Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische Friedrich-Wilhelms-Universität Bonn

Master's Thesis

Extrinsic Camera Calibration for Multiple Smart Edge Sensors using Person Keypoint Detections

Author:
Bastian Pätzold

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Dr. Volker Steinhage

Supervisor:
Simon Bultmann

Date: April 10, 2022

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.

Place, Date

Signature

Abstract

Camera calibration is an essential prerequisite for many methods in the field of computer vision and robotics. The extrinsic calibration of a multi-camera system is defined by the relative poses between the cameras. Obtaining them is typically achieved by the application of offline methods that utilize traditional checkerboard calibration targets. These methods can be perceived to be cumbersome and lengthy, considering that a new calibration is required each time any camera poses change. In this thesis, we propose an online method for the extrinsic calibration of multiple smart edge sensors, relying solely on 2D human keypoint detections that are extracted locally on the sensor boards, and omitting any further processing of the original RGB camera images. The person keypoint detections from multiple views are received at a central backend where they are synchronized, filtered, and assigned to person hypotheses. We use these person hypotheses to repeatedly construct optimization problems in the form of factor graphs. The knowledge obtained from their solutions accumulates over time and causes the estimated camera poses to converge toward their optimal pose. Our method assumes the intrinsic camera parameters to be known and requires priming with a rough initial estimate of the extrinsic calibration. Given a suitable sequence of one or multiple persons traversing the scene, the extrinsic calibration will converge within a few minutes. We show that the calibration with our method achieves lower reprojection errors compared to a reference calibration generated by an offline method utilizing a traditional calibration target.

Contents

1. Introduction

Sensor calibration is an essential task in any robotics application. Typically, robots utilize numerous sensors to perform perception, planning, and action tasks. In order to successfully interpret and utilize the measurements taken by a sensor, one must first obtain knowledge about its characteristics. An imprecise calibration can lead to a degradation in performance and possibly cause critical safety issues.

For a number of reasons, the topic of camera calibration is an inherently difficult one to solve [1]. First, the calibration parameters can change over time, due to vibration, thermal expansion, loose parts or other common problems that follow from normal usage. Therefore, it is not sufficient to calibrate the parameters only once during the construction of the system. Instead, calibration must be performed repeatedly throughout its lifetime. While, in practice, this problem might be considered negligible for the *intrinsic* parameters, e.g. the focal length of a camera, it is typically of great relevance for the *extrinsic* parameters, i.e. the rigid transformation between the camera's coordinate system and a reference coordinate system. Second, the calibration parameters cannot be measured directly with sufficient precision. Instead, they must be inferred from the data captured by the considered cameras. This introduces a variety of practical problems for the calibration procedure. Typically, this inference is performed by actively deploying, or utilizing some existent, calibration target of known correspondences in front of the cameras. Using a checkerboard pattern for this task is one of the most popular methods in computer vision [2]. However, the application of such a method requires expertise and might be perceived as cumbersome and lengthy, considering it has to be done repeatedly for a large multi-camera system. Additional challenges for inferring calibration parameters from image data involve accommodating for noisy measurements taken by the cameras or collecting a sufficient amount of data-points spread over the entirety of the image planes. In the case of finding the extrinsic parameters of a multi-camera system, synchronicity between the image streams must be established, data association has to be performed to find correspondences between images from multiple views, and the fields of view (FoV) of all cameras must overlap sufficiently.

In this thesis, we aim to develop a self-supervised method for calibrating the extrinsic parameters of an existing system of static smart edge sensors, in which each sensor runs inference for 2D human pose estimation. In particular, we want to infer the relative poses between the cameras on all deployed smart edge sensor boards in real-time, by using only the person keypoint detections being transmitted by the sensors.

The method is supposed to be capable of handling an unknown number of persons present in the scene simultaneously. Neither the dimensions or height of the persons, nor their precise orientation towards the cameras is assumed to be known. As mentioned above, the camera layout is assumed to have sufficiently overlapping FoVs to perform the task. We do not make any further assumptions on the scene that is observed by the cameras. For example, the method should support a non-planar ground surface, as well as arbitrary occlusions from tables or pillars. We assume the intrinsic parameters to be known for all considered cameras as well as a reasonable initial guess for the extrinsic calibration to be available. A detailed discussion on these two last requirements, their demanded accuracy and effect on the calibration procedure can be found in Sec. 6.3.

Our current solution for obtaining the extrinsic calibration of the sensor network is the *kalibr* toolkit [3]. While its precision is sufficient for our purposes, we perceive the calibration procedure to be cumbersome and lengthy. We hope to replace this solution with the developed method.

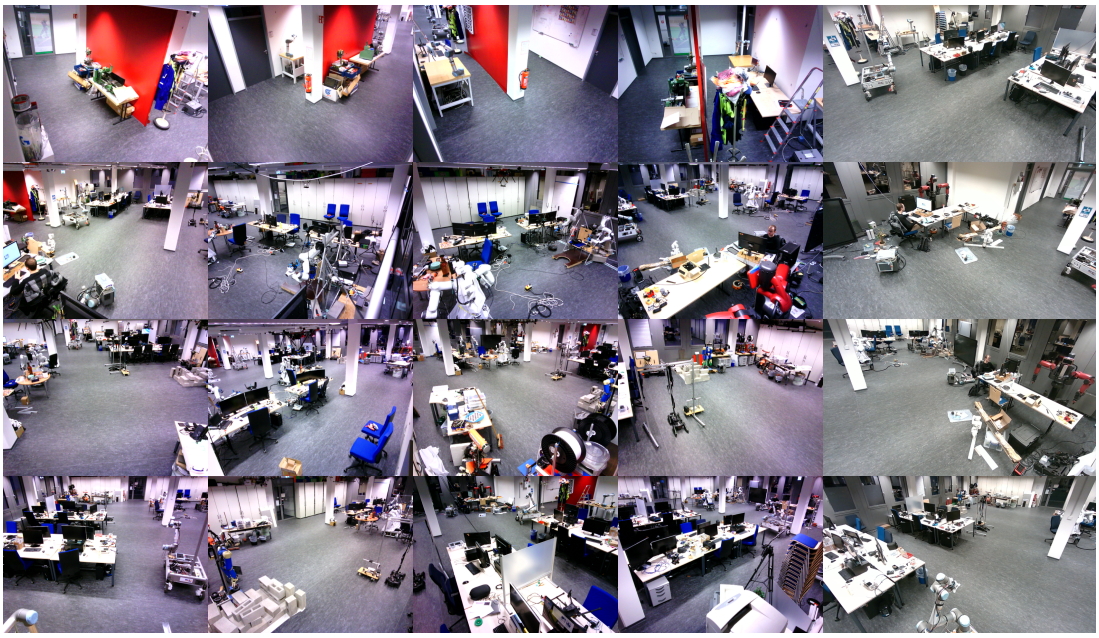

Figure 1.1: RGB images from the targeted system of 20 smart edge sensors showing the scene that serves as the scenario for this thesis.

2. Related Work

2.1. Camera Calibration

Traditional methods for camera calibration are based on using artificial image features, so called *fiducials*. The common idea behind this approach is to deploy some reference frame with known correspondences in front of all cameras that are to be calibrated. Zhang *et al.* [2] utilize a checkerboard pattern on a planar surface to perform intrinsic calibration of single cameras. The *kalibr* toolkit [3] uses a planar grid of *AprilTags* [4] to perform extrinsic calibration of multiple cameras, which allows to fully resolve its orientation towards the cameras, and is robust against occlusions. Another similarity of these traditional methods is that they require the recording of a calibration sequence in which the calibration target is positioned in front of the considered cameras and in their overlapping FoVs. The calibration sequence is then processed offline to obtain the results. In practice, this process can be tedious when considering larger multi-camera setups. In such a scenario, the calibration procedure must be repeated as soon as a single camera changes its pose or new cameras are added to the system. In our experience, offline processing alone can easily take days, depending on the number of cameras, the length of the calibration sequence and the used compute hardware.

To cope with the problems of these traditional approaches, methods for extrinsic camera calibration have been proposed that are not based on extracting fiducial features from classical calibration targets but use naturally occurring features instead. All approaches mentioned in the following assume the intrinsic parameters to be known. Komorowski *et al.* [5] extract SIFT-features [6] and find correspondences between multiple views using RANSAC [7]. They use segmentation to remove dynamic objects and validate their approach on stereo vision datasets. Their method is targeted towards one or few small baseline stereo cameras and offline processing of a small batch of images. Bhardwaj *et al.* [8] calibrate traffic cameras by deploying deep learning paradigms to extract vehicle instances and match them to a database of popular car models. The extracted features and known dimensions of the car models are then used to formulate a PnP-problem [7]. They assume a planar ground surface in the vicinity of the cars and process results offline. Guan *et al.* [9][10] detect two keypoints (Head & Feet) for each observable

pedestrian in a surveillance scenario. They perform pairwise triangulation of the pedestrians by assuming an average height for all visible persons in the image pair. Then they compute the calibration offline, using RANSAC, followed by a gradient descent based refinement scheme. Their resulting calibration is only defined up to an unknown scale factor, which must be resolved manually. The method assumes the center lines between all pedestrians to be parallel, or in other words, all persons are assumed to stand upright during the calibration, whereas other poses, e.g. sitting persons, are not supported.

At last, we want to mention Reinke *et al.* [11], who propose an offline method for finding the relative poses between a set of (two) cameras and the base frame of a quadruped robot. They use a fiducial marker mounted on the end-effector of a limb as the calibration target. The poses of the cameras mounted on the head of the robot are resolved by using a *factor graph* [12]. Kinematic constraints between the marker frame and the base frame are encoded as unary factors nodes.

2.2. Human Pose Estimation

Human pose estimation refers to the approach of recognizing and localizing anatomical keypoints of a person on some given image material. Early works use manually designed feature extractors like HOG-descriptors [13] or pictorial structures [14][15] for this task. In recent years, approaches using convolutional neural networks (CNNs) have become a popular choice and yield impressive results. Two well known state-of-the-art and publicly available methods for human pose estimation are *OpenPose* [16] and *AlphaPose* [17][18][19].

Figure 2.1: 2D human pose estimation for multiple persons simultaneously. Keypoint detections belonging to the same person are linked. [16]

Both methods allow the estimation of the poses of multiple persons simultaneously and in real-time. Another similarity is that they train and validate CNNs on some public datasets, which contain diverse video sequences of multiple persons carrying out complex tasks as well as groundtruth keypoint annotations for the validation set that is used for supervised learning. Afterward, the raw results generated by the CNNs are refined by various methods such as part affinity fields, transformers, or factor graphs. Numerous methods in the field use the *Common Objects in Context* (COCO) [20] and *MPII Human Pose* (MPII) [21] datasets. One of the key differences between them is their usage of different keypoint formats for the annotations contained in the validation set. In particular, the COCO dataset uses a total of 17 keypoints of which 5 are facial keypoints, while the MPII dataset uses a total of 16 keypoints of which only one is a facial keypoint.

So far we have only considered human pose estimation for single 2D image streams. Another interesting subject is the estimation of human poses in an allocentric 3D space. Here, the idea is to fuse 2D human poses from multiple views, i.e. using multiple cameras viewing a common scene from different perspectives. The available methods investigating this subject typically require a calibrated set of cameras. Another common problem that all 3D human pose estimation methods share is the requirement of performing data association. Here, one must find the correspondences between 2D detections from multiple views. Only this association allows to perform triangulation and infer depth information associated with the detections.

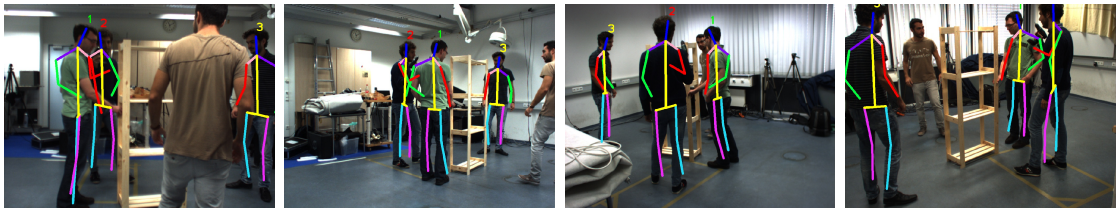

Figure 2.2: 3D human pose estimation on the Shelf dataset using four cameras and four persons. Only three persons are detected. Data association between views is indicated by numbers above the heads. [22]

Again, there exist a number of datasets that allow to train and validate such methods. Commonly used is the *Human3.6M* dataset (H3.6M) [23], as well as the *Campus* and *Shelf* datasets [22]. When training a 3D human pose estimation method based on an existing implementation for 2D human pose estimation, the dataset must use the same keypoint format as the dataset used for 2D estimation. Another important aspect concerning 2D and 3D human pose estimation is the distinction between online and offline methods. Online methods process

image streams and provide the corresponding results in real-time. Typically, these methods deploy smaller-sized networks or find other means of reducing their computational complexity, in order to become suitable for application in real-time. The reduction of computational complexity typically comes at the cost of reduced quality of the pose estimation. Some available methods offer to configure this trade-off between speed and accuracy [16].

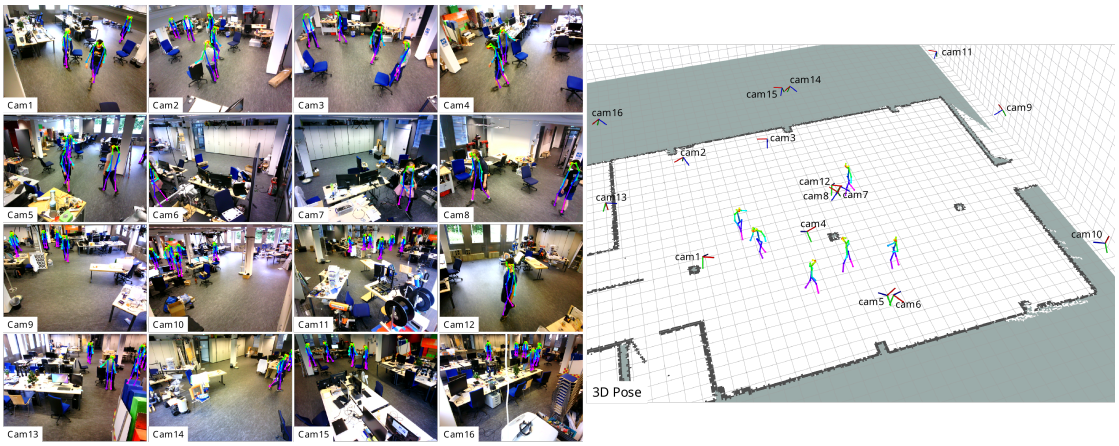

Figure 2.3: *Real-Time Multi-View 3D Human Pose Estimation using Semantic Feedback to Smart Edge Sensors* [24]. The image data is only shown for illustration purposes.

The goal of this thesis is to develop a method for extrinsic camera calibration on multiple smart edge sensors using person keypoint detections. We consider an existing network of smart edge sensors, deployed by Bultmann *et al.* [24] to perform 3D human pose estimation in real-time. Figure Fig. 2.3 illustrates the 2D and 3D pose estimation of this method, as well as the considered sensor network within this thesis. We will directly adopt this implementation for 2D pose estimation und use it as the foundation of our work.

In this method [24], each smart edge sensor is performing 2D human pose estimation in real-time using the COCO keypoint format. The advantages of processing the image data locally on the sensor boards and transmitting only the obtained keypoint data is to save network bandwidth and avoid privacy-related issues, as the original image data is not required to ever leave the sensor boards. Afterward, a central backend fuses the data received by the sensors to perform 3D human pose estimation in real-time, using a greedy matching algorithm for data association [25], triangulating corresponding detections between multiple views based on the direct linear transformation algorithm [26], refining the triangulation results by applying factor graph based skeleton models, and deploying a semantic feedback loop to improve the quality of the 2D human pose estimations.

3. Theory

In this chapter, we establish the theoretical background on which this thesis is based. We start by introducing the projective geometry required to formally describe the problem of camera calibration. Then we establish the concepts of factor graphs which we will use to solve this problem. Finally, we discuss and define various error metrics that allow us to assess the quality of its solution.

3.1. Projective Geometry

In this section, we establish the definitions for the camera model, the intrinsic and extrinsic calibration, as well as all the other aspects that concern the projective geometry utilized by the method proposed in this thesis. This section is largely based on the great book by Hartley and Zisserman [26].

3.1.1. Intrinsic Camera Calibration

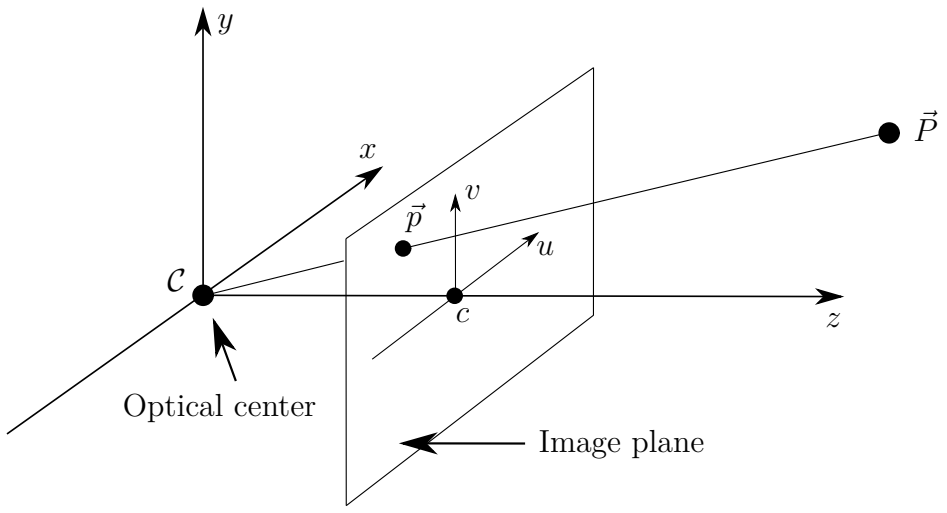

Figure 3.1: Illustration of the projective camera model. c is the principle point and $\mathcal{C}$ is the optical center. The local coordinate system of the camera is placed at the optical center. The image plane is depicted in front of the optical center.

