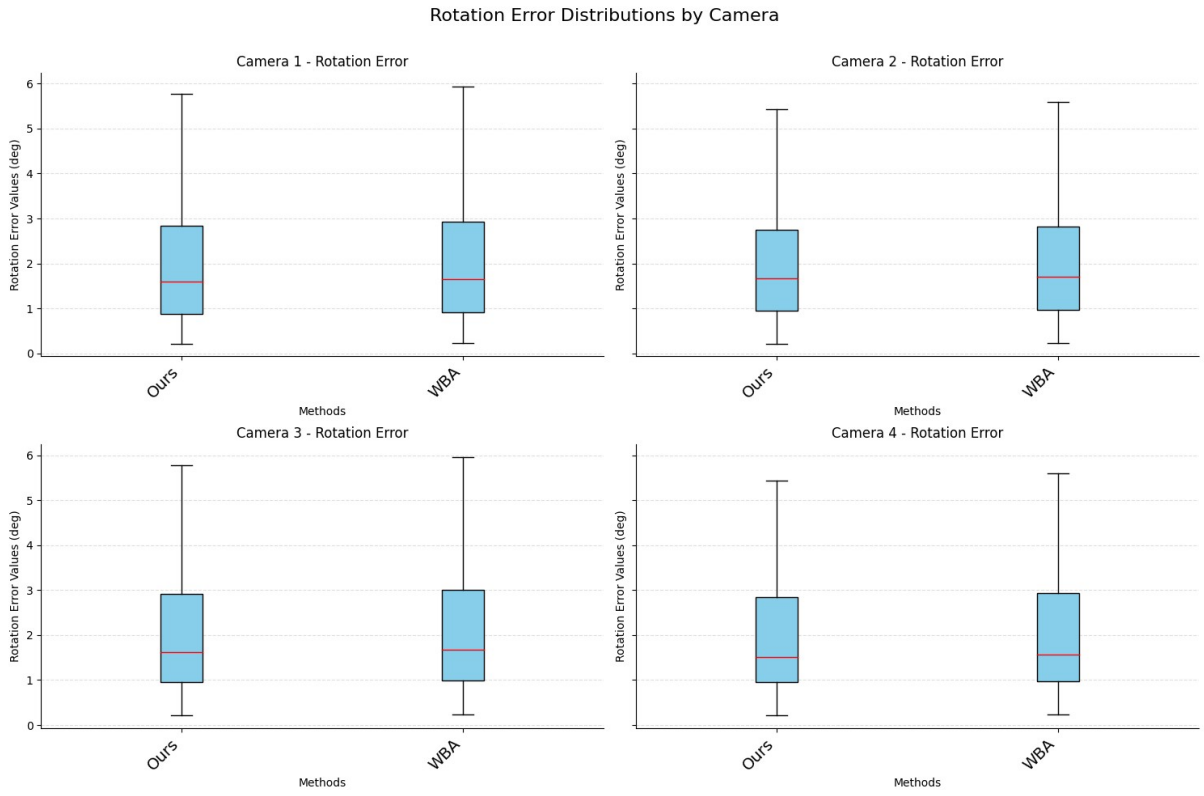

Figure 5.21: Ablation study results demonstrating the impact of the model's steps on the rotation error in camera-to-camera pose estimation.

Table 5.14: Translation error results of the camera-to-camera pose estimation for the ablation study, expressed in meters. 'Ours' refers to the proposed method with all the steps and 'WBA' is the model without the last two-stage bundle adjustment.