The finite projective camera model that we utilize in this work is defined by its calibration matrix

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where the focal length $(f_x, f_y)^\mathsf{T}$ denotes the distance between the optical center $\mathcal{C}$ and the principle point c in pixels. Expressing the focal length in two components, one for the x- and one for the y-direction, accommodates for the possibility of having non-square pixels. The principal point denotes the origin of the *camera coordinate system* and is located where the line from the optical center intersects with the image plane perpendicularly. The parameters $(c_x, c_y)^\mathsf{T}$ are used to offset the origin of the *image coordinate system* from the principle point. We follow the convention of using the top left corner of the image plane as the origin of the image coordinate system. The *skew* parameter s is only non-zero in very rare cases and we assume it to be zero within the scope of this work.

The calibration matrix of a camera allows the projection of a point $\vec{P}$ in the three-dimensional space for which the cameras optical center $\mathcal{C}$ is the origin, we refer to it as the *local coordinate system*, to the corresponding two-dimensional point $\vec{p}$ in the image coordinate system, by

$$\vec{p} = \mathbf{K}\,[\mathbf{I}|0]\,\vec{P} \;\equiv\; \begin{pmatrix} f_x X + c_x Z \\ f_y Y + c_y Z \\ Z \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{3.2}$$

where $[\mathbf{I}|0]$ is a 3×4 matrix, consisting of the 3×3 identity matrix and the 1×3 zero vector in the last column, while $\vec{p}$ and $\vec{P}$ are expressed in homogeneous coordinates. Of course, the depth information gets lost in the projection. The projection mapping from local to image coordinates is described by

$$(X, Y, Z)^\mathsf{T} \mapsto (f_x X + c_x Z,\; f_y Y + c_y Z)^\mathsf{T}. \tag{3.3}$$

The above definitions hold for ideal cameras. In practice, however, camera lenses or entire cameras are imperfect in many ways. These imperfections lead to various non-linear distortion effects, that introduce an error for the projection from $\vec{P}$ to $\vec{p}$ w.r.t. Eq. 3.2. Tang *et al.* [27] describe a number of camera distortion models. In the scope of this work we will accommodate for two types of distortions: Radial distortions originate from imperfections in the manufacturing process of camera lenses and manifest in the form of *fish-eye* or *barrel* effects. Tangential distortions

occur when a camera lens is not positioned exactly planar to the image plane of the camera. These distortions can be reversed, or *undistorted*, by obtaining the appropriate distortion coefficients for the considered camera, and applying them to the measured image coordinates by using the inverse distortion models.

We refer to the calibration matrix $\mathbf{K}$, together with the (optional) distortion coefficients $\vec{\text{dist}}$, as the *intrinsic calibration* of a camera. Typically, the intrinsic calibration of a camera can be viewed as constant. In practice, this assumption might not always hold, e.g. when using an optical zoom lens, when the camera construction is not perfectly rigid, or due to changes in temperature. However, we disregard these effects in the scope of this work and do assume the intrinsic calibration to be constant. The intrinsic calibration can obtained for an individual camera, independent from other cameras in case it is embedded in a multi-camera system. Numerous methods for this purpose are available, most of which use a checkerboard type pattern with known sizes on a planar surface as the calibration target [2][3].

It is worth stressing, that we use a number of different coordinate systems, which must not be mixed up. In particular, we distinguish between the 2D *camera coordinate system*, originating from the priciple point c in the image plane, the 2D *image coordinate system*, for which the origin deviates from the principle point by $(c_x, c_y)^\mathsf{T}$, the 3D *local coordinate system*, for which the origin is the optical center $\mathcal{C}$ of the considered camera, and lastly, the 3D *global coordinate system*, in which the local coordinate systems of multiple cameras are embedded.

3.1.2. Extrinsic Camera Calibration

The *extrinsic calibration* of a camera typically refers to its pose, consisting of a position and an orientation component, toward some reference frame, e.g. an object in its FoV, another camera, or some world coordinate frame.
Within the scope of this work, we define the extrinsic calibration of a multi-camera system $\mathcal{S}_N$, with $N > 1$ finite projective cameras $\mathcal{C}_i$ for $i \in [0\,..\,N-1]$, where the pose for the optical center of each camera

$$\mathcal{C}_i = \{\vec{T}_i = (X_i, Y_i, Z_i)^\mathsf{T}, \vec{R}_i = (\alpha_i, \beta_i, \gamma_i)^\mathsf{T}\}\,, \tag{3.4}$$

consists of the position vector $\vec{T}_i$ and the orientation vector $\vec{R}_i$, embedded in one global three-dimensional Euclidean vector space, as knowing the relative poses $\Delta\mathcal{C}_{ij}$, i.e. knowing the translation vectors $\Delta\vec{T}_{ij}$ and the rotation vectors $\Delta\vec{R}_{ij}$, between all camera pairs ij for $i, j \in [0\,..\,N-1]$.

To spare the need for relating all camera poses to some common external refer-

ence frame, we define $\mathcal{C}_0$ to be the origin of our global coordinate system by

$$\mathcal{C}_0 \coloneqq \{\vec{T}_0 = (0\text{m}, 0\text{m}, 0\text{m})^\mathsf{T}, \vec{R}_0 = (0°, 0°, 0°)^\mathsf{T}\}\,. \tag{3.5}$$

In other words, the local coordinate system of $\mathcal{C}_0$ is the global coordinate system.

Note, that the knowledge of some relative camera poses $\Delta\mathcal{C}_{ij}$ can imply the knowledge of some other relative camera poses in $\mathcal{S}_N$. For example, for $N = 3$, the knowledge of $\Delta\mathcal{C}_{01}$ and $\Delta\mathcal{C}_{02}$ implies $\Delta\mathcal{C}_{12}$, such that knowing the relative poses of any two pairs of cameras can be considered as the full extrinsic calibration of $\mathcal{S}_3$.

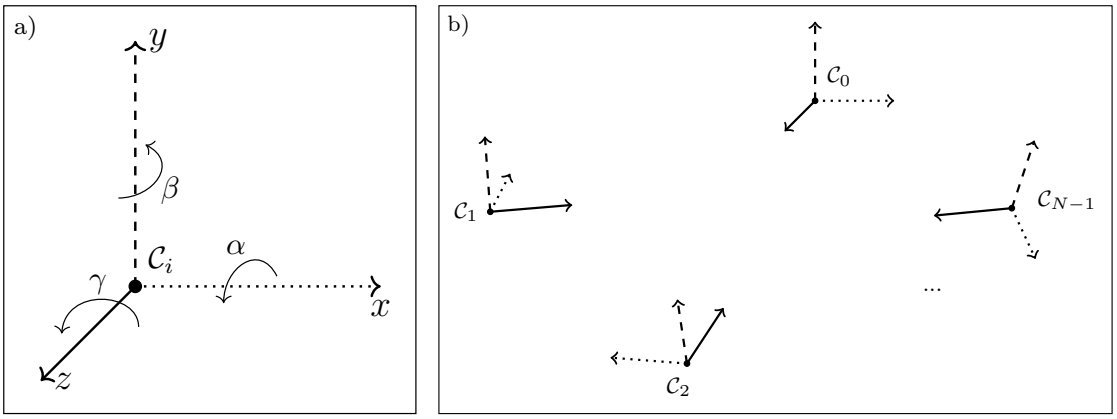

Figure 3.2: Convention for positive rotations of a local coordinate system (a) and an illustration of the extrinsic calibration where each camera is represented by its local coordinate system embedded in the global vector space (b).

The adopted conventions for rotations and axis labels are illustrated in Fig. 3.2. In accordance with Fig. 3.1, each camera looks straight down the z-axis of its local coordinate system. We refer to rotations around the three Euler angles α, β and γ as pitch, yaw and roll.

Knowing the extrinsic parameters $\vec{T}_i$ and $\vec{R}_i$ of a camera i allows us to adapt Eq. 3.2 for describing the mapping from a point $\vec{p}$ in the local coordinate system of $\mathcal{C}_i$ to the same point $\vec{P}$ in the global coordinate system, defined by

$$\vec{P} = \mathbf{K}_i\,[\,\mathbf{R}'_i\,|\,-\vec{T}_i\,]\,\vec{p} = \mathbf{P}_i\,\vec{p}\ , \tag{3.6}$$

where $\mathbf{R}'_i$ is the 3×3 rotation matrix for $\vec{R}_i$ and $\mathbf{P}_i = \mathbf{K}\,[\mathbf{R}'_i|\,-\vec{T}_i]$ is commonly referred to as the 3×4 *projection matrix* of $\mathcal{C}_i$.

3.2. Factor Graphs

This section gives a brief introduction into the concept of factor graphs and elaborates on certain aspects that are relevant for this work. For a more thorough introduction on factor graphs, please refer to the works of Koller *et al.* [28] and Dellaert *et al.* [12].

Let g be a joint probability distribution, ranging over multiple random variables, which can be factorized into a product of factors by

$$g(X_0, X_1, ..., X_{N-1}) = \prod_i f_i(S_i) \tag{3.7}$$

where each factor f_i only depends on a (small) subset of the random variables

$$S_i \subseteq \{X_0, X_1, ..., X_{N-1}\}\,. \tag{3.8}$$

A *factor graph* is a graphical model that is capable of representing such a factorization while allowing to run various kinds of message passing algorithms on it in an efficient way. Furthermore, one can show that a factor graph can represent any Bayesian network or Markov random field [29]. Figure 3.3 illustrates a factor graph, which represents the factorization of an exemplary joint probability distribution

$$g(X_0, X_1, X_2, X_3) = f_0(X_0) \cdot f_1(X_0, X_1) \cdot f_2(X_1, X_2, X_3)\,. \tag{3.9}$$

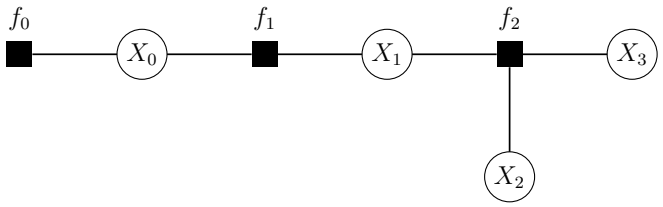

Figure 3.3: Illustration of an exemplary factor graph consisting of four variable nodes $(X_0, ..., X_3)$ and one unary (f_0), binary (f_1) and trinary (f_2) factor node each.

Formally, a factor graph $\mathcal{G}$ is an undirected graph with two types of nodes. Variable nodes represent the random variables X_n and factor nodes represent the factors f_i of g. A factor node representing the factor f_i is connected by an edge to all the variable nodes, that represent the random variables S_i on which the factor f_i depends. Therefore, a factor graph is a bipartite graph, where all edges connect

a variable node to a factor node.

Typically, we distinguish between unary and binary factor nodes, which either connect to one or two variable nodes. While representing factors depending on more random variables is possible, the strength of factor graphs lies in describing the independence relations between the random variables of complex joint probability distributions and in its convenience of allowing the addition of many simple factors to it, based on heuristic knowledge about the underlying problem. In particular, for an unknown joint probability distribution and a given set of measurements, which contains knowledge about the set of random variables it depends on, we can exploit heuristic knowledge about the underlying problem to obtain factors that constrain all (or a subset) of the random variables. Once a sufficient number of suitable constraints are added to the factor graph, we can run some optimization algorithm on it to find the maximum a-posteriori assignment for all constrained random variables. If not sufficiently many constraints were added to the graph or the factor graph is numerically poorly constrained, the optimization will fail, because a unique maximum a-posteriori does not exist [12].

If some of the contained factors are non-linear in nature, the graph must be linearized by minimizing the non-linear squared error specified by the factors before optimization can be performed [30]. A typical example for such non-linear factors are poses which contain an orientation component. Another requirement before being able to perform optimization is the initialization of all random variables with a concrete value. For this, we must first find some scheme to estimate all required random variables outside the scope of the factor graph. Finally, the optimization can be performed, for example, by using a Levenberg-Marquardt style optimizer, which is a mixture between gradient-descent based optimization and the Gauss-Newton method [12].

3.3. Error Metrics

In this section, we elaborate on all instances in this work, where the notion of *error* occurs. First, we define the distance between positions and orientations. Then, we show a method for intentionally creating errors position and orientation errors of some magnitude. Lastly, we discuss an interesting metric that distributes the position error equally over all cameras.

Position and Orientation Errors

In order to evaluate the quality of an extrinsic calibration, we can compare it to an existing reference calibration. Therefore, we must define an error metric for

each type of data that defines a camera pose, namely its position and orientation components. In order to find the error between an estimated camera pose and the corresponding reference camera pose, we consider both components independently and describe them by a scalar distance. For the distance $d_{pos}(\cdot,\cdot)$ between two position vectors, we simply use the Euclidean distance or L^2-norm defined by

$$d_{pos}(\vec{T}_0, \vec{T}_1) = ||\vec{T}_0 - \vec{T}_1||_2 = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}\,, \tag{3.10}$$

where the Cartesian coordinates of the position vector $\vec{T}_i$ are defined by

$$\vec{T}_i = (x_i, y_i, z_i)^\mathsf{T}\,. \tag{3.11}$$

However, finding the distance between two orientations is not as straightforward. First, we must discuss the different representations that an orientation can take on. We define an orientation vector, analogously to the position vector, by a 3D vector of Euler angles

$$\vec{R}_i = (\alpha_i, \beta_i, \gamma_i)^\mathsf{T}\,. \tag{3.12}$$

Typically, we only use this representation for illustrative purposes, as it is intuitive for us to understand. Adopting the same distance measure

$$d'_{rot}(\vec{R}_0, \vec{R}_1) = ||\vec{R}_0 - \vec{R}_1||_2 \tag{3.13}$$

w.r.t. Eq. 3.10, does not yield the desired result. In particular, we want to find the smallest possible rotation about a single rotation axis through the origin of the coordinate system.

Computational methods involving orientations predominately use rotation matrices or quaternions as means of representation. In this context, quaternions are often preferred for efficiency reasons and because they are not prone to the problem of gimbal-lock [31]. We use the quaternion representation to define an error metric between two orientations.

A quaternion $\vec{Q}$, as the name suggests, is a 4-tupel defined by

$$\vec{Q} = q_0 + q_1\,\mathbf{i} + q_2\,\mathbf{j} + q_3\,\mathbf{k}\,, \tag{3.14}$$

where $q_0, q_1, q_2, q_3 \in \mathbb{R}$ and $\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$ are imaginary units defined by

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{i}\,\mathbf{j}\,\mathbf{k} = -1\,. \tag{3.15}$$

This definition implies a number of interesting and unique properties w.r.t. the algebraic operations between quaternions. For a more detailed introduction on

quaternions, please refer to the works of Kuipers [32][33]. However, it is worth pointing out, that the quaternions form a non-commutative algebra $\mathbb{H}$ over the real numbers $\mathbb{R}$. In order to represent orientations in 3D space, we use $\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$ to denote the standard orthonormal basis for $\mathbb{R}^3$ by

$$\mathbf{i} = (1, 0, 0) \qquad \mathbf{j} = (0, 1, 0) \qquad \mathbf{k} = (0, 0, 1)\,. \tag{3.16}$$

The components q_1, q_2, and q_3 then specify the axis of rotation, while q_0 specifies the rotation angle around this axis.

To convert Euler angles to a quaternion representation, there exist a number of conventions w.r.t. to the order of the sequence with which the three Euler angles are applied [32][34]. We map a rotation vector $\vec{R}$ to a quaternion $\vec{Q}$ by

$$\mathbf{m}_{VQ}(\alpha, \beta, \gamma) = \begin{bmatrix} \cos(\gamma/2)\cos(\beta/2)\cos(\alpha/2) + \sin(\gamma/2)\sin(\beta/2)\sin(\alpha/2) \\ \sin(\gamma/2)\cos(\beta/2)\cos(\alpha/2) - \cos(\gamma/2)\sin(\beta/2)\sin(\alpha/2) \\ \cos(\gamma/2)\sin(\beta/2)\cos(\alpha/2) + \sin(\gamma/2)\cos(\beta/2)\sin(\alpha/2) \\ \cos(\gamma/2)\cos(\beta/2)\sin(\alpha/2) - \sin(\gamma/2)\sin(\beta/2)\cos(\alpha/2) \end{bmatrix}. \tag{3.17}$$

In order to compare two quaternions, we must ensure that they are both normalized to one. A unit quaternion of norm one obeys

$$||\vec{Q}||_2 = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1\,. \tag{3.18}$$

Any nonzero quaternion $\vec{Q}$ can be scaled to unit length by

$$\frac{\vec{Q}}{||\vec{Q}||_2} = \frac{q_0}{||\vec{Q}||_2} + \frac{q_1}{||\vec{Q}||_2}\,\mathbf{i} + \frac{q_2}{||\vec{Q}||_2}\,\mathbf{j} + \frac{q_3}{||\vec{Q}||_2}\,\mathbf{k}\,. \tag{3.19}$$

Finally, we are prepared to compute the distance between two orientations represented as quaternions. However, there exist many different methods to obtain such a distance, all of which fulfill different properties. Hunynh [35] presents and compares a number of metrics for 3D rotations. We adapt one of the presented metrics (ϕ_3) to return an error in the range $[0, 2\pi]$ defined by

$$d_{rot}(\vec{Q}_0, \vec{Q}_1) = \arccos(2 \cdot \langle \vec{Q}_0, \vec{Q}_1 \rangle^2 - 1)\,, \tag{3.20}$$

where the inner product $\langle \vec{Q}_0, \vec{Q}_1 \rangle$ is defined by

$$\langle q_0 + q_1\,\mathbf{i} + q_2\,\mathbf{j} + q_3\,\mathbf{k}\ ,\ q_0' + q_1'\,\mathbf{i} + q_2'\,\mathbf{j} + q_3'\,\mathbf{k} \rangle = q_0 q_0' + q_1 q_1' + q_2 q_2' + q_3 q_3'\,. \tag{3.21}$$

We use the presented metrics not only for comparing camera poses, but also for

other intermediate results that contain a position or an orientation component.