Camera	Method	Mean	Median	IQR	Maximum	Minimum
Camera 1	Ours	**0.397699**	**0.218898**	**0.244141**	**0.723365**	**0.0263545**
	WBA	0.413607	0.227654	0.253907	0.7523	0.0274087
Camera 2	Ours	**0.396916**	**0.200586**	**0.283205**	**0.795215**	**0.0216972**
	WBA	0.412793	0.20861	0.294533	0.827023	0.0225651
Camera 3	Ours	**0.406761**	**0.199195**	**0.258687**	**0.619808**	**0.0258096**
	WBA	0.423031	0.207163	0.269034	0.6446	0.026842
Camera 4	Ours	**0.431416**	**0.217936**	**0.225141**	**0.661**	**0.0202863**
	WBA	0.448672	0.226653	0.234146	0.68744	0.0210977

6

Conclusions and Future Works

In this thesis, we address the challenge of calibrating a network of cameras using only a single frame per camera, leveraging a robot as the calibration pattern and simultaneously estimating the robot's pose relative to the cameras. This problem has gained recent interest, particularly with the advancements in deep learning. We propose a novel method that utilizes multi-view geometry to enhance accuracy and reduce errors.

Our approach begins by estimating inter-camera poses in a Structure from Motion (SfM) manner, exploiting the feature matching obtained with Lightglue [35]. Subsequently, we employ C2P [61] to compute rotation matrices and translation vectors, refined using 2D joint detections from the DREAM model [31]. In the second step we generate heatmaps and keypoints corresponding to joints, and successively we fuse these heatmaps using the process detailed in Section 4.4. Triangulation of points is then performed with the addition of constraints based on the kinematic structure of the robot. Once triangulated points are obtained, we estimate poses from the robot to the cameras, followed by a two-step bundle adjustment process to refine the poses.

We evaluate our model against the single-camera scenario and various configurations and numbers of cameras. As demonstrated in Chapter 5, our approach significantly outperforms the single-camera model, improving rotation estimation by up to 38% and translation estimation by up to 40% in some cases. It proves more robust and resilient to noise.

We observe that while the number of cameras influences camera-to-robot poses, the impact is less significant than anticipated. The improvement from 3 to 5 cameras results in only about a 4% reduction in rotation error and a 5% reduction in translation error, with errors slightly decreasing as more cameras are added.

The distance between cameras and the robot also affects results; positioning cameras farther away tends to degrade performance, potentially due to limitations in the single-

camera model's training for handling such cases. Conversely, greater camera distance improves relative camera pose estimation due to expanded field of view, thereby providing more matches for the C2P solver.
Additionally, we conduct an ablation study to assess the influence and importance of each step. We find that pre-calibrating cameras yields the least error, around 6.04° for rotation error compared to about 6.45° for the uncalibrated setup, and 0.43 m compared to 0.455 m for the translation error, respectively. This result aligns with our expectations, as pre-calibration allows for more efficient leveraging of epipolar geometry during model fusion.
Constraints applied to triangulation and the specific methods employed are crucial, with constrained triangulation and calibrated cameras showing the highest impact, followed by heatmap fusion and bundle adjustment. Our study also highlights the effectiveness of bundle adjustment in reducing errors, from 3.11° without to 3.03° with bundle adjustment for the rotation error, and from 0.427 m without to 0.405 m with bundle adjustment for the translation error in relative camera pose estimation.

6.1 Future Works

In future research, several promising directions could further advance the methods proposed in this thesis. Firstly, exploring advanced deep learning techniques could enhance feature detection and matching processes [40], potentially reducing computational overhead while improving accuracy in pose estimation tasks. Investigating methods to dynamically adapt to varying camera configurations and distances from the robot could mitigate the current limitations observed, particularly in scenarios where cameras are positioned at greater distances. Additionally, extending the model to support a broader range of robot types is essential.
Currently, the DREAM model only supports a limited set of robots, such as the Panda Emika Franka. Future work could involve developing adaptable algorithms that can handle different robotic platforms, ensuring flexibility and applicability across diverse industrial or research environments.
Furthermore, enhancing the robustness and scalability of the system to accommodate multiple robots within its field of view presents another area of improvement. This capability would enable the system to track and estimate poses for multiple robots simultaneously, facilitating collaborative tasks or scenarios where multiple robots are deployed. Integrating multi-robot coordination and communication protocols could also be explored to enhance efficiency and coordination among robots within the network.
Moreover, investigating real-time adaptation and learning mechanisms could further enhance the system's performance in dynamic environments. Techniques such as on-

line learning or adaptive control strategies could enable the system to continuously improve and adapt its pose estimation capabilities based on real-world feedback and environmental changes.

References

[1] Simoni Alessandro et al. “Semi-Perspective Decoupled Heatmaps for 3D Robot Pose Estimation from Depth Maps”. In: *IEEE Robotics and Automation Letters* (2022).

[2] Sowmya Munukutla Axel Barroso-Laguna and Eric Brachmann Victor Adrian Prisacariu. “Matching 2D Images in 3D: Metric Relative Pose from Metric Correspondences”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 4852–4863.

[3] Sven Behnke Bastian Pätzold Simon Bultmann. “Online Marker-free Extrinsic Camera Calibration using Person Keypoint Detections”. In: *DAGM German Conference on Pattern Recognition (GCPR)*. 2022.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Speeded-Up Robust Features (SURF)”. In: *Computer Vision and Image Understanding* 110.3 (2008), pp. 346–359.

[5] Vasileios Belagiannis et al. “3D Pictorial Structures for Multiple Human Pose Estimation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1669–1676. DOI: 10.1109/CVPR.2014.216.

[6] Shihong Xia Boyuan Jiang Lei Hu. “Probabilistic Triangulation for Uncalibrated Multi-View 3D Human Pose Estimation”. In: *IEEE International Conference on Computer Vision* (2023).

[7] Michael Calonder et al. “BRIEF: Binary Robust Independent Elementary Features”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2010, pp. 778–792.

[8] Zhuo Chen, Xu Zhao, and Xiaoyue Wan. “Structural Triangulation: A Closed-Form Solution to Constrained 3D Human Pose Estimation”. In: *European Conference on Computer Vision (ECCV)*. 2022, pp. 695–711.

[9] Zhuo Chen, Xu Zhao, and Xiaoyue Wan. “Supplementary Material for Structural Triangulation: A Closed-Form Solution to Constrained 3D Human Pose Estimation”. In: *European Conference on Computer Vision (ECCV)*. 2022.

[10] Hainan Cui et al. "Efficient and robust large-scale structure-from-motion via track selection and camera prioritization". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 156 (2019), pp. 202–214. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2019.08.005. URL: https://www.sciencedirect.com/science/article/pii/S0924271619301893.

[11] Shen S. Cui H Gao X. *MCSfM: Multi-Camera-Based Incremental Structure-From-Motion*. 2023. DOI: 10.1109/TIP.2023.3333547.

[12] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "SuperPoint: Self-Supervised Interest Point Detection and Description". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 224–236.

[13] Olof Enqvist, Fredrik Kahl, and Carl Olsson. "Non-sequential structure from motion". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011, pp. 264–271. DOI: 10.1109/ICCVW.2011.6130252.

[14] Valassakis Eugene, Dreczkowski Kamil, and Johns Edward. "Learning Eye-in-Hand Camera Calibration from a Single Image". In: *Conference on Robot Learning (CoRL)*. 2021.

[15] Daniele Evangelista et al. "A Graph-Based Optimization Framework for Hand-Eye Calibration for Multi-Camera Setups". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2023. DOI: 10.1109/icra48891.2023.10160758. URL: http://dx.doi.org/10.1109/ICRA48891.2023.10160758.

[16] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[17] Qiu Haibo et al. "Cross View Fusion for 3D Human Pose Estimation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 4342–4351.

[18] Tu Hanyue, Wang Chunyu, and Zeng Wenjun. "VoxelPose: Towards Multi-Camera 3D Human Pose Estimation in Wild Environment". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

[19] T. Heikkila et al. "Flexible hand-eye calibration for multi-camera systems". In: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*. Vol. 3. 2000, 2292–2297 vol.3. DOI: 10.1109/IROS.2000.895310.

[20] Karim Iskakov et al. "Learnable Triangulation of Human Pose". In: *International Conference on Computer Vision (ICCV)*. 2019.