Obtaining an Initial Estimate for the Extrinsic Calibration

The method proposed in this work requires priming with a rough estimate of the extrinsic calibration w.r.t. the targeted multi-camera system. There exist many ways to obtain such an estimate. In practice, these estimates could come from utilizing existing measurements of the room, e.g. a floor plan, using tape measure, or from simply guessing to ones best knowledge. While it is relevant for our method to accommodate for and improve from such inaccurate initial estimates, we also require a reliable reference measurement for evaluation. However, we can utilize this reference calibration to construct initial estimates that have certain properties. In particular, we want to specify a desired error for the position and orientation components of all camera poses. For this, we distinguish between two cases. In the first case, all camera poses are spaced equidistantly to the corresponding reference poses, w.r.t. the distance measures $d_{pos}(\cdot,\cdot)$ and $d_{rot}(\cdot,\cdot)$ and the respective target distances ε_{pos} and ε_{rot} defined by

$$d_{pos}(\vec{T}_i, \dot{\vec{T}}_i) = \varepsilon_{pos} \;\wedge\; d_{rot}(\vec{Q}_i, \dot{\vec{Q}}_i) = \varepsilon_{rot} \quad \forall\, i > 0\,, \tag{3.22}$$

where $\mathcal{C}_i^0 = \{\vec{T}_i, \vec{Q}_i\}$ is the initial estimate for the pose of camera i and $\dot{\mathcal{C}}_i = \{\dot{\vec{T}}_i, \dot{\vec{Q}}_i\}$ is the corresponding reference pose. The directions of these offsets are uniformly distributed.

In the second case, the specified error is normally distributed w.r.t. the additional parameters σ_{pos} and σ_{rot}. We adapt the first approach, such that

$$d_{pos}(\vec{T}_i, \dot{\vec{T}}_i) = t_i \cdot \varepsilon_{pos} \;\wedge\; d_{rot}(\vec{Q}_i, \dot{\vec{Q}}_i) = r_i \cdot \varepsilon_{rot} \quad \forall\, i > 0\,, \tag{3.23}$$

where t_i and r_i are normally distributed random variables, defined by

$$t_i \sim \mathcal{N}(\mu = 1, \sigma^2 = \sigma_{pos}^2) \quad \wedge \quad r_i \sim \mathcal{N}(\mu = 1, \sigma^2 = \sigma_{rot}^2)\,. \tag{3.24}$$

Note that we do not tamper with $\mathcal{C}_0$, as we define this pose to be the origin of our coordinate system.

Distributing the Position Error between all Cameras

In Eq. 3.5 we defined the extrinsic calibration in a way, such that $\mathcal{C}_0$ is the origin of the coordinate system against which the other cameras are positioned. Both, the reference extrinsic calibration, as well as the resulting calibration from our

proposed method use this definition. However, this has an effect on the comparison between the two, and thereby, the resulting error. In particular, the measured position error for $\mathcal{C}_0$ will always be zero, while the underlying error of the proposed calibration will be distributed over the remaining cameras. We solve this issue by estimating the transformation that minimizes the error between the two sets of points, following Umeyamas method [36].

Fig. 3.4 illustrates this method by transforming a set B to B', such that the distance between corresponding points is minimal between B' and A. Umeyamas method allows to include a scaling factor for the optimal transformation alongside the rotation matrix and the translation vector. In this example, the distance error is greatly reduced by applying this scaling factor. More so, the scaling factor gives an interesting insight into the two point sets.

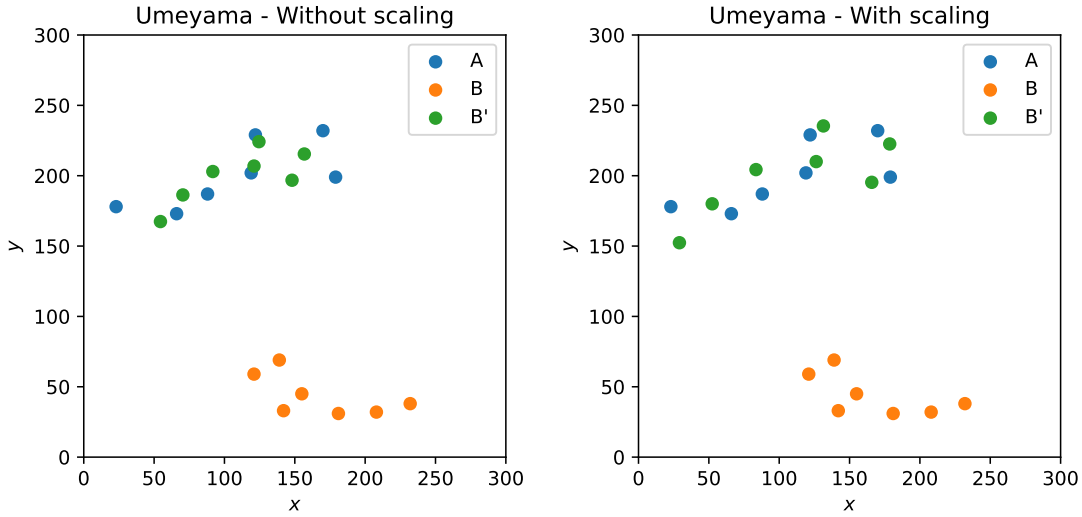

Figure 3.4: Least-squares estimation of transformation parameters between two point patterns A and B according to Umeyama [36].

Employing Umeyamas method to transform the position components of a proposed extrinsic camera calibration in order to minimize the error toward the reference calibration introduces an error > 0 for $\mathcal{C}_0$, while reducing the total error of the system. As the scale of the obtained camera positions might be considered a relevant property of the extrinsic calibration, we do not use scaling for reporting errors. However, we can exploit the optimal scaling factor to assess the scaling ambiguity of the proposed calibration and to report errors that are corrected w.r.t. scaling. To some degree, all camera calibration methods utilizing naturally occurring objects of unknown exact dimensions suffer from scaling ambiguities [1].

4. Method

We propose a method which aims to utilize the image streams of a multi-camera system $\mathcal{S}_N$ with $N > 1$ finite projective cameras $\mathcal{C}_i$, for $i \in [0 \,..\, N-1]$, in order to extract and maximize knowledge about the relative poses between all cameras in real-time, i.e. finding the translation vectors $\Delta\vec{T}_{ij} = (X_j - X_i, Y_j - Y_i, Z_j - Z_i)^\mathsf{T}$ and the rotation vectors $\Delta\vec{R}_{ij} = (\Psi_j - \Psi_i, \theta_j - \theta_i, \varphi_j - \varphi_i)^\mathsf{T}$ between all camera pairs ij for $i, j \in [0 \,..\, N-1]$, where the pose of the optical center of camera i is defined by

$$\mathcal{C}_i = \{\vec{T}_i = (X_i, Y_i, Z_i)^\mathsf{T}, \vec{R}_i = (\Psi_i, \theta_i, \varphi_i)^\mathsf{T}\} \tag{4.1}$$

consisting of the position vector $\vec{T}_i$ and the orientation vector $\vec{R}_i$.
We call this the *extrinsic calibration* of the multi-camera system $\mathcal{S}_N$.

The camera poses $\mathcal{C}_i$ are embedded in one global three dimensional Euclidean vector space for which we define $\mathcal{C}_0$ to be the (static) origin by

$$\mathcal{C}_0 \coloneqq \{\vec{T}_0 = (0\text{m}, 0\text{m}, 0\text{m})^\mathsf{T}, \vec{R}_0 = (0°, 0°, 0°)^\mathsf{T}\}\,. \tag{4.2}$$

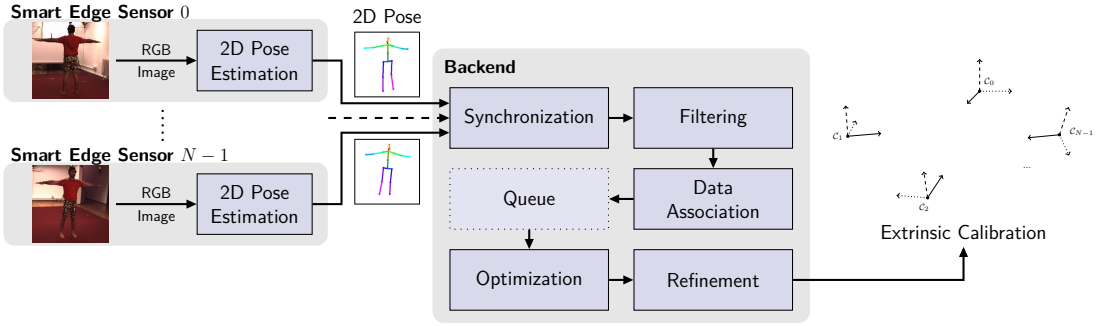

Figure 4.1: Overview of the proposed pipeline for extrinsic camera calibration using smart edge sensors and person keypoint detections. Images are analyzed locally on the sensor boards. Keypoint detections are transmitted to the backend where multiple views are fused to construct and solve optimization problems using factor graphs. A queue decouples the pre-processing and optimization stages.

The optical center of each camera can be viewed as the origin of its own local coordinate system in which we can express position vectors as $\vec{p_i} = (x, y, z)^\mathsf{T}$ and orientation vectors as $\vec{\omega_i} = (\alpha, \beta, \gamma)^\mathsf{T}$ relative to $\mathcal{C}_i$. When rotating these local coordinate systems within the global vector space, we refer to rotations around the three Euler angles α, β and γ as pitch, yaw and roll. In its local coordinate system, each camera looks straight down the z-axis in positive direction, or the roll axis, respectively. The local coordinate system of $\mathcal{C}_0$ resembles the global vector space.

Fig. 4.1 gives an overview of our proposed pipeline. Each camera stream is fed into an onboard person keypoint detector capable of detecting multiple persons simultaneously. We refer to the unity of camera and detector as *smart edge sensor*. The keypoint detections are transmitted to a central backend to be processed further. The *pre-processing stage* first synchronizes the detection streams into sets of time-corresponding detections. Each frame-set is then filtered to remove redundant and noisy detections after which data association is performed, where the correspondences between person detections from multiple views are found. Corresponding person detections are fused to form a person hypothesis. After pruning unique detections, person hypotheses with at least one valid detection are considered eligible to be used in the optimization stage. To conclude the pre-processing stage, eligible person hypotheses are analyzed and attached to a queue, which serves to decouple the pre-processing stage from the rest of the pipeline. The *optimization stage* repetitively reads from this queue, selects several person hypotheses, according to a fitness function and uses them to construct and solve an optimization problem in the form of a factor graph [12]. The *refinement stage* updates the *current estimate of the extrinsic calibration* by smoothing the intermediate results generated by the optimization stage and estimates the calibration error in order to configure the next optimization cycle. Given a suitable sequence of person detections in areas where the FoV of at least two cameras overlap, the extrinsic calibration will converge to an optimum.

In order to apply this method, some requirements must be fulfilled. Each camera in $\mathcal{S}_N$ must provide an RGB image stream, together with a unique timestamp for every image. The FoVs of all cameras must overlap in such a way that $\mathcal{S}_N$ forms a connected graph. Two FoVs are considered to be overlapping when a person fits in the considered area. Furthermore, we assume the calibration matrices $\mathbf{K}_i$ to be known for all cameras i. The corresponding distortion coefficients $\vec{\text{dist}}_i$ can be provided to increase the precision of the resulting extrinsic calibration. Lastly, this method must be initialized with a rough estimate of the extrinsic calibration. A detailed discussion on these requirements can be found in Sec. 6.3.

4.1. Person Keypoint Detection

It is fair to assume, that most multi-camera systems are placed in such way, that the observed areas of all cameras are easily accessible by humans. Therefore, instead of using traditional calibration targets, like checkerboards or AprilTag [4] grids, for example, we want to use the persons that are present in the scene as calibration targets, to gain knowledge about the extrinsic calibration of a multi-camera system. Traditional calibration targets are often favorable from a methodological point of view, because exact dimensions of the targets are known and the problem of data association can be solved easily. However, the use of person calibration targets does have some advantages over the traditional methods. In particular, its often much easier to have persons walking around the scene, instead of crafting and positioning traditional targets. Furthermore, persons are dynamic objects and can easily cover large areas of the FoVs of all cameras. They can be viewed and detected from any angle and they have a relatively large baseline, all allowing for and contributing to persons being suitable and effective calibration targets. On the contrary, the dimensions of a person are not precisely known, which creates a scaling ambiguity in the resulting extrinsic calibration.

In order to utilize persons as calibration targets, we directly adopt the method for person keypoint detection proposed by Bultmann *et al.* [24], with one exception: The semantic feedback loop, described in this work, which improves on the quality of the detections, is not used, as it relies on a known extrinsic calibration. The deployed smart edge sensors are equipped with an RGB camera and a tensor processing unit (TPU), running inference of a lightweight MobileNetV3 feature extractor [37], adopted for human pose estimation, locally on the sensors. This architecture allows the person keypoint detection of all cameras, to be executed in parallel and real-time. Furthermore, transmitting compact keypoint data to the backend directly, instead of transmitting larger size image data, results in lower latencies, due to the lower computational overhead needed for transmission and implies softer requirements on the network infrastructure, w.r.t. bandwidth. The clocks of all sensors are software synchronized. This allows for a precise comparison of the timestamps associated with the detections between different sensors, later on in the pipeline.

The feature extractors generate heatmaps encoded as multi-channel images, where each channel reflects the confidence of one specific human joint, being present at a pixel location. From these heatmaps, the sensors derive messages of the following format: For each image, the sensor $\mathcal{C}_i$ transmits a person keypoint detection message $\{\mathcal{D}_j^p\}_i$, where joint j corresponds to person p.

A single keypoint detection is defined as

$$\mathcal{D} = \{ (u, v)^\mathsf{T}, c, \mathbf{\Sigma} \}, \tag{4.3}$$

where $(u, v)^\mathsf{T}$ are the image coordinates, $c \in [0, 1]$ is the confidence and $\mathbf{\Sigma} \in \mathbb{R}^{2\times2}$ is the covariance of the detection. Only detections with a confidence above zero are considered as valid detections. Invalid detections are always included in each message, using a confidence of zero. Note that the order of p is arbitrary, for each consecutive message of one sensor, or between time-corresponding messages of multiple sensors. The order of j is fixed and follows the definition, established for the COCO dataset [20], using a total number of 17 joints (See Tab. 4.1). The covariance $\mathbf{\Sigma}$ describes the uncertainty region around the detection, referring to the initial heatmap, where the highest confidence c is at $(u, v)^\mathsf{T}$. All detections d_i of a sensor $\mathcal{C}_i$ are annotated with the same timestamp t_i.

Table 4.1: Ordered list of detectable joints according to the COCO dataset [20].

0: nose	1: left_eye	2: right_eye
3: left_ear	4: right_ear	5: left_shoulder
6: right_shoulder	7: left_elbow	8: right_elbow
9: left_wrist	10: right_wrist	11: left_hip
12: right_hip	13: left_knee	14: right_knee
15: left_ankle	16: right_ankle	

While the sensors are capable of detecting multiple persons simultaneously, the number of detections does increase the demand on their limited computational resources. In particular, too many simultaneous detections can cause drops in frame rates and increased latencies, making it more difficult to find a time-corresponding set of detections between all cameras and leading to negative effects on the goal of performing extrinsic calibration. Therefore, as many persons as necessary, but as few as possible, should be present in the scene during the calibration procedure. In practice, the deployed sensors (See Fig. 4.2) are capable of detecting up to three persons simultaneously, at uncompromised frame rates of 30 FPS and latencies around 2ms [24].

4.2. Pre-Processing Stage

The pre-processing stage is the first stage of the pipeline being located at the backend. It receives N person keypoint detection streams and processes them so that they can be used for optimization. First, incoming streams are synchronized into

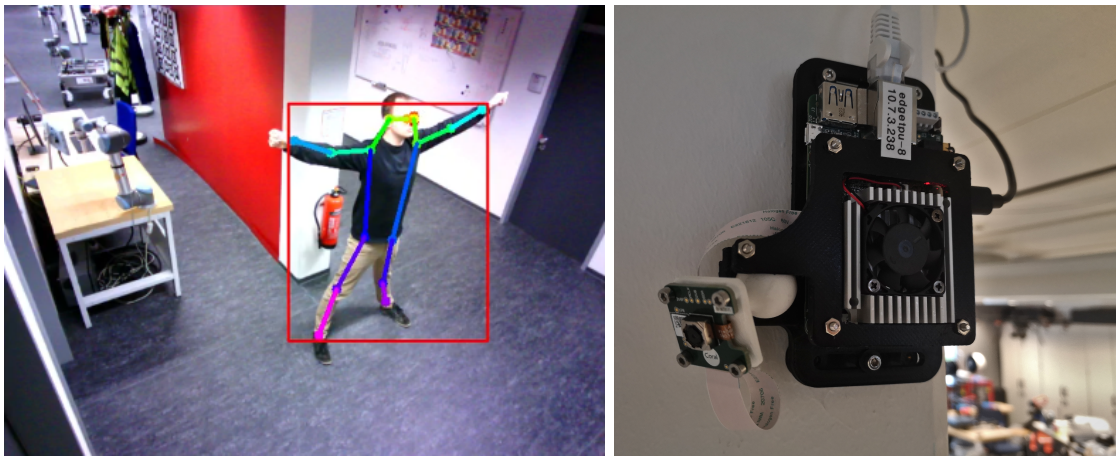

Figure 4.2: Visualization of the person keypoint detections for one person with the corresponding RGB image shown in the background for illustration purposes (left). Image of one deployed sensor running TPU based inference locally and directly transmitting person keypoint detections to the backend (right).

sets of time-corresponding detection messages of size N. In the following, we will refer to such a set as a *frame-set*. After synchronization, the only mandatory step that must be performed before being able to carry out optimization is data association, where correspondences between detections from different sensors within a frame-set are obtained. This step is necessary because the optimization stage uses constraints based on multi-view geometry, so that any 3D joint is required to be observed from at least two cameras.

However, we apply various filter constraints on each frame-set, before handing them to the optimization stage, in order to forward only those detections, that are considered accurate and suitable for contributing to improving the extrinsic calibration. Detections can be noisy, w.r.t. their temporal neighbors, which is reflected in their confidence values. Also, false detections can occur. For example, person-like objects, e.g. humanoid robots or coat stands, can be wrongly identified as persons, which is usually indicated by very noisy detections. Furthermore, many detections are redundant, whenever there is little movement in the scene and thus, do not contain any new knowledge to learn from. Fortunately, the rate of incoming detections is much larger, than the rate of detections feasible to be processed by the optimization stage. This requires us to align the two rates, by dropping enough frame-sets, to avoid the waste of computational resources on analyzing frame-sets that will and should not be used in the optimization stage. As the computational effort for a frame-set that passes all filter constraints, as well as all consecutive steps in the pre-processing stage, is much higher, than the computational effort for a frame-set, that is rejected by an early filter stage, the filter constraints should

be set strict enough, to let the pre-processing frame rates not drop significantly below the sensor frame rates. Otherwise, valuable incoming detections might be missed, due to the computational resources being already used at full capacity.