[21] Bohg Jeannette et al. “Robot Arm Pose Estimation through Pixel-Wise Part Classification”. In: *IEEE International Conference on Robotics and Automation (ICRA)* (2014).

[22] Lambrecht Jens, Grosenick Philipp, and Meusel Marvin. “Optimizing Keypoint-based Single-Shot Camera-to-Robot Pose Estimation through Shape Segmentation”. In: *IEEE International Conference on Robotics and Automation (ICRA)* (2021).

[23] Sun Jiaming et al. “LoFTR: Detector-Free Local Feature Matching With Transformers”. In: 2021, pp. 8922–8931.

[24] Yiran Zhong Jianyuan Wang et al. “Deep Two-View Structure-From-Motion Revisited”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 8953–8962.

[25] Lawrence Jim, Bernal Javier, and Witzgall Christoph. “A Purely Algebraic Justification of the Kabsch-Umeyama Algorithm”. In: *Journal of Research of the National Institute of Standards and Technology* (2019).

[26] Lu Jingpei, Richter Florian, and Yip Michael. “Pose Estimation for Robot Manipulators via Keypoint Optimization and Sim-to-Real Transfer”. In: *IEEE Robotics and Automation Letters*. 2022.

[27] Wolfgang Kabsch. “A solution for the best rotation to relate two sets of vectors”. In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (1976), pp. 922–923. DOI: 10.1107/S0567739476001873.

[28] Fadi Khatib et al. “GRelPose: Generalizable End-to-End Relative Camera Pose Regression”. In: *DeepAI* (2021).

[29] Laurent Kneip, Hongdong Li, and Yongduek Seo. “UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Springer International Publishing, 2014, pp. 127–142.

[30] Zuzana Kukelova, Martin Bujnak, and Tomas Pajdla. “Polynomial Eigenvalue Solutions to the 5-pt and 6-pt Relative Pose Problems”. In: Jan. 2008. DOI: 10.5244/C.22.56.

[31] Timothy E. Lee et al. “Camera-to-Robot Pose Estimation from a Single Image”. In: *ICRA 2020*. 2020.

[32] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. “EPnP: An accurate O(n) solution to the PnP problem”. In: *International Journal of Computer Vision*. Vol. 81. 2. Springer, 2009, pp. 155–166.

[33] Hongdong Li and R. Hartley. "Five-Point Motion Estimation Made Easy". In: *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 1. 2006, pp. 630–633. DOI: 10.1109/ICPR.2006.579.

[34] Yuheng Li et al. "Learning to Estimate 3D Rigid Body Transformations using 6D Pose Supervision". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[35] Philipp Lindenberger et al. "LightGlue: Local Feature Matching at Light Speed". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[36] Chen Lingling et al. "Multi-View Auto-Calibration Method Based on Human Pose Estimation". In: *4th International Conference on Robotics and Computer Vision (ICRCV)*. 2022.

[37] H. C. Longuet-Higgins. "A computer algorithm for reconstructing a scene from two projections". In: *Nature* 293.5828 (1981), pp. 133–135. ISSN: 1476-4687. DOI: 10.1038/293133a0. URL: https://doi.org/10.1038/293133a0.

[38] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[39] Jingpei Lu, Florian Richter, and Michael C. Yip. *Markerless Camera-to-Robot Pose Estimation via Self-supervised Sim-to-Real Transfer*. 2023. arXiv: 2302.14332.

[40] Wei-Chiu Ma et al. *Virtual Correspondence: Humans as a Cue for Extreme-View Geometry*. 2022. arXiv: 2206.08365 [cs.CV]. URL: https://arxiv.org/abs/2206.08365.

[41] Loper Matthew et al. "SMPL: A skinned multi-person linear model". In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*. 2015, 248:1–248:16.

[42] Gyeongsik Moon, Juyong Chang, and Kyoung Mu Lee. "Camera Distance-aware Top-down Approach for 3D Multi-person Pose Estimation from a Single RGB Image". In: *The IEEE Conference on International Conference on Computer Vision (ICCV)*. 2019.