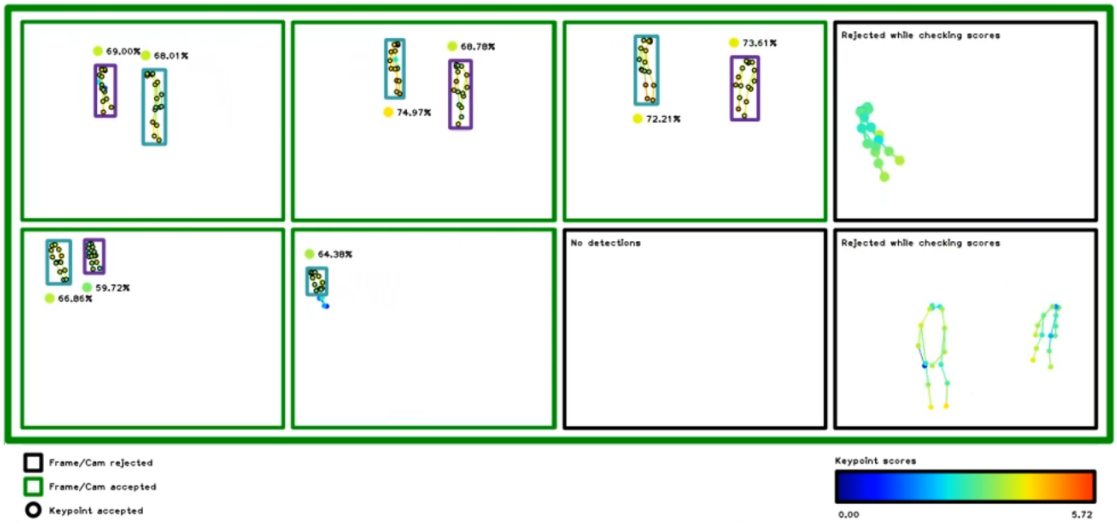

Figure 4.3: Illustration of the pre-processing stage where a synchronized set of detections from eight sensors is filtered and associated. Keypoint detection confidences are colored according to a heat color-map. Accepted sensors and keypoints are highlighted while rejections are provided with reasoning. Corresponding person detections between sensors are surrounded by bounding boxes of the same color.

4.2.1. Synchronization

In order to extract knowledge about the extrinsic calibration of a multi-camera system from incoming person keypoint detection messages, we must analyze sets, with one detection message from each sensor, where the timestamps of all messages are as close to each other as possible. This ensures, that detections in one frame-set correspond to images, that all captured the scene at the same point in time, but from different viewing angles. In other words, there was no significant movement of any person in the interval between the timestamps of the first and last message.

The person keypoint detection streams from each sensor arrive at the pre-processing stage in an asynchronous fashion. As described earlier, cameras within the sensors are not hardware-synchronized w.r.t. their frame rates. Assuming that all cameras precisely operate with r frames per second, we must expect a maximum timing offset of $\pm 1/(2 \cdot r)$ seconds. Furthermore, sensors with many persons in their FoV might momentarily drop their frame rates, due to computational

limitations. Other phenomena, such as jitter or caching, contribute to further deviations. The clocks of all sensors are software synchronized, within an accuracy of approximately two milliseconds. This allows for a precise determination of the timing offset between detection messages from different sensors. While all of the above problems are reflected in the timestamp of each message, we must also consider the latency between capturing the initial RGB images and the arrival of the corresponding detection messages at the backend. Predominantly, the inference for person keypoint detection on each sensor, as well as the transmission over a network infrastructure to the backend, contribute to this.

We perform synchronization by deploying the *Approximate Time Synchronizer* [38], which is provided by the *Robot Operating System* [39]. It buffers all incoming messages for a time, that exceeds the maximum offset that is to be expected between the arrival of corresponding detection messages at the backend, in order to find sets of one message for each sensor, with a minimal timing offset between their timestamps. By this, the matching quality of the resulting sets solely depends on the message timestamps and is independent from the latency that is required for person keypoint detection and transmission to the backend. Furthermore, the algorithm satisfies some properties that we consider desirable:

- A message can only be assigned to one set.
- For any two sets, the timestamps from one set, are either all smaller or all larger, than the timestamps from the other set, w.r.t. to each sensor.
- As few messages as possible are dropped, in that at least for one sensor, there is no dropped message between two consecutive sets.

Higher buffer sizes allow to accommodate for larger inference and transmission latencies, at the cost of increasing the throughput latency of the pipeline. Further timing-related constraints, that consider statistical properties of the distribution of the timestamps within a frame-set, will be applied during the filtering step.

4.2.2. Filtering

Filtering aims to reject false, noisy, redundant, and low-quality detections, while letting the most useful detection pass through. We do this by setting up various constraints for each frame-set, that either address the number of detections by each sensor, the timestamps associated to each sensor, or the confidence value of each detection:

- If a frame-set has no detections by any sensor, or only detections by one sensor, we reject the entire frame-set. The reason for rejecting frame-sets

with detections by only a single sensor is that we cannot possibly apply any multi-view geometry on these detections, which is required during the optimization stage.

- If the number of person detections for a sensor exceeds a specified number of persons j_{max}, all detections of this sensor are rejected. This helps to resolve scenarios, where both, valid detections and false detections occur simultaneously in one sensor. Additionally, we can use this parameter to automatically reject all detection during moments, where the number of persons in the scene is too large for a successful calibration. This can improve the overall quality of the used detections and help to solve the problem of data association.

- For all sensors with at least one valid detection, we compute the maximum span between the first and the last timestamp, as well as the mean and the standard deviation of all timestamps. If this maximum span exceeds a specified value Δt_{max}, or the standard deviation exceeds a specified value σ_{max}, we reject the entire frame-set. If the deviation of the timestamp from a single sensor exceeds the standard deviation, multiplied by some constant weight w_σ, all detections from the respective sensor are rejected.

- We apply a similar approach to address the confidence values of each detection. If the standard deviation of the valid confidence values for a person exceeds a specified value σ^c_{max}, all detections corresponding to this person are rejected. If a single detection deviates further than this standard deviation, multiplied by some constant weight w^c_σ, the respective detection is removed from the person. Instead of considering the maximum span of the detections for each person, we simply reject all detections below some threshold c_{min}. Additionally, we define a minimum number of valid detections per person detection and a set of specific joints that are required to be valid within each person detection. Person detections that violate these constraints are rejected. In particular, we demand valid detections of both hips and both shoulders for each person, in order to perform data association.

Note that most filter constraints are interactive with one another. Rejecting a detection due to one filter constraint can lead to a violation of another filter constraint. Suggesting a set of parameters, that works well for any scenario, is difficult, as this choice depends heavily on the scene and sensor layouts, the implementation and parametrization of the person keypoint detection, the network infrastructure, as well as the available computational resources on the backend. Setting them up

carefully, w.r.t. to a concrete scenario, is essential for obtaining a precise extrinsic calibration.

After filtering, we use the distortion coefficients of each camera, if available, and undistort the coordinates of all valid detections using the *OpenCV* library [40].

4.2.3. Data Association

To extract knowledge about the extrinsic calibration of a multi-camera system, we will exploit the projective geometry between corresponding detections from multiple views during the optimization stage. Therefore, in the data association step, we try to find these correspondences between the detections from all sensors. Fig. 4.3 illustrates the goal of this step.

While an approach for data association based on multi-view geometry is computationally less expensive than using, for example, an image-based approach, it has one obvious drawback: The geometry of the scene is not precisely known. The relative camera poses are only roughly known at the beginning of the calibration procedure, as finding them, is the overlying goal of this method. Also, the precision of the detections is not exact. Therefore, a geometry-based approach can only be successful, if the precision of both, detections and camera poses is good enough. The necessary precision for this approach to work as intended, will be discussed in Sec. 6.3. Moreover, a method using visual descriptors for data association, which can be easily used in conjunction with or instead of the geometry-based approach presented here, is described in Sec. 7.1.

However, we can try to reduce the required precision in both domains. The precision of the detections is easily boosted, by only letting detections with accurate timing and high confidence pass through the filtering step. The negative effect, caused by inaccurate camera poses, can be neglected to some degree, by rejecting all person detections that are too close to each other, within each sensor. Note that the orientation error of the camera poses is more problematic than the position error, especially for distant detections. Therefore, we will apply further filtering of entire person detections, according to this, before the actual data association takes place. In particular, we reject all person detections within a sensor, where the bounding box that surrounds the valid detections of a person detection intersect with that of another.

Due to the functioning of the method used for person keypoint detection, we already know the association of every single joint detection to a corresponding person detection. We will exploit this knowledge, by considering the distance of the joints of a person detection, to the respective joints of other person detections.

In order to increase the number of available joints, we will not only use those joint detections, that survived the filtering step, but also consider all other joints of the person detections in question, that exceed some confidence threshold $c_{\text{min\_da}}$, lower than the confidence threshold c_{min} used during filtering. Furthermore, when searching for corresponding candidates of a specific person detection, we can safely exclude all other person detections within the same sensor from the search space. The distances between the joints will be computed within the global coordinate system, by back-projecting 2D detections into 3D rays, and then reducing each ray to a line segment, according to a depth estimation for each person detection. By comparing the average distances of the line segments between all person detections, we obtain an assignment for each person detection to one person hypothesis.

Ray-Casting

First, we back-project 2D detections into global 3D coordinates. As we have not obtained any depth information for these 2D detections yet, each back-projection results in a 3D ray, where the origin of the ray is the optical center of the respective camera, propagating toward infinity and facing the same direction as the camera. Therefore, the 3D coordinates corresponding to a 2D detection must be somewhere, i.e. at some depth, on this ray. We do this for all detections that are considered valid according the filtering step, as well as for all invalid detections, which belong to a person detection that contains at least one valid detection and exceed some confidence threshold $c_{\text{min\_da}}$.

Using the focal length $(f_x, f_y)^\mathsf{T}$ and the principle point $(c_x, c_y)^\mathsf{T}$ from the projection matrix $\mathbf{K}_i$ for camera $\mathcal{C}_i$, all points $\vec{p}_\mathcal{D}$ on the back-projection ray for detection $\mathcal{D}_i$, with the image coordinates $(u, v)^\mathsf{T}$, are expressed in local coordinates by

$$\vec{p}_\mathcal{D}(z) = \begin{pmatrix} (u - c_x) \cdot z / f_x \\ (v - c_y) \cdot z / f_y \\ z \end{pmatrix} , \tag{4.4}$$

where $z \in [0, \infty)$ is the unknown depth of detection $\mathcal{D}_i$. We transform $\vec{p}_\mathcal{D}$ into the global coordinate system by

$$\vec{P}_\mathcal{D}(z) = \mathbf{R}'_i \cdot \vec{p}_\mathcal{D}(z) + \vec{T}_i \, , \tag{4.5}$$

where $\mathbf{R}'_i$ is the rotation matrix corresponding to the rotation vector $\vec{R}_i$, and $\vec{T}_i$ is the position vector of camera $\mathcal{C}_i$.

Depth Estimation

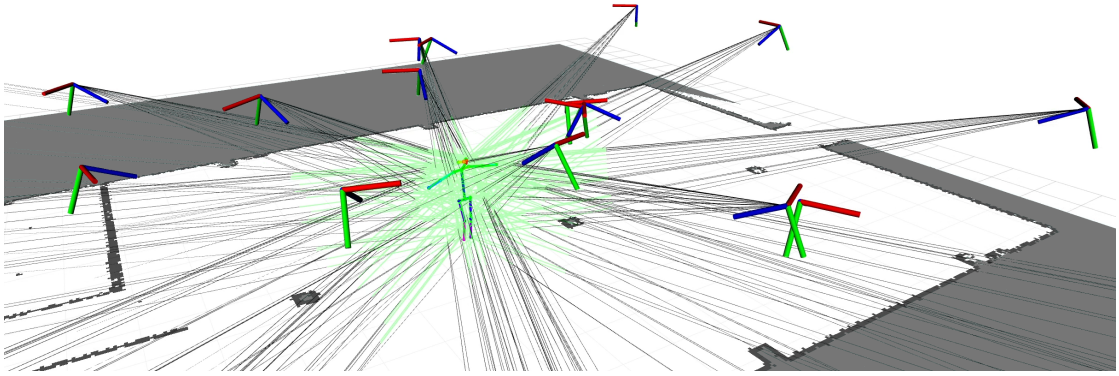

Figure 4.4: 3D back-projection rays embedded in the global coordinate system for the joint detections of one person (black) and the corresponding reduction to line segments after applying depth estimation (green). 3D human pose estimation according to [24] shown for illustration purposes only.

Next, we reduce each ray $\vec{P}_{\mathcal{D}}$ to a line segment, by estimating the interval $[z_{min}, z_{max}]$ in which the depth z of each detection $\mathcal{D}$ lies. We estimate this depth once for each person detection and then apply it to each ray, corresponding to all valid joint detections associated with the respective person detection.

From the size of a person detection we can try to infer its depth, i.e. its distance from the optical center of the camera it was detected by. Using the height of the bounding box, that surrounds all joint detections associated with a person detection, would provide the largest possible baseline and thereby the most accurate result. By specifying parameters for the minimum person size h_{min} and the maximum person size h_{max}, that are to be expected to be present in the scene during the calibration procedure, we could easily compute z_{min} and z_{max} for a person detection p with the bounding box height h_p by

$$\begin{pmatrix} z_{min} \\ z_{max} \end{pmatrix} = \frac{f_x + f_y}{2 \cdot \|h_p\|_2} \begin{pmatrix} h_{min} \\ h_{max} \end{pmatrix} . \tag{4.6}$$

However, using this method would induce three assumptions:

- Each person detection must have at least one valid joint detection on its head (nose/eyes/ears) and one valid detection of an ankle.
- All persons in the scene must always stand upright during the entire calibration procedure.

- All persons in the scene must always stand roughly planar to the image planes of the cameras they are observed by.

When any of the assumptions is violated, the height of the bounding box for a person shrinks in size, leading to a substantial underestimation of z_{min} and z_{max}. Enforcing the existence of the required joint detections as an additional filter constraint is possible, but both head and ankle detections are often occluded in practice, which would lead to many rejections of otherwise useful detections.

Instead, we propose a similar, but more robust approach. In the filtering stage, we enforce the existence of valid shoulder and hip joints for each person detection (See Sec. 4.2.2). Usually, these joints are not occluded and deliver stable detections. Also, they are the joints with the least amount of movement, which reduces the problem of synchronicity between multiple views. By averaging the distances between the detections of the left shoulder and hip, as well as the right shoulder and hip, we obtain the length of the torso in pixels. As the length of the torso, as well as the distances between all four considered joints, can be assumed to as constant for a person, due to the human anatomy, we can drop the assumption of persons always standing upright and support any orientation instead. As the axis of the torso is always roughly perpendicular to the axes between both shoulders and both hips, a camera directed at a person can always see one of the three axes with a distance between the endpoints of an axis > 0, independent from the orientation of the person. In fact, for any orientation of a person toward a camera, at least one of the axes can be seen by the camera with a minimum angle of 45°.

However, to save computational resources, we do not try to estimate the orientation of a person toward the cameras it is observed by, implying that each joint detection of a person detection will use the same depth estimation. Instead, we expect the best case orientation of 90° for the h_{min} case and the worst case orientation of 45° for the h_{max} case, yielding the largest possible interval $[z_{min}, z_{max}]$, defined by

$$z_{min} = \min \left(\frac{f_x + f_y}{2} \cdot h_{min} \cdot \begin{pmatrix} \phi_{shoulders} \\ \phi_{torso} \\ \phi_{hips} \end{pmatrix} \Big/ \begin{pmatrix} \|h_{shoulders}\|_2 \\ \|h_{torso}\|_2 \\ \|h_{hips}\|_2 \end{pmatrix} \right) \tag{4.7}$$

and

$$z_{max} = \min \left(\tfrac{f_x+f_y}{2} \cdot h_{max} \cdot \begin{pmatrix} \phi_{shoulders} \\ \phi_{torso} \\ \phi_{hips} \end{pmatrix} \Big/ \left(\sin(45°) \cdot \begin{pmatrix} \|h_{shoulders}\|_2 \\ \|h_{torso}\|_2 \\ \|h_{hips}\|_2 \end{pmatrix} \right) \right), \tag{4.8}$$

where $\min(\cdot)$ refers to the smallest vector element, h is the measured distance

between the respective joints of a person detection in pixels, and the ratios ϕ relate the height of a person to the considered distances between joints. We obtain the values for ϕ by applying the method of Bultmann *et al.* [24] and measuring the considered joint distances of the generated 3D person models for persons with known height. After carrying out this experiment for a small number of persons, we obtain

$$\begin{pmatrix} \phi_{shoulders} \\ \phi_{torso} \\ \phi_{hips} \end{pmatrix} = \begin{pmatrix} 0.17526 \\ 0.29242 \\ 0.15261 \end{pmatrix} . \tag{4.9}$$

Note that this approach mitigates the problem of the total height of a person not being reflected properly in the keypoint detection data, as the distance between the ankle and eye/ear joints is not capturing the full height of a person.

Finally, we apply the depth estimation, by reducing each ray $\vec{P}_{\mathcal{D}}$ to a line segment

$$\hat{\vec{P}}_{\mathcal{D}}(z) = \vec{P}_{\mathcal{D}}(z)\,, \tag{4.10}$$

for $z \in [z_{min}, z_{max}]$.
Figure 4.4 illustrates this reduction.

Line Segment Distances

As a metric for calculating the distance between two line segments, we use the Closest point-distance described by Wirtz *et al.* [41], of the form

$$\mathcal{D}_{closestpoint}(l_1, l_2) = \min(\mathcal{D}_{\perp}(l_1, l_2), \mathcal{D}_{\perp}(l_2, l_1)). \tag{4.11}$$

Hypothesis Matching

In order to find the correspondences between person detections from multiple views, we deploy an iterative greedy search method, similar to the approach of Tanke *et al.* [25]. Instead of determining the assignment costs for assigning person detections to person hypotheses, based on epipolar distances, we use the introduced line segment distances described above. To further improve the robustness of the approach, w.r.t. the extrinsic calibration not being precisely known, we iterate over all person detections in a specific order. In particular, we utilize the depth estimation $(z_{min} + z_{max})/2$ of each person detection, to sort all person detections by depth in ascending order. This exploits, that near person detections have a relatively short interval $[z_{min}, z_{max}]$ and thereby, a more constrained localization in 3D space.