[43] Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. "Accurate Non-Iterative O(n) Solution to the PnP Problem". In: *2007 IEEE 11th International Conference on Computer Vision*. 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4409116.

[44] E. Mouragnon et al. "Generic and real-time structure from motion using local bundle adjustment". In: *Image and Vision Computing* 27.8 (2009), pp. 1178–1193. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2008.11.006. URL: https://www.sciencedirect.com/science/article/pii/S0262885608002436.

[45] D. Nister. "An efficient solution to the five-point relative pose problem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.6 (2004), pp. 756–770. DOI: 10.1109/TPAMI.2004.17.

[46] Saini Nitin et al. "SmartMocap: Joint Estimation of Human and Camera Motion using Uncalibrated RGB Cameras". In: *IEEE Robotics and Automation Letters*. 2023.

[47] Sangxia Huang Olivier Moliner and Kalle Astrom. "Better Prior Knowledge Improves Human-Pose-Based Extrinsic Camera Calibration". In: *25th International Conference on Pattern Recognition (ICPR)*. 2020.

[48] Paul-Edouard Sarlin Philipp Lindenberger, Viktor Larsson, and Marc Pollefeys. "Pixel-Perfect Structure-From-Motion with Featuremetric Refinement". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. 2021, pp. 5987–5997.

[49] R. C. Prim. "Shortest connection networks and some generalizations". In: *Bell System Technical Journal* 36.6 (1957), pp. 1389–1401. DOI: 10.1002/j.1538-7305.1957.tb01515.x.

[50] Praveen Kumar Rajendran et al. "RelMobNet: End-to-end relative camera pose estimation using a robust two-stage training". In: *arXiv preprint arXiv:2202.12838* (2022).

[51] Chris Rockwell et al. "FAR: Flexible, Accurate and Robust 6DoF Relative Camera Pose Estimation". In: 2024.

[52] Edward Rosten and Tom Drummond. "Machine learning for high-speed corner detection". In: *European Conference on Computer Vision (ECCV)*. Springer, 2006, pp. 430–443.

[53] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2011, pp. 2564–2571.

[54] Paul-Edouard Sarlin et al. "SuperGlue: Learning Feature Matching with Graph Neural Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4938–4947.

[55] Bernard Schmidt and Lihui Wang. "Automatic work objects calibration via a globallocal camera system". In: *Robotics and Computer-Integrated Manufacturing* 30.6 (2014), pp. 678–683. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2013.11.004. URL: https://www.sciencedirect.com/science/article/pii/S0736584513001014.

[56] Tanner Schmidt et al. "DeepV2D: Video to Depth with Differentiable Structure from Motion". In: *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*. 2019, pp. 4463–4472.

[57] Johannes L. Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4104–4113.

[58] Y.C. Shiu and S. Ahmad. "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB". In: *IEEE Transactions on Robotics and Automation* 5.1 (1989), pp. 16–29. DOI: 10.1109/70.88014.

[59] Kosuke Takahashi et al. "Human Pose as Calibration Pattern: 3D Human Pose Estimation with Multiple Unsynchronized and Uncalibrated Cameras". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 1856–18567. DOI: 10.1109/CVPRW.2018.00230.

[60] Dengqing Tang et al. "AprilTag array-aided extrinsic calibration of cameralaser multi-sensor system". In: *Robotics and Biomimetics* 3.1 (2016), p. 13. DOI: 10.1186/s40638-016-0044-0. URL: https://doi.org/10.1186/s40638-016-0044-0.

[61] Javier Tirado-Garn and Javier Civera. "From Correspondences to Pose: Non-minimal Certifiably Optimal Relative Pose without Disambiguation". In: *CVPR* (2024).

[62] R.Y. Tsai and R.K. Lenz. "Real time versatile robotics hand/eye calibration using 3D machine vision". In: *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. 1988, 554–561 vol.1. DOI: 10.1109/ROBOT.1988.12110.

[63] Micha Tyszkiewicz, Pascal Fua, and Eduard Trulls. "DISK: Learning local features with policy gradient". In: *Advances in Neural Information Processing Systems* 33 (2020).