First, we create a new person hypothesis for the nearest person detection. Then,

we iterate over all other person detection, as described above. For each iteration, we calculate the cost for assigning the person detection to all currently existing person hypotheses. If the lowest assignment cost is smaller than a threshold θ, we assign the person detection to the considered hypothesis. Otherwise, we create a new person hypothesis and assign the person detection to it.

Starting the iteration with near person detections generally increases the probability of more joint detections being valid within a person detection. This allows us to increase the certainty of the assignments, by only making an assignment when a minimum number of shared valid detections > 1 exist on both sides of the comparison. Creating new hypotheses instead of making correct assignments is possible, but this is only a problem when it causes unnecessary pruning. Otherwise, the included detections can still be utilized in the optimization stage. Being conservative with making assignments, and thereby slowing down the convergence of the calibration procedure, is necessary, as wrong associations can cause tremendous distortions in the calibration procedure, especially when they get used repeatedly.

4.2.4. Pruning

In the filtering step, we reject all frames, where there are only detections in one sensor, because we cannot possibly apply or exploit any multi-view geometry for these detections (See Sec. 4.2.2). However, this constraint is not sufficient, to remove all detections of this kind. After data association, it is possible, that a person hypothesis contains joint detections that were only detected by one sensor. For the same reason as above, we do not want to forward such unique detections into the optimization stage, because each joint of a hypothesis must be detected by at least two sensors, in order to contribute to the solution of the optimization problems, created during the optimization stage. Removing unique detections does not directly affect the resulting extrinsic calibration, but it reduces the overall computational complexity of the method, which passively does improve on the results, in case the available computational resources are used at full capacity.

4.2.5. Analysis and Queueing

To conclude the pre-processing stage, we analyze and shape the gathered data in each frame-set, in order to prepare its utilization in the optimization stage.

First, we compute a number properties for each obtained person hypothesis $\hbar$, which will allow for a comparison between multiple person hypotheses from different frame-sets, during the hypothesis selection step (See Sec. 4.3.1) in the optimization stage:

- We count the number of valid detections $\vec{n}_\hbar$ within each person hypothesis $\hbar$ for each camera i.
- We count the total number of valid detections $n'_\hbar$ within each person hypothesis $\hbar$.
- We estimate the center of mass $\vec{\zeta}_\hbar$ for each person hypothesis $\hbar$, by averaging over the center points of all assigned line segments $\vec{\hat{P}}_\mathcal{D}$ within each hypothesis, by

$$\vec{\zeta}_\hbar = \frac{1}{n'_\hbar} \cdot \sum_\mathcal{D} \vec{\hat{P}}_\mathcal{D} \left(\frac{z^\mathcal{D}_{min} + z^\mathcal{D}_{max}}{2} \right) , \tag{4.12}$$

  for all valid detections $\mathcal{D} \in \hbar$.
- We calculate the mean $z^\mu_\hbar$ and the standard deviation $z^\sigma_\hbar$ for the distances between the center of mass $\vec{\zeta}_\hbar$ and the camera position vectors $\vec{T}_\mathcal{D}$, associated with all valid detections $\mathcal{D} \in \hbar$.
- We calculate the mean $v^\mu_\hbar$ and the standard deviation $v^\sigma_\hbar$ of the number of valid detections per joint j for all valid detections $\mathcal{D}_j \in \hbar$.
- We calculate the mean $c^\mu_\hbar$ and the standard deviation $c^\sigma_\hbar$ of the detection confidences $c_\mathcal{D}$ for all valid detections $\mathcal{D} \in \hbar$.
- We calculate the mean $t^\mu_\hbar$ and the standard deviation $t^\sigma_\hbar$ of the timestamps $t_\mathcal{D}$ for all valid detection $\mathcal{D} \in \hbar$.

Finally, we append each person hypothesis $\hbar$, containing its valid detections $\mathcal{D}_\hbar$ and the properties $\mathcal{P}_\hbar$ listed above, to a queue in descending order, w.r.t. their mean timestamps $t^\mu_\hbar$.

4.3. Optimization Stage

The optimization stage processes the person hypotheses obtained in the preprocessing stage, in order to extract knowledge about the extrinsic calibration of the utilized multi-camera system. We do this by repeatedly constructing and solving optimization problems, during the entire calibration procedure. Each optimization problem is constrained by exploiting methods from multi-view projective geometry, as well as prior or heuristic knowledge.

In particular, we construct a factor graph [30] in each optimization cycle (See Sec. 4.3.2), to encode projective constraints, based on a selection of person hypotheses, as well as statistical constraints on the camera poses w.r.t. the obtained

camera poses in previous optimization cycles, or the initial estimate of the extrinsic calibration in case of the first optimization cycle. The optimal selection of person hypotheses used in an optimization cycle is determined by a selection algorithm based on assigning a fitness value for all available person hypotheses (See Sec. 4.3.1). We solve each factor graph for the most likely camera poses, by applying a Levenberg-Marquardt optimization scheme (See Sec. 4.3.4). As an intermediate result, we will estimate 3D poses for all selected person hypotheses (See Sec. 4.3.3).

In this section, we describe one optimization cycle, starting with the selection of appropriate person hypotheses, and ending with the optimization of the constructed factor graph. Adapting parameters and fusing results over multiple optimization cycles, which will improve the convergence of the extrinsic calibration, will be performed in the refinement stage. However, the general approach of the refinement stage will be summarized here, as it is helpful for understanding this section.

Notes on the Refinement Stage

All constraints encoded in the factor graph are equipped with (Gaussian) noise models. They specify the tolerance each constraint is susceptible to, but also allow for controlling the dynamics between individual constraints, or the entire system. In general, applying optimizations with looser noise models will cause larger deviations of all camera poses, w.r.t. their previous pose. In contrast, strict noise models will only allow for small deviations, but increase the risk of failing an optimization cycle, not yielding any new results. At the beginning of the optimization procedure, noise models must be wide enough, to handle a rough initialization of the camera poses. Over time, they must become stricter, in order to enable convergence and to obtain a precise final extrinsic calibration. Mixtures of loose and strict noise models can be used simultaneously, to reflect the individual confidence in single camera poses, detections, or components thereof.

All noise models must be initialized with parameters, that reflect the certainty in the initial estimate of the extrinsic calibration. During the calibration procedure, all noise models are dynamically adapted, by estimating the error of the current calibration, w.r.t. to the trajectory of the camera poses. Specifying wider noise models to start with is possible, but will lead to slower convergence and can lead to locally optimal calibrations, from which it is hard to recover.

The error estimation process, the adaptive control of all noise models, the temporal smoothing of results over multiple optimization cycles, as well as other post-processing schemes, are located in the refinement stage (See Sec. 4.4), which is

executed between all consecutive optimization cycles.

4.3.1. Person Hypothesis Selection

The pre-processing stage extracts person hypotheses from incoming person keypoint detection streams, where each person hypothesis consists of up to 17 joints (See Tab. 4.1), while each joint is detected by a minimum of two and a maximum of N sensors. They are queued up to serve for constraining the optimization problem, which will be constructed in form of a factor graph. Due to the filtering constraints each detection passed in the pre-processing stage, all person hypotheses contained in the queue are considered eligible for this task. The number of selected hypotheses should be specified w.r.t. the scene layout and the available computational resources on the backend. In general, the larger the scene is, the less overlap there is between the FoVs of all cameras. Therefore, the number of selected hypotheses must be large enough, to constrain the majority of cameras in each optimization cycle.

We can improve on the resulting extrinsic calibration of this method, by selecting specific person hypotheses, according to some fitness function, instead of choosing them at random. The goal of hypothesis selection is to select the set of hypotheses contained in the queue, that is the most promising, to yield the largest possible improvement on the current estimate of the extrinsic calibration.

In the final analysis step of the pre-processing stage, we obtained a total of ten properties for each hypothesis. We base the decision on which hypotheses to choose on comparing these properties for available hypotheses. Furthermore, we introduce two additional properties for each hypothesis, concerning its usage and performance during the calibration procedure, which is maintained in the refinement stage.

Retrieving Fitness Values

Let $\mathcal{Q}_t$ be the queue of hypotheses from which we draw our selection at optimization cycle t. We aim to obtain a fitness value $f_\hbar$ for each person hypothesis $\hbar \in \mathcal{Q}_t$, which depends solely on the properties $\mathcal{P}_\hbar$ of $\hbar$. Most of these properties are obtained in the pre-processing stage (See Sec. 4.2.5). In addition, we introduce two new properties $n_\hbar^+$ and $n_\hbar^-$ for each hypothesis $\hbar$, which denote how often $\hbar$ was selected in previous optimization cycles ($n_\hbar^+$) and how often the respective optimizations failed ($n_\hbar^-$). Explicitly, we use the set of properties

$$\mathcal{P}_\hbar = \{n'_\hbar, z_\hbar^\mu, z_\hbar^\sigma, v_\hbar^\mu, v_\hbar^\sigma, c_\hbar^\mu, c_\hbar^\sigma, t_\hbar^\mu, t_\hbar^\sigma, n_\hbar^+, n_\hbar^-\}\,. \tag{4.13}$$

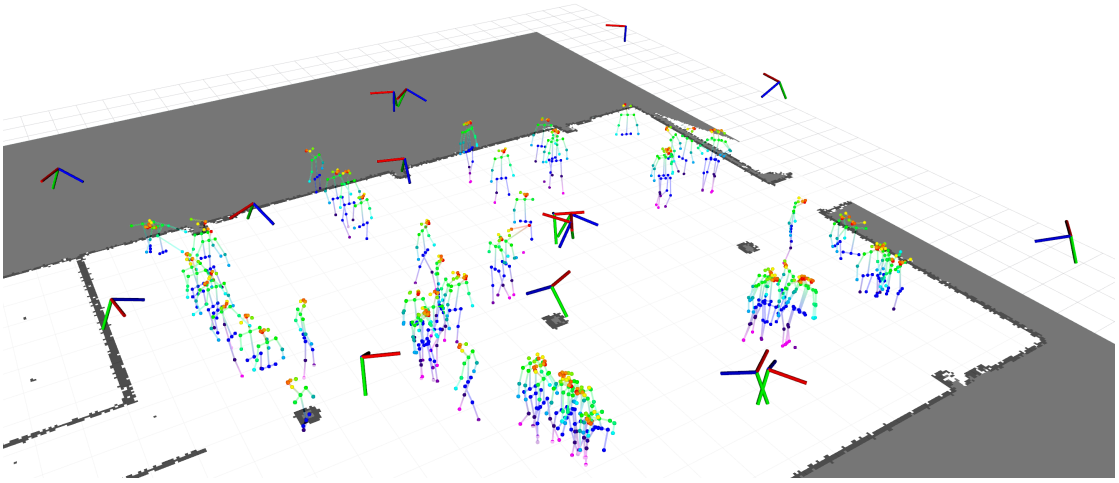

Figure 4.5: Illustration of the selection of hypotheses for a person who has traversed most parts of the room. The skeleton models are obtained after optimization using the selected hypotheses. Free areas are occupied or occluded by objects.

They are easy to interpret, in that for all of them, either larger or smaller values are considered preferable for being selected. In particular, we want to emphasize the selection of hypotheses that have, more valid detections (n'), shorter distances toward the cameras they are observed by (z), more valid detection per joint (v), higher confidence values (c), larger (newer) timestamps ($t^{\mu}_{\hbar}$), smaller timing offsets ($t^{\sigma}_{\hbar}$), fewer previous uses ($n_{\hbar}$) and less failed optimizations during previous uses ($n^{-}_{\hbar}$). Note that for z, v and c, we consider both, their mean μ and standard deviation σ, to obtain more meaningful insights on their distribution. Here, the selection probability must always increase for smaller σ values.

In order to allow for a comparison between the properties of all $\hbar \in \mathcal{Q}_t$, we normalize each property $p \in \mathcal{P}_{\hbar}$ to the interval $[-1, +1]$, by

$$f_p(x) = \frac{x - p_{\min}}{0.5 \cdot (p_{\max} - p_{\min})} - 1\,, \tag{4.14}$$

where f_p is the fitness value for a concrete value x that p takes on, while $p_{\max}$ and $p_{\min}$ are the maximum and minimum values for all instances of p over all hypotheses $\hbar \in \mathcal{Q}_t$. However, this only holds for properties, where an increase in value is considered good. Wherever lower values are considered good, as pointed out above, we must invert f_p.

We introduce yet another property p' for each $\hbar \in \mathcal{Q}_t$, based on the number of detections per camera $\vec{n}_{\hbar}$ contained in hypothesis $\hbar$. For this, we first introduce a set of N global counters $\vec{o}$, which count the number of detections in all previous optimization cycles for each camera in $\mathcal{S}_N$. We normalize the counters to the

interval [0 .. 1], such that the camera with the least detections maps to 1, and the camera with the most detections maps to 0. Then, we multiply both vectors, accumulate their elements, and normalize by the number of detections $n'_{\hbar}$ in $\hbar$, by

$$f_{p'} = \frac{||\vec{o} \cdot \vec{n}_{\hbar}||_1}{n'_{\hbar}} \,. \tag{4.15}$$

This mechanisms rectifies spatial clusters of person hypotheses by enforcing that the selection of under-determined cameras are favored. Over time, this property is supposed to increase the probability for evenly representing all cameras within the optimization stage, once at least some detections are available for all cameras.

Finally, one global set of weighting factors $s_p \geq 0$ allows us to determine the contribution of each property p on the overall fitness value of a hypothesis, defined by

$$f_{\hbar} = \sum_{p} s_p \cdot f_p \,. \tag{4.16}$$

Ensuring Minimum Spacing

So far, we have not considered one property obtained in the pre-processing stage, that plays a unique role in the selection process. We use the center of mass $\vec{\zeta}_{\hbar}$ of each person hypothesis $\hbar$ as a pre-condition in the selection algorithm, to find a selection that ensures a minimum spacing ζ between the center of mass of all selected hypotheses. In each optimization cycle t, we find the mean distance ζ_{μ} between the center of mass $\vec{\zeta}_{\hbar}$ of all hypotheses $\hbar$ in the queue $\mathcal{Q}_t$, by

$$\zeta_{\mu} = \frac{1}{|\mathcal{Q}_t|} \sum_{i=0}^{\lfloor |\mathcal{Q}_t|/2 \rfloor} \sum_{j=|\mathcal{Q}_t|-1}^{\lceil |\mathcal{Q}_t|/2 \rceil} ||\vec{\zeta}_i - \vec{\zeta}_j||_2 \,. \tag{4.17}$$

In order to control the required distance of this spacing constraint, we use s_{ζ} to scale ζ_{μ}, such that

$$\zeta = s_{\zeta} \cdot \zeta_{\mu} \,. \tag{4.18}$$

This ensures, that all selected person hypotheses are well spaced apart from each other, independent from the actual distribution of $\vec{\zeta}_{\hbar}$ existent in $\mathcal{Q}_t$, while at the same time, permitting the selection based on all other properties $p_{\hbar}$ to take effect. As the mechanism for obtaining the fitness value for each hypothesis follows the same idea of being independent of the input distributions, the entire hypothesis selection algorithm is agnostic to the quality and distribution of the hypotheses data. Note that this means, that the selection algorithm does no further filtering

of the data and instead always emphasizes each property equally, regardless of the data it is provided with. Ensuring that all selection hypotheses are spatially spread out to a feasible amount, is the key objective for the selection of hypotheses. By this, we emphasize the construction of optimization problems in each optimization cycle, that are constrained by detections of the majority of cameras. The advantage of training bigger subsets of the multi-camera system jointly is to remove valid, but only locally optimal calibration results from the search space, that consider only a smaller subset of the complete multi-camera system. The effect is, that we increase the probability of obtaining optimization results, that are consistent w.r.t. the complete multi-camera system and thereby, are suitable candidates for resembling the global optimum. Over time, this will lead to less movement in the camera poses between optimization cycles, leading to faster and deeper convergence toward the correct extrinsic calibration.

Selection Algorithm

Let $\mathcal{H}_t$ be the set of M selected person hypotheses from the queue $\mathcal{Q}_t$, for optimization cycle t. First, we calculate the fitness values $f_\hbar$ for all new $\hbar$ in $\mathcal{Q}_t$, based on the specified set of weighting factors s_p, for each property p that contributes to $f_\hbar$. Then, we define $\mathcal{F}$ to be $\mathcal{Q}_t$ sorted in ascending order, such that $\mathcal{F}_0$ is the person hypothesis with the largest fitness value in $\mathcal{Q}_t$. Finally, we calculate the minimum distance requirement ζ between the centers of mass $\vec{\zeta}_\hbar$ for all $\hbar$ in $\mathcal{F}$, based on the weighting factor s_ζ. We start the selection process by adding $\mathcal{F}_0$ to $\mathcal{H}_t$. To select all other $M-1$ hypotheses, we iterate over $\mathcal{F}$ in ascending order, starting with $\mathcal{F}_1$, until the size of $\mathcal{H}_t$ reaches M. For each iteration i, we check if the Euclidean distances between the center of mass $\vec{\zeta}_{\mathcal{F}_i}$ and $\vec{\zeta}_\hbar$, for all hypotheses $\hbar$ in $\mathcal{H}_t$, are below ζ. If this is not the case, we increase i until this condition is met. If i reaches the end of $\mathcal{F}$, we ignore the distance requirement and fill $\mathcal{H}_t$ with the first hypotheses in $\mathcal{F}$, that are not already contained in $\mathcal{H}_t$. Whenever the distance condition is met, we add $\mathcal{F}_i$ to $\mathcal{H}_t$. We refer to Alg. 1 for a pseudocode implementation of this algorithm.

The selection algorithm is fully parameterized by s_ζ and s_p for all properties p. In practice, we set most parameters to 1, except those that condition the same fundamental attribute. For example, $c^\mu_\hbar$ and $c^\sigma_\hbar$ both concern the confidence values within a hypothesis. In order to not be overly selective w.r.t. confidence values, we could use $s_{c^\mu_\hbar} = 0.5$ and $s_{c^\sigma_\hbar} = 0.5$. Also, if the initial estimation of the extrinsic calibration is known to be good, $s_{t^\mu_\hbar}$ can be set near zero, in order to increase the number of selections to draw from, while using a feasible length for $\mathcal{Q}$ w.r.t. memory usage.

Algorithm 1: Person Hypothesis Selection. Selecting M person hypotheses $\hbar$ with the highest fitness values $f_\hbar$ from the queue $\mathcal{Q}_t$ at optimization cycle t, while ensuring a minimum spacing ζ between all selected hypotheses $\mathcal{H}_t$. $\mathcal{F}$ is defined as $\mathcal{Q}_t$, sorted in ascending order by $f_\hbar$, where $f_\hbar$ is parameterized by $\vec{s_p}$. ζ is defined as the average distance between the center of mass $\vec{\zeta}_\hbar$ for all person hypotheses $\hbar \in \mathcal{Q}_t$, scaled by s_ζ. The second while-loop handles the case in which the ζ-constraint cannot be ensured.

Data: Queue of hypotheses $\mathcal{Q}_t$ at time t; scaling parameters $\vec{s_p}$ and s_ζ
Result: Selection of M hypotheses $\mathcal{H}_t$

$\mathcal{F} \leftarrow \mathcal{Q}_t, f_\hbar, s_p$;
$\zeta \leftarrow \mathcal{Q}_t, s_\zeta$;
$\mathcal{H}_t \leftarrow \{\{\mathcal{F}_0\}\}$;
$i \leftarrow 1$;
while $|\mathcal{H}_t| < M \wedge i < |\mathcal{F}|$ **do**
    if $||\vec{\zeta}_{\mathcal{F}_i} - \vec{\zeta}_{\mathcal{H}_j}||_2 < \zeta \ \ \forall\, j \in \mathcal{H}_t$ **then**
        $\mathcal{H}_t \leftarrow \mathcal{H}_t \bigcup \{\mathcal{F}_i\}$;
    end
    $i \leftarrow i + 1$;
end
$i \leftarrow 1$;
while $|\mathcal{H}_t| < M$ **do**
    if $\mathcal{F}_i \notin \mathcal{H}_t$ **then**
        $\mathcal{H}_t \leftarrow \mathcal{H}_t \bigcup \{\mathcal{F}_i\}$;
    end
    $i \leftarrow i + 1$;
end

4.3.2. Factor Graph Construction

A person hypothesis resembles a set of corresponding person keypoint detections from multiple views. From these 2D observations, we try to find the corresponding 3D locations in the global coordinate system. We use these 3D locations as an intermediate result, to find the optimal camera poses that are consistent with all considered 3D locations.

For each optimization cycle t, we construct a factor graph $\mathcal{G}_t$, by using a selection of person hypotheses $\mathcal{H}_t$. A factor graph is a bipartite graph, for which we refer to one partition as *variable nodes* and the other partition as *factor nodes*. Variable nodes represent the unknown random variables of the optimization problem, which are the 3D skeleton models for all person hypotheses in $\mathcal{H}_t$ and all considered camera poses of the multi-camera system $\mathcal{S}_N$. Before optimization can be performed, all variable nodes must be initialized with a concrete numeric value that we need to estimate beforehand. Factor nodes constrain the variable nodes, by encoding any available knowledge about the underlying distribution of the considered random variables. In particular, this refers to the obtained observations contained in $\mathcal{H}_t$, as well as the resulting camera poses from previous optimization cycles. Each factor node uses a noise model that reflects the confidence in the constraint it represents, controlling how susceptible the connected variable nodes are to a change in value. Tuning and balancing all noise models, allows us to determine whether the estimated 3D skeleton models will adapt to explain the camera poses or vice versa. As we carefully filter and select the observations used in the factor graphs, we generally trust them and want the camera poses to adapt and explain them. Once $\mathcal{G}_t$ is constructed, we optimize it to find the most likely solution for all variable nodes. In particular, we are interested in the estimated camera poses of $\mathcal{S}_N$. If the optimization succeeds, we will obtain a new estimate for all camera poses that were constrained in $\mathcal{G}_t$. We smooth these estimates over multiple optimization cycles (See Sec. 4.4.1) to obtain a new current estimate of the extrinsic calibration. We use this current estimate to initialize the next optimization cycle. On the other hand, if the optimization problem is ill-posed, the optimization will fail and not yield any new solution. The mechanism for obtaining $\mathcal{H}_t$ is designed to avoid this from happening, but this does not create any further problems, such that we can simply continue with the next optimization cycle t. Figure 4.6 illustrates $\mathcal{G}_t$.

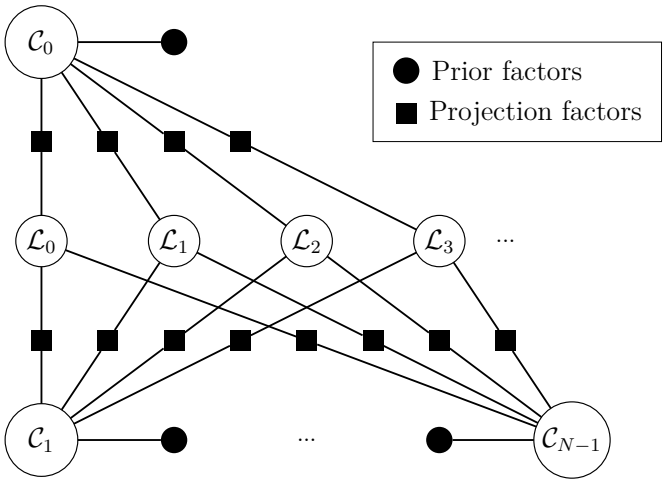

Figure 4.6: Factor graph with camera variable nodes for the camera poses $\mathcal{C}$ and landmark variable nodes for the 3D joint positions $\mathcal{L}$. Camera and landmark nodes can be connected via binary projection factors to constrain the reprojection error of a person keypoint detection. Each landmark node must be connected to at least two projection factors for allowing triangulation. All camera nodes are connected to a unary prior factor that encodes the uncertainty of the camera pose.

Variable Nodes - Cameras & Landmarks

In a factor graph, variable nodes are used to depict the unknown random variables of the optimization problem. We deploy two types of variable nodes, which we refer to as *camera nodes* and *landmark nodes*.

Camera nodes represent the camera poses

$$\mathcal{C}_i^t = \{\vec{T}_i^t, \vec{R}_i^t\}\,, \tag{4.19}$$

where $\vec{T}_i^t$ is the position vector and $\vec{R}_i^t$ is the orientation vector of camera i in the multi-camera system $\mathcal{S}_N$ for $i \in [1\,..\,N-1]$ at optimization cycle t, embedded in the global coordinate system. Note that we only deploy one variable node for each camera $\mathcal{C}_i^t$ in a factor graph $\mathcal{G}_t$, while the selected person hypotheses $\hbar \in \mathcal{H}_t$ consist of detections from different points in time $t_\hbar^\mu$. This approach implies the assumption, that all true camera poses are static. Here, the notion of *true camera poses* refers to the ideal extrinsic calibration, which we try to approximate with this method. However, by increasing the emphasis for selecting person hypotheses with newer timestamps $t_\hbar^\mu$ in the hypothesis selection algorithm, we can loosen this assumption and allow for small or slow movements of all true camera poses during the calibration procedure.

Landmark nodes represent a 3D-joint location of a person hypothesis in the

global coordinate system. In contrast to camera nodes, landmark nodes only represent a position vector without an orientation component. During each optimization cycle t, we deploy one landmark node for each contained joint in the selected person hypotheses $\mathcal{H}_t$. For camera nodes, we possess a direct connection between the sets of nodes for the factor graphs $\mathcal{G}_t$ of two consecutive optimization cycles t, whereas, for landmark nodes, there is no such relation. Therefore, we must estimate the 3D locations of all landmark nodes, based on the corresponding 2D detections contained in the selected person hypotheses $\mathcal{H}_t$.

Factor Nodes - Priors & Projections

Factor nodes connect to variable nodes and constrain them. We distinguish between unary factor nodes, that connect to a single variable node, and binary factor nodes, that connect to two variable nodes.

We equip every camera node $\mathcal{C}_i^t$ with a dedicated prior factor, which is a unary factor node. Prior factors encode prior knowledge about the random variable, which is represented by the variable node they are connected to. In this case, they specify a value for $\mathcal{C}_i^t$, as well as a 6D Gaussian noise model, that reflects the uncertainty of the specified value. In particular, the noise model specifies the uncertainty for all three components of the position $\vec{T}_i^t$ and all three components of the orientation $\vec{R}_i^t$ of $\mathcal{C}_i^t$, providing control over each individual component of the pose. We refer to these noise models as $\vec{N}_{\mathcal{C}_i^t}$. We initialize them by specifying $\vec{N}_{\mathcal{C}_i^0}$ for $i > 0$ to reflect the uncertainty of the initial estimate for the extrinsic calibration. For all following optimization cycles $t > 0$, $\vec{N}_{\mathcal{C}_i^t}$ will be adapted by the refinement stage. The above definition excluded the case $i = 0$. As described earlier, $\mathcal{C}_0^t$ serves as the static origin for the global coordinate system. Therefore, we lock $\mathcal{C}_0^t$ in place, by specifying all components of $\vec{N}_{\mathcal{C}_0^0}$ to be zero. The refinement stage will then retain $\vec{N}_{\mathcal{C}_0^t}$ for all $t > 0$.

We use projection factors to encode constraints based on person keypoint detections. They are binary factor nodes and connect camera nodes to landmark nodes. In particular, for every detection $\mathcal{D}$, contained in all selected person hypotheses $\mathcal{H}_t$ in optimization cycles t, we place a projection factor between the camera node that corresponds to the camera that $\mathcal{D}$ was detected by, and the landmark node that corresponds to the joint that $\mathcal{D}$ represents. Projection factors calculate the reprojection error for a 2D detection w.r.t. the corresponding camera pose and landmark position. In order to do this, we must provide the camera calibration matrix K_i of the respective camera. As we do not include these calibration matrices as random variables in the optimization problem, we assume them to be known and constant. Note that every landmark node is connected to at least two

projection factors, as the pre-processing stage is pruning all unique detections. This property is necessary to enable triangulation.

4.3.3. Variable Initialization

Before the constructed factor graph $\mathcal{G}_t$ can be optimized, we must initialize all variable nodes with concrete values.

The initialization of a camera node must coincide with the value of its associated prior factor. Therefore, we initialize the camera nodes, by directly adopting the values used for the prior factor connected to the camera node. Both represent the current estimate for the extrinsic calibration of the multi-camera system. For $t = 0$, we use the initial estimate for the extrinsic calibration, which is assumed to be known. For $t > 0$, the refinement stage will interpret the results of the optimization stage at optimization cycle $t - 1$, in order to estimate the current extrinsic calibration and use it to specify $\mathcal{C}_i^t$.

Triangulation using the Direct Linear Transformation

In order to initialize a landmark node, we must estimate the joint position of the person hypothesis it represents. So far, we have enforced that all joints contained in a person hypothesis are detected from at least two camera perspectives. This allows us two triangulate all joints contained in a person hypothesis. We use the optimal triangulation algorithm, based on the direct linear transformation, as proposed by Hartley and Zissermann [26] to triangulate all joints contained in the selected person hypotheses $\mathcal{H}_t$ in optimization cycle t.

If such a joint is detected from more than two camera perspectives, we use all available corresponding detections to increase the precision of the triangulation result. We control the emphasis of each detection on the triangulation result by assigning weights, that reflect the confidence in each detection w.r.t. the other detections of the same joint. Obtaining these weights is closely related to the approach for selecting person hypotheses, as we use a subset of the same properties, to retrieve a fitness value for all considered detections. In particular, we put a higher emphasis on detections with larger confidence values, on detections with a timestamp that is closer to the mean of all considered detections, and on detections that are closer to the camera they are observed by. We normalize each property to the interval $[0, 1]$ using a similar approach as in Sec. 4.3.1. Lastly, we sum over all terms and divide through the number of properties, to obtain the final weights.

Note that we perform triangulation in every optimization cycle, even when we use a person hypothesis, that was already utilized in a previous optimization cycle.

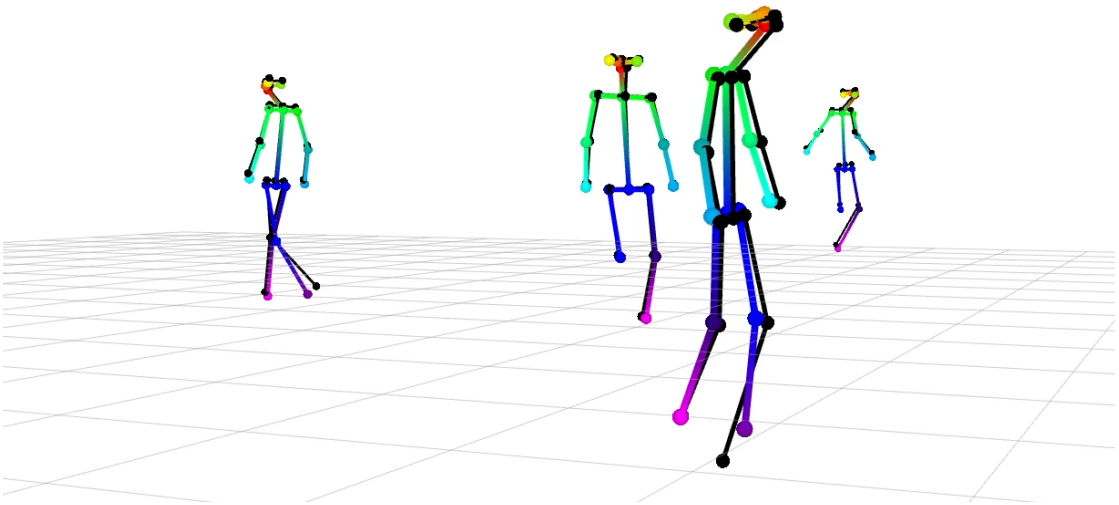

Figure 4.7: Illustration of a multiple 3D skeleton models where the joint locations obtained in the triangulation procedure are depicted by the black skeletons and the corresponding landmark locations after performing optimization are depicted by the multicolored skeletons.

By this, we ensure to update the triangulation results based on the current estimate of the extrinsic calibration, as opposed to reusing previous triangulation results, that are based on an old estimate for the extrinsic calibration. We also need to recompute the confidence weights for each detection, as the property of preferring nearer detections, is dependent on the extrinsic calibration. However, when a person hypothesis is used for the first time, we can store partials of the confidence weights, in order to reuse them when the person hypothesis gets selected again.

4.3.4. Optimization

Once $\mathcal{G}_t$ is constructed and initialized, we can perform optimization. We deploy the default dogleg optimizer provided by the GTSAM framework [30].

The optimization will fail if the variable nodes are underconstrained by the factor nodes, meaning that the system is underdetermined from a mathematical point of view, in that there exists no unique solution. Generally, we try to avoid this, by choosing enough person hypotheses, w.r.t. to the number of camera nodes in $\mathcal{G}_t$. More so, we ensure a minimum distance between all selected person hypotheses, to increase the number of camera nodes being constrained by a projection factor. Also, we only use detections of high quality, w.r.t. several metrics (See Sec. 4.3.1), increasing the probability for obtaining a consistent set of constraints. However, in the beginning of the calibration procedure, when the current estimate of the ex-

trinsic calibration is still relatively inaccurate, the available set person hypotheses to choose from is homogeneous, or when the noise models of the prior factors that constrain the camera nodes are relatively loose, to allow for larger deviations of the camera poses, failures during optimization are likely to happen. In this case, we keep the current estimate for the extrinsic calibration and reuse it for constructing and initializing the next factor graph $\mathcal{G}_{t+1}$. We repeat this procedure, until an optimization is successful.

A successful optimization yields a new candidate for the extrinsic calibration of $\mathcal{S}_N$. We forward this candidate to the refinement stage, where the current estimate for the extrinsic calibration will be updated, based on this candidate. The updated estimate for the extrinsic calibration will then be used for constructing and initializing the factor graph in the next optimization cycle. This way, the obtained knowledge of one optimization cycle, propagates into the next optimization cycles, where it gets fused with the knowledge contained in new or other person keypoint detections. This allows the method to learn over time and minimize the error of the extrinsic calibration.

4.4. Refinement Stage

The refinement stage concludes each optimization cycle. It is executed directly after each (successful or failed) optimization and interprets the result of it, in order to determine its effect on the current estimate of the extrinsic calibration (See Sec. 4.4.1), and to determine the noise models of the prior factors that will be used in the next optimization cycle (See Sec. 4.4.2). Both tasks intent to increase the precision of the current estimate of the extrinsic calibration and the overall robustness of the method. As their effects interact with one another, it is important to parameterize them jointly.

As described in Sec. 4.3.1, we update the properties of all person hypotheses that were selected in the current optimization cycle. In particular, we increment their counter for the optimization-usages, and if the optmization failed, we additionally increment their counter for the optimization-failures. Lastly, if the optimization was successful, we count the number of projections for each camera in the current graph and increment the counters for all cameras accordingly.

4.4.1. Temporal Smoothing

After each successful optimization t, we obtain new candidates $\hat{\mathcal{C}}_i^t$ for the extrinsic calibration of a subset of cameras $i \subseteq \mathcal{S}_N$ in the multi-camera system $\mathcal{S}_N$ that

were constrained by the factor graph $\mathcal{G}_t$. Instead of directly updating the current estimate of the extrinsic calibration $\mathcal{C}_i^t$ with these candidates $\hat{\mathcal{C}}_i^t$, we smooth the candidates with past estimates $\mathcal{C}_i^j$ for some $j < t$ to obtain $\mathcal{C}_i^t$. This further increases the propagation of knowledge about the true extrinsic calibration between optimization cycles, while also accounting for locally consistent solutions, w.r.t to the limited selection of person hypotheses encoded in $\mathcal{G}_t$. This approach significantly boosts the precision that can be achieved by the method.

We deploy an exponential moving average filter, for which the filter coefficients are defined by

$$\epsilon_n = \frac{\lambda(1-\lambda)^n}{\sum_{m=0}^{M-1} \lambda(1-\lambda)^m}, \tag{4.20}$$

for $n \in [0 \,..\, M-1]$, where M is the number of considered past optimization cycles and λ is the smoothing factor of the exponential moving average filter. The denominator normalizes the sum of all filter coefficients to one, which is important to avoid unwanted scaling effects. Note that λ and M parameterize the entire temporal smoothing algorithm.