[64] Shinji Umeyama. "Least-squares estimation of transformation parameters between two point patterns". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (1991), pp. 376–380. DOI: 10.1109/34.88573.

[65] Ben Usman et al. *MetaPose: Fast 3D Pose from Multiple Views without 3D Supervision*. 2021. arXiv: 2108.04869.

[66] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[67] Tomas Izo W. and Eric L. Grimson. "Simultaneous Pose Estimation and Camera Calibration from Multiple Views". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR*. IEEE, 2004.

[68] Xiaoyue Wan, Zhuo Chen, and Xu Zhao. *View Consistency Aware Holistic Triangulation for 3D Human Pose Estimation*. 2023.

[69] John Wang and Edwin Olson. "AprilTag 2: Efficient and robust fiducial detection". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 4193–4198. DOI: 10.1109/IROS.2016.7759617.

[70] Xingkui Wei et al. *DeepSFM: Structure From Motion Via Deep Bundle Adjustment*. 2020. arXiv: 1912.09697.

[71] Size Wu et al. *Graph-Based 3D Multi-Person Pose Estimation Using Multi-View Images*. 2021. arXiv: 2109.05885.

[72] Yihong Wu and Zhanyi Hu. 2006. DOI: 10.1007/s10851-005-3617-z.

[73] Hailun Xia and Qiang Zhang. "VitPose: Multi-view 3D Human Pose Estimation with Vision Transformer". In: *2022 IEEE 8th International Conference on Computer and Communications (ICCC)*. 2022, pp. 1922–1927. DOI: 10.1109/ICCC56324.2022.10065997.

[74] Tian Yang et al. "RoboKeyGen: Robot Pose and Joint Angles Estimation via Diffusion-based 3D Keypoint Generation". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2024).

[75] Tian Yang et al. "Robot Structure Prior Guided Temporal Attention for Camera-to-Robot Pose Estimation from Image Sequence". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[76] Labbe Yann et al. "Single-view Robot Pose and Joint Angle Estimation via Render & Compare". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

[77] Zhixuan Yu et al. "Multiview Human Body Reconstruction from Uncalibrated Cameras". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 7879–7891. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/33610fba262d7b6fed0810b89f5Paper-Conference.pdf.

[78] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. DOI: 10.1109/34.888718.

[79] Zhe Zhang et al. "AdaFuse: Adaptive Multiview Fusion for Accurate Human Pose Estimation in the Wild". In: *IJCV* (2020), pp. 1–16.

[80] Xiaoming Zhao et al. "ALIKE: Accurate and Lightweight Keypoint Detection and Descriptor Extraction". In: *IEEE Transactions on Multimedia* (2022). DOI: 10.1109/TMM.2022.3155927. URL: http://arxiv.org/abs/2112.02906.

[81] Xiaoming Zhao et al. "ALIKED: A Lighter Keypoint and Descriptor Extraction Network via Deformable Transformation". In: *IEEE Transactions on Instrumentation & Measurement* 72 (2023), pp. 1–16. DOI: 10.1109/TIM.2023.3271000. URL: https://arxiv.org/pdf/2304.03608.pdf.

[82] Xiaopin Zhong et al. *G-SAM: A Robust One-Shot Keypoint Detection Framework for PnP Based Robot Pose Estimation*. 2023. DOI: 10.1007/s10846-023-01957-5.

Acknowledgments

I would like to express my deepest gratitude to my thesis advisor, Professor Stefano Ghidoni, and my co-advisor, Davide Allegro, for their invaluable guidance, assistance, and insightful feedback throughout this research endeavor. Their expertise and encouragement have been instrumental in shaping this thesis and my academic journey.
I extend my heartfelt appreciation to the Università degli Studi di Padova and the Intelligent Autonomous Systems Laboratory (IAS-lab) in particular for providing the necessary resources and conducive environment for conducting this research.
Special thanks go to my family for their unconditional love, patience, and understanding during this demanding period. Their encouragement and support have been a constant source of motivation.