The filter coefficients can be applied directly to compute the position vector $\vec{T}_i^t$ of the camera pose $\mathcal{C}_i^t$, by

$$\vec{T}_i^t = \epsilon_0 \cdot \vec{\hat{T}}_i^t + \sum_{n=1}^{M-1} \epsilon_n \cdot \vec{T}_i^{t-n}, \tag{4.21}$$

where $\vec{\hat{T}}_i^t$ is the position component of the candidate $\hat{\mathcal{C}}_i^t$.

Obviously, this assumes $t \geq M$, so that we will not use any temporal smoothing at the beginning of the calibration procedure until enough optimization cycles t have been performed. Furthermore, the above definitions only hold if all optimization cycles t finish with a successful optimization and all cameras $\mathcal{C}_i^t$ are constrained in all factor graphs $\mathcal{G}_t$. In practice, this assumption is often violated, which requires us to first collect the $M-1$ past camera poses $\mathcal{C}_i^t$, where i was constrained in $\mathcal{G}_t$ and the optimization of $\mathcal{G}_t$ succeeded.

Averaging over multiple orientations is more cumbersome and requires a different approach. We use the method proposed by Markley *et al.* [42] to determine the weighted average from a set of quaternions. This requires us to provide the orientation vectors $\vec{R}_i^t$ of the camera poses $\mathcal{C}_i^t$ and the orientation vectors $\vec{\hat{R}}_i^t$ of the candidates $\hat{\mathcal{C}}_i^t$ as quaternions. The weights for this method are then specified by ϵ_t.

Note that we use the same filter coefficients, for the position and orientation

components of the camera poses $\mathcal{C}_i^t$. This is important, because both components of a camera pose relate to one another, w.r.t. being a solution for a factor graph $\mathcal{G}_t$. Using different filters for both components would decouple this correlation and create new artificial camera poses, inconsistent with the observations. On the contrary, using the same filter coefficients for both components of $\mathcal{C}_i^t$ averages between existing optimization results, fulfilling the task of smoothing previously obtained knowledge.

We refer to Alg. 2 for a pseudocode implementation of this algorithm.

Algorithm 2: Temporal Smoothing

Data: Candidate poses $\hat{\mathcal{C}}_i^t$; N filter coefficients ϵ
Result: Current estimate for all camera poses $\mathcal{C}_i^t$ at time t

$\mathcal{C}_0^t \leftarrow \hat{\mathcal{C}}_0^{t-1}$;
if $t < N$ **then**
  $\mathcal{C}_i^t \leftarrow \hat{\mathcal{C}}_i^t \; \forall \; i > 0$;
else
  foreach $i \in \mathcal{C}_i^t \setminus 0$ **do**
    if $|\{\mathcal{O}_+^j$ ***is*** $True$ ***and*** $i \in \mathcal{O}_\mathcal{C}^j \,|\, \forall \; j \in [0 \,..\, t-1] \,\}| < N-1$ **then**
      $\mathcal{C}_i^t \leftarrow \hat{\mathcal{C}}_i^t$;
    else
      $\mathcal{L}_{pos} \leftarrow \{\,\}$; $\mathcal{L}_{rot} \leftarrow \{\,\}$; $\leftarrow -1$;
      while $|\mathcal{L}_{pos}| < N-1$ ***and*** $|\mathcal{L}_{rot}| < N-1$ **do**
        if $\mathcal{O}_+^j$ ***is*** $True$ ***and*** $i \in \mathcal{O}_\mathcal{C}^j$ **then**
          $\mathcal{L}_{pos} \leftarrow \mathcal{L}_{pos} \bigcup \{\mathcal{C}_i^j\}_{pos}$;
          $\mathcal{L}_{rot} \leftarrow \mathcal{L}_{rot} \bigcup \{\mathcal{C}_i^j\}_{rot}$;
        $j \leftarrow j - 1$;
      end
      $\vec{T}_i^t \leftarrow \epsilon_0 \cdot \{\hat{\mathcal{C}}_i^j\}_{pos} + \sum_{n=1}^{N-1} \epsilon_n \cdot \mathcal{L}_{pos}[N-1-n]$;
      $\vec{R}_i^t \leftarrow quaternion\_avg(\mathcal{L}_{rot} \bigcup \{\hat{\mathcal{C}}_i^j\}_{rot}\,, \epsilon)$;
      $\mathcal{C}_i^t \leftarrow \{\vec{T}_i^t, \vec{R}_i^t\}$;
    end
  end
end

4.4.2. Error Estimation

The noise models of all prior factors in a factor graph $\mathcal{G}_t$ must be set according to a number of conditions. On the one hand, they must be set loose enough to compensate for an inaccurate initial estimate of the extrinsic calibration. If they would be set too strict in this case, optimizations would be likely to fail and even when they succeed, convergence would be very slow. On the other hand, noise models that are set too loosely lead to high fluctuations in the camera poses $\mathcal{C}_i^t$ between consecutive optimization cycles t. This would prohibit a precise extrinsic calibration. Thus, we want the noise models to adapt over time. In the beginning, they should be set loose enough to accommodate for the quality of the initial estimate of the extrinsic calibration. While the current estimate of the extrinsic calibration is improving, the noise models should become stricter, to reduce the likelihood of high fluctuations, and to increase the obtained precision to a maximum.

As a solution to this problem, we propose a simple algorithm that analyses the trajectories of the camera poses $\mathcal{C}_i^t$ over multiple optimization cycles t, to determine whether the standard deviation, measured against the current camera pose, is increasing or decreasing within this scope. Whenever a camera $\mathcal{C}_i^t$ is converging toward its true pose, we expect a steady reduction of this standard deviation. On the other hand, whenever this standard deviation is increasing, this indicates an increase in error w.r.t. to the true camera pose. We exploit this behavior, by adapting the noise models of each camera accordingly. As it is difficult to precisely estimate the calibration error and to infer suitable noise values from this estimate, we either reduce or increase the noise models by fixed percentages. Allowing the noise models to increase in value can help to escape local minima for the current estimate of the extrinsic calibration, or to compensate initial noise values that were set too small w.r.t. the quality of the initial estimate for the extrinsic calibration. However, this increase should be used conservatively and set to a smaller value than the corresponding decrease. In general, the orientation components of each camera pose converge faster and with greater accuracy. Therefore, we decouple the noise models of the position and orientation components of each camera pose and allow the algorithm to adapt them separately. Furthermore, we limit the minimum and maximum values for both components, in order to avoid unreasonably loose noise models, or noise models that are set so strictly that they practically lock the camera in place, without any chance for it to recover from its current pose. This yields a total of eight parameters to configure the error estimation. We refer to Alg. 3 for a pseudocode implementation of this algorithm.

Algorithm 3: Error Estimation

Data: Optimization success $\mathcal{O}_+^t$; constrained cameras $\mathcal{O}_\mathcal{C}^t$; parameters p
Result: Specification of noise models $\mathcal{N}_i^t$ for cameras i at time t

$\mathcal{N}_0^t \leftarrow \mathcal{N}_0^{init}$;
if $t < N$ **then**
  foreach $\mathcal{C}_{i>0}^t$ **do**
    $\mathcal{N}_i^t \leftarrow \mathcal{N}_i^{init}$;
  end
else
  foreach $\mathcal{C}_{i>0}^t$ **do**
    if $|\{\mathcal{O}_+^j$ ***is*** $True$ ***and*** $i \in \mathcal{O}_\mathcal{C}^j \mid \forall\, j \in [0\,..\,t-1]\,\}| \geq N$ **then**
      $\mathcal{N}_i^t \leftarrow \mathcal{N}_i^{init}$;
    else
      $j \leftarrow -1$; $\mathcal{L}_{new} \leftarrow \{\,\}$; $\mathcal{L}_{old} \leftarrow \{\,\}$;
      while $|\mathcal{L}_{new}| < \lfloor N/2 \rfloor$ ***and*** $|\mathcal{L}_{old}| < \lceil N/2 \rceil$ **do**
        if $|\mathcal{L}_{new}| < \lfloor N/2 \rfloor$ ***and*** $\mathcal{O}_+^j$ ***is*** $True$ ***and*** $i \in \mathcal{O}_\mathcal{C}^j$ **then**
          $\mathcal{L}_{new} \leftarrow \mathcal{L}_{new} \bigcup \{\mathcal{C}_i^j\}$;
        else if $\mathcal{O}_+^j$ ***is*** $True$ ***and*** $i \in \mathcal{O}_\mathcal{C}^j$ **then**
          $\mathcal{L}_{old} \leftarrow \mathcal{L}_{old} \bigcup \{\mathcal{C}_i^j\}$;
        $j \leftarrow j - 1$;
      end
      $\vec{\mu} \leftarrow mean(\mathcal{L}_{new})$;
      $\vec{\sigma}_{new} \leftarrow std(\mathcal{L}_{new})$;
      $\vec{\sigma}_{old} \leftarrow std(\mathcal{L}_{old} - \mu)$;
      foreach $j \in [0\,..\,6]$ **do**
        if $j < 3$ **then**
          if $\sigma_{new}[j] \leq \sigma_{old}[j]$ **then**
            $\mathcal{N}_i^t[j] \leftarrow \max(p_{min}^{pos}, \mathcal{N}_i^{t-1}[j] \cdot p_{dec}^{pos})$;
          else
            $\mathcal{N}_i^t[j] \leftarrow \min(p_{max}^{pos}, \mathcal{N}_i^{t-1}[j] \cdot p_{inc}^{pos})$;
          end
        else
          if $\sigma_{new}[j] \leq \sigma_{old}[j]$ **then**
            $\mathcal{N}_i^t[j] \leftarrow \max(p_{min}^{rot}, \mathcal{N}_i^{t-1}[j] \cdot p_{dec}^{rot})$;
          else
            $\mathcal{N}_i^t[j] \leftarrow \min(p_{max}^{rot}, \mathcal{N}_i^{t-1}[j] \cdot p_{inc}^{rot})$;
          end
        end
      end
    end
  end
end

5. Implementation

In this chapter, we present the implementation for our proposed method. After describing the tools we utilized during development, we describe the central features of our implementation, give advice on its practical application and share insights into the parameter setup procedure.

5.1. Utilized Frameworks & Hardware

The proposed method is implemented in roughly 5000 lines of python code. This does not include the code running on the sensor boards performing 2D pose estimation by Bultmann *et al.* [24], which is publicly available at `https://github.com/AIS-Bonn/SmartEdgeSensor3DHumanPose`. We use the *Robot Operating System* (ROS) [39] as middleware for communication between the sensor boards and the backend, as well as for publishing various intermediate results, and the current estimate for the extrinsic calibration. We use the *NumPy* library [43] for implementing vector algebra operations and the *OpenCV* library [40] for implementing image based computations. Conversions between different representations for orientations are carried out by the *SciPy* library [44]. For construction and optimization of factor graphs we utilize the *GTSAM* framework [30].

In our experiments, we deploy two types of smart edge sensors:

- Sensor type A is based on a *Google EdgeTPU Dev Board* [45], equipped with a quad-core *ARM Cortex-A53* CPU, an Edge TPU, and 1GB of shared RAM. It is connected to the 5-MP RGB camera accessory module with a $87.6°(H) \times 84.0°(V)$ FoV, running at 30 FPS with a resolution of 640×480 pixels. This is the same sensor that is introduced by Bultmann *et al.* [24].

- Sensor type B is based on a *Nvidia Jetson Xavier NX Developer Kit* [46], equipped with a 6-core *NVIDIA Carmel ARM* CPU, 8GB of RAM, and 384 *NVIDIA CUDA* cores. It is connected to a 1-MP *Intel RealSense D455* RGB-D camera with a $90°(H) \times 65°(V)$ FoV, running at 30 FPS with a resolution of 848×480 pixels. We do not utilize the depth sensor of this camera.

While sensor type B is the more powerful one, we run the same method and parameters on both sensor types. In practice, there is no significant difference between the two, w.r.t. to our experiments. Fig. 5.1 shows both sensor boards.

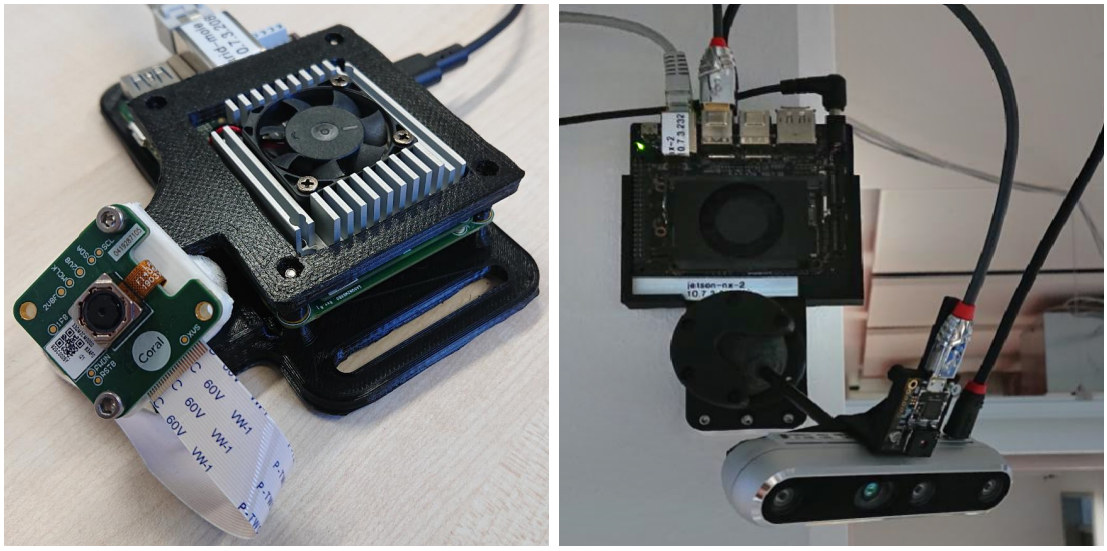

Figure 5.1: Comparison of the deployed smart edge sensor types. Type A (left) is based on a Google EdgeTPU Dev Board [45]. Type B (right) is based on a Nvidia Jetson Xavier NX Developer Kit [46].

The sensor boards are connected via LAN to the backend. The backend runs on a workstation computer equipped with an *Intel Core i9-9900K*, 64GB of system memory, and a *NVIDIA GeForce RTX 2080 Ti* graphics card. We developed and tested this method under ROS *Melodic Morenia* with *Ubuntu 18.04* and *Noetic Ninjemys* with *Ubuntu 20.04*.

5.2. Features & Usage

The implementation can be easily integrated into an existing ROS setup by providing the required data. The initial estimate for the extrinsic calibration, as well as the intrinsic camera parameters, can either be read from a file or from a ROS topic. Furthermore, reference data for the extrinsic calibration and the 3D human pose estimation can be provided, in order to test and evaluate the method, by monitoring and comparing with the current results. We deploy two tools for visualizing various components of the method in real-time. We use the *rviz* 3D visualizer provided by the ROS framework to depict the initial, current and reference camera poses, the estimated and reference 3D human poses, as well as other components, such as ray-casting and depth estimation. Additionally, we use a custom

2D visualizer to depict the received person keypoint detections, where valid detections are marked, reasoning for rejected detections is provided, and corresponding detections between views are illustrated (See Fig. 4.3).

Once the required data is provided, the calibration procedure can be started, using the default parameters. For a practical description on how to obtain a suitable set of parameters for a concrete scenario, please refer to Sec. 5.3. If the initial estimate of the extrinsic calibration is very inaccurate, it is advisable to prefabricate a better initial estimate, by running the method multiple times with very forgiving and loose parameters according to Sec. 5.3, until multiple runs yield similar results. Then, the results can be refined, by running the method with stricter (default) parameters, until all camera poses become stable. The results can be validated, by repeating the calibration procedure with the candidate for the optimal set of parameters and checking if all camera poses reliably converge toward the same poses.

5.3. Parameters

This section gives a brief overview of the essential parameters of our proposed method. While we use a relatively large number of parameters, none of them requires extensive fine-tuning. However, some parameters do need to be adapted for the concrete scenario, w.r.t. the camera and room layouts, the quality of the initial estimate of the extrinsic calibration, as well as the behavior and count of the persons present in the scene. As most parameters interact with other parameters, we categorize them in groups of related parameters, following the structure of Sec. 4. We give a brief description for the parameterss, as well as a default value that proved to be optimal in many scenarios during our experiments (See Sec. 6.2). For more detailed descriptions, please refer to Sec. 4. Finally, we describe the practical effect of the parameters in each group and give a general advice on how to set and use them.

The parameters in Tab. 5.1 must be tuned w.r.t. to the properties of the received person keypoint detections and the available computational resources on the backend. In general, the idea is to filter enough detections for being able to process most incoming detections by the pre-processing stage and not miss a significant amount. Here, the presence of many persons in the scene requires stricter filtering to compensate for the larger amount of obtained detections. A good starting point is to increase the minimum confidence threshold until most inputs get processed. However, the minimum confidence threshold must bet set low enough to let the pre-processing stage find a reasonable amount of person hypotheses. If a lot of

Table 5.1: Parameters for the filtering step.

Parameter	Default	Description
fi_stamps_span	20.0	Maximum timing offset in a frame-set (in ms)
fi_stamps_std_max	5.0	Maximum std. for timing offsets (in ms)
fi_stamps_std_w	1.5	Scaling the std. to remove outliers
fi_scores_min	0.6	Minimum considered detection score
fi_scores_std_max	0.6	Maximum std. for scores
fi_scores_std_w	1.5	Scaling the std. to remove outliers
fi_scores_zeros	8	Max. number of invalid detections/person

movement is to be expected, w.r.t. the persons present in the scene, stricter filtering constraints on the timing offsets help to reduce synchronicity related issues.

Table 5.2: Parameters for Data Association.

Parameter	Default	Description
da_scores_min	0.4	Minimum considered detection score
da_dist_treshold	1.0	Minimum distance to assign hypothesis
da_joints_min	4	Minimum number of joints per hypothesis
da_person_min	1.70	Minimum expected person height (in m)
da_person_max	2.00	Maximum expected person height (in m)

The parameters in Tab. 5.2 must be tuned w.r.t. to the height of the persons present in the scene, the expected spacing between them, and the quality of the initial estimate for the extrinsic calibration. Estimating the geometric distance between all person detections becomes easier, once the current estimate for the extrinsic calibration is relatively accurate and the distance between the persons present in the scene is large. Reducing the distance threshold can help to resolve scenarios where this is not the case. The downside of this is, that a person detection might not be assigned to the appropriate person hypothesis, but instead initialize a new person hypothesis. As unique detections are pruned because triangulation is not possible there, this essentially filters correct associations. This is not a problem, if such a multiples contain enough non-unique detections. Increasing the number of required joints for a valid person hypothesis increases the certainty in the associations, but also acts as a filter for incomplete person detections. This should be set up in conjunction with the maximum number of invalid detection in Tab. 5.1. The minimum confidence threshold for a detection to be utilized during data association should be set slightly lower then the minimum confidence threshold in the filtering step. This effectively increases the certainty

in the associations while not significantly increasing the computational load.

Table 5.3: Parameters for the person hypothesis selection step.

Parameter	**Default**	**Description**
hs_cam_usage	2.0	Scaling effect of equal camera usage
hs_cam_dist_mean	0.25	Scaling effect of mean camera distance
hs_cam_dist_std	0.125	Scaling effect of std. camera distance
hs_valid_mean	0.5	Scaling effect of mean valid detections
hs_valid_std	0.25	Scaling effect of std. valid detections
hs_conf_mean	1.0	Scaling effect of mean confidence value
hs_conf_std	0.5	Scaling effect of std. confidence value
hs_stamps_mean	0.1	Scaling effect of mean timestamps
hs_stamps_std	1.0	Scaling effect of timing offsets
hs_opt_uses	1.0	Scaling effect of previous usages
hs_opt_fails	1.0	Scaling effect of previous opt. failures
hs_spacing	1.0	Scaling effect of minimum spacing

The parameters in Tab. 5.3 are relatively independent from the scenario, as each considered property internally accommodates for its distribution. Multiple properties that originate from the same fundamental attribute, should be weighted lower, such that their sum is roughly one. Setting a factor to zero, bypasses the effect of the property on the fitness values of all person hypotheses. Increasing the effect of up-ranking detections with newer timestamps can be useful to accommodate for cameras that are not mounted perfectly stable and slightly move over time.

Table 5.4: Parameters for the optimization stage.

Parameter	**Default**	**Description**
op_interval	0.5	Interval between optimization cycles (in s)
op_scope	50	Number of person hypotheses per opt. cycle

The parameters in Tab. 5.4 must be tuned w.r.t. to the quality of the current estimate of the extrinsic calibration. In general, there are two strategies that can be applied. If the quality of the current estimate of the extrinsic calibration is not accurate, one should perform many optimizations utilizing a small number of person hypotheses in each cycle, in order to increase the chance for performing a successful optimization by constraining only a few cameras in each cycle. Once the quality of the current estimate of the extrinsic calibration is improving, one should utilize more person hypotheses in each optimization cycle, in order to find results

that are consistent with all camera poses. As this increases the computational load, the optimization interval should be lengthened to accommodate for this.

Table 5.5: Parameters for the error estimation step.

Parameter	**Default**	**Description**
ee_pos_init	0.075	Initial prior position noise value
ee_pos_incr	0.001	Percentual stepsize for value increase
ee_pos_decr	0.002	Percentual stepsize for value decrease
ee_pos_max	0.75	Maximum value
ee_pos_min	0.015	Minimum value
ee_ori_init	2.5	Initial prior orientation noise value
ee_ori_incr	0.001	Percentual stepsize for value increase
ee_ori_decr	0.005	Percentual stepsize for value decrease
ee_ori_max	10.0	Maximum value
ee_ori_min	0.2	Minimum value
ee_history	10	Number considered opt. cycles

The parameters in Tab. 5.5 must be tuned w.r.t. to the quality of the current estimate of the extrinsic calibration. They are used to allow an initialization of the calibration procedure with a relatively rough estimate of the extrinsic calibration, where larger noise values are required, while also enabling convergence toward a precise extrinsic calibration, for which lower noise values are required. In order to enforce this reduction over time, the step-size for increasing noise values should be set smaller then the step-size for decreasing noise values. In general, these parameters should be used conservatively. Smaller step-sizes can be applied for longer calibration procedures. The initial noise values can be reduced if the confidence in the initial estimate for the extrinsic calibration is high.

Table 5.6: Parameters for the temporal smoothing step.

Parameter	**Default**	**Description**
ts_alpha	0.3	Decay for exponential moving avarage
ts_history	10	Number considered opt. cycles

The parameters in Tab. 5.6 must be tuned according to the parameters for the error estimation in Tab. 5.5. In particular, large prior noise values should be accompanied by stronger smoothing and vice versa. Choosing larger alpha values leads to less smoothing as newer poses are used with a higher magnitude. Accordingly, large alpha values should be accompanied by small history values to save computational resources.

6. Evaluation

In this chapter, we evaluate our proposed method. First, we give a detailed description of the setup for our experiments. Then, we carry out experiments to analyze the behavior and the components of our method. Finally, we summarize and discuss the results of our experiments.

6.1. Setup

In this section, we describe the preliminary steps we must take before conducting experiments and evaluating their results. We elaborate on the general scenarios for our experiments, the used reference calibration, the testing procedures and the used initial estimates of the extrinsic calibration.

Scenarios

We conduct all experiments in the robotics hall of our institute which is a large room, with an area of 237m^2 and a height of 3.2m. The cameras are distributed throughout the room at heights between 2.5m and 2.7m. As the room is partly a robotics workshop and partly a regular desk-based workspace, it is densely filled with all kinds of objects, which can cause false detections. One particular challenge with this room is the presence of multiple humanoid robots, which easily get detected by the person keypoint detectors. We do not become proactive in avoiding such wrong detections and trust the method in rejecting them autonomously. Furthermore, tables and pillars can occlude the view of single keypoint detections or complete persons.

We distinguish between two scenarios:

- In *Scenario* 1, we deploy 16 smart edge sensors of type A (See Sec. 5.1). Fig. 6.1 depicts the camera poses within the room and Fig. 2.3 shows the respective RGB images.
- In *Scenario* 2, we deploy 20 smart edge sensors, 16 of which are of type A and 4 are of Type B. Fig. 6.2 depicts the camera poses within the room and Fig. 1.1 shows the respective RGB images.

Note that some sensors are used in both scenarios.

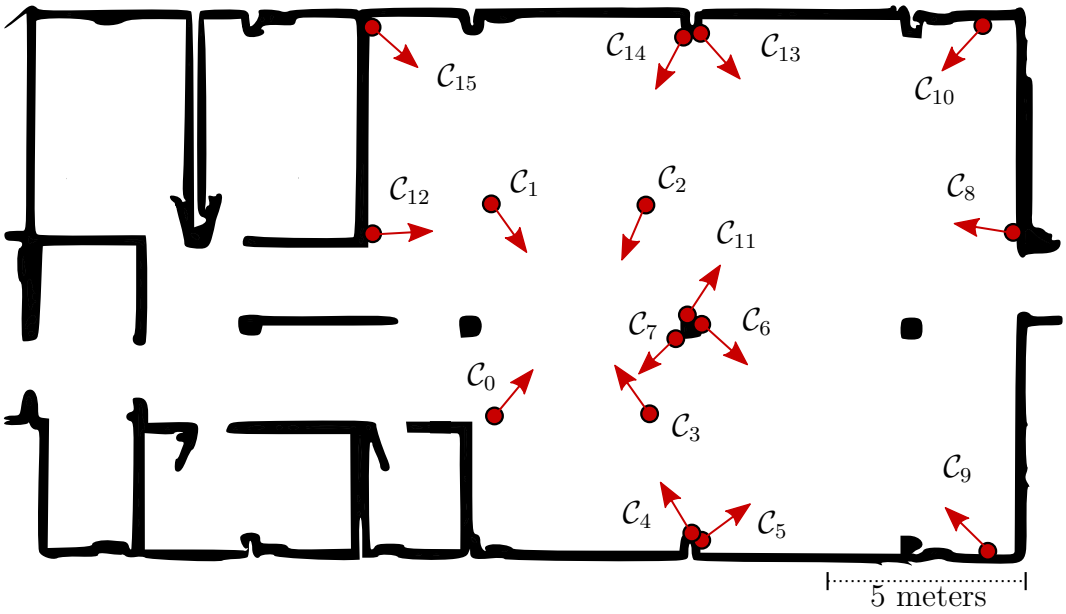

Figure 6.1: Sketched floor plan with camera poses for Scenario 1.

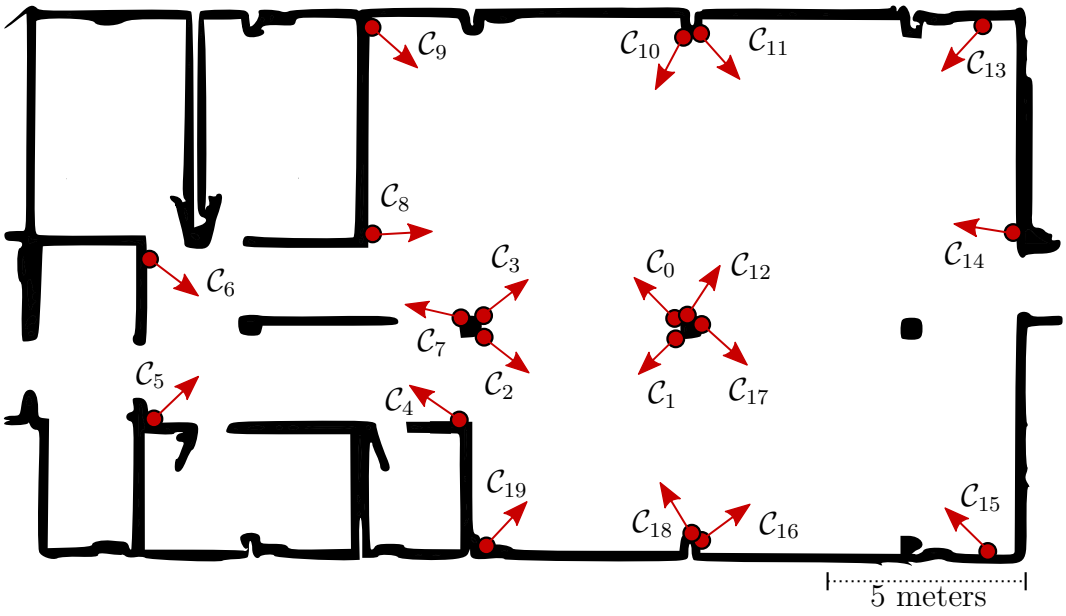

Figure 6.2: Sketched floor plan with camera poses for Scenario 2.

Reference Calibration

In all our experiments, we compare the extrinsic calibration obtained by our method with a reference extrinsic calibration obtained by a traditional calibration method. In particular, we use the *kalibr* [3] toolbox, which provides an offline calibration method that requires the recording of a calibration sequence using an *AprilGrid* calibration target. An Aprilgrid resembles a checkerboard type pattern on a planar surface, where the cells are replaced by *AprilTags* [4]. Since the exact dimensions of the calibration target are known, it is possible to fully resolve its 3D pose, independent of its orientation toward the cameras, even if the target is only partially visible. During the recording of the calibration sequence, the target must be positioned in areas where the FoV of multiple cameras overlaps so that it can be detected simultaneously by as many cameras as possible. Kalibr not only resolves this calibration sequence for the extrinsic calibration the cameras, but also for their intrinsic parameters consisting of a calibration matrix and distortion coefficients for each camera. In order to maximize the precision of the intrinsic calibration, we additionally position the calibration target directly in front of each camera to cover its entire FoV, ignoring its visibility from the other cameras. The calibration sequence is then processed offline by Kalibr in a lengthy optimization procedure. The results of this optimization must be interpreted and manually tuned to obtain the final calibration. We refer to this calibration using the Kalibr toolbox as *reference calibration* or *reference*.

It is worth noting that this reference is not ideal w.r.t. to the true camera poses and the true intrinsic parameters. We performed the calibration with Kalibr multiple times, including the recording of new calibration sequences, and obtained similar but slightly different results.

As described in Sec. 6.1, we evaluate our method on two different scenarios. Tab. B.1 and Tab. B.2 show the concrete extrinsic calibrations that we consider our reference.

Testing procedures

We distinguish between two testing procedures w.r.t. the type of input we use.

In one setup, we do not forward the person keypoint detections transmitted by the sensors directly to the backend. Instead, we record the data first and then play it back later in real-time. From the perspective of the backend, this setup is indistinguishable from using live inputs. The advantage of this approach is being able to playback the same data multiple times. However, due to the real-time nature of our method and its implementation on a computer, where many processes run in a race condition with each other and the scheduler of the

system is switching between them, multiple runs with the same parameters and the same input sequence can yield different results. Therefore, we run each experiment multiple times and report average or representative results wherever possible. This setup allows us to evaluate the effect of specific components or parameters on the calibration procedure.

In a second setup, we forward the person keypoint detections transmitted by the sensors directly to the backend. This setup allows us to analyze and interpret intermediate results in real-time and reactively shape the input data, for example by explicitly walking toward a specific camera and posing in front of it. However, we explicitly do not look at the reported errors and only introduce the reference calibration for evaluation purposes after we consider the calibration procedure finished. Since this is a less controlled environment, this setup represents a realistic application of our proposed method in practice.

Initial Estimates

Our method requires priming with a rough estimate of the extrinsic calibration of the targeted multi-camera system.

In Sec. 3.3, we describe a method for obtaining such estimates by manipulating the reference calibration. Here, we distinguish between two strategies. On the one hand, we enforce that the position and orientation errors between the reference poses and the initial estimate poses exactly match a specified distance, whereas the direction of this distance is picked at random. On the other hand, we do not enforce the initial estimate poses to match the specified distances exactly, but normally distribute them. In both cases, we must specify the desired distances for the position error and the rotation error.

In our experiments, we consider various initial error distances and evaluate the performance of our method to accommodate them. However, the magnitude of this error in a practical application of our method is unclear. To assess the realistic quality of the initial estimate of the extrinsic calibration in practice, we conduct an experiment where we measure the extrinsic calibration of Scenario 2 by hand, using a floor plan and tape measure (See App. A). The results of this experiment suggest that we can expect an average distance error of 0.1737m and 11.14° for cameras $\mathcal{C}_1$ to $\mathcal{C}_{19}$. Obviously, these expected errors do not hold in the general case, as they depend on the camera layout and the applied method of measurement.

6.2. Experiments

In Experiments 1 to 3, we demonstrate the general behavior and usage of our method within Scenario 1 by using a recording of a single person and analyzing the position and orientation errors toward the reference calibration. In Experiments 4 and 5, we switch to live inputs and Scenario 2 where we test our method under realistic conditions with multiple persons. Additionally, we compare the reprojection errors for our calibration results and the reference calibration. Experiments 6 and 7 serve to demonstrate the sensitivity of the method w.r.t. the intrinsic camera parameters and its behavior w.r.t. scaling ambiguities.

Experiment 1

We start our experiments by replaying a recording of a single person walking across the entire room for roughly three minutes within Scenario 1. We use default parameters (See Sec. 5.3) and an initial error of exactly 50cm and 15° in a random direction w.r.t. the reference calibration for all cameras $\mathcal{C}_i$ for $i > 0$. We stop the calibration procedure after the recording ends and analyze the results.

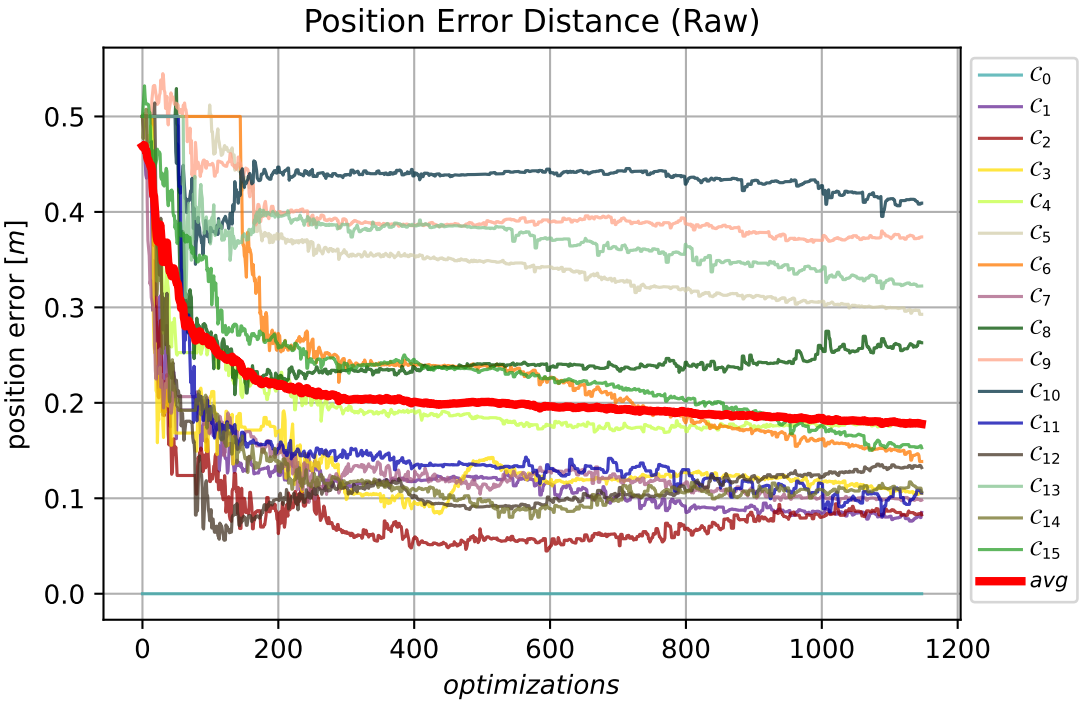

Figure 6.3: Raw position error toward the reference calibration for Experiment 1.

Fig. 6.3 illustrates the raw position errors toward the reference calibration based

on the direct results obtained by our method. We see a fast decrease in error to 0.25m within the first 100 optimization cycles ($\sim$ 20 seconds) and reach 0.2m after 300 optimization cycles. From there, the error continues to decrease slowly and steadily until we stop the calibration procedure at 0.17m after 1100 optimization cycles. Note that some cameras already show a relatively small position error. For example, $\mathcal{C}_2$ reaches an error of 0.04m after only 400 optimization cycles. In general, we observe that cameras with a central position in the room, or cameras that are directed at the larger accessible areas of the room converge faster and deeper compared to cameras that are positioned in or directed at the corners of the room. For this last group of cameras, we can even see the trace of the person at the beginning of the calibration procedure, as the cameras only start to change their position after one of their detections was encoded in a factor graph. This behavior is best visible for $\mathcal{C}_6$.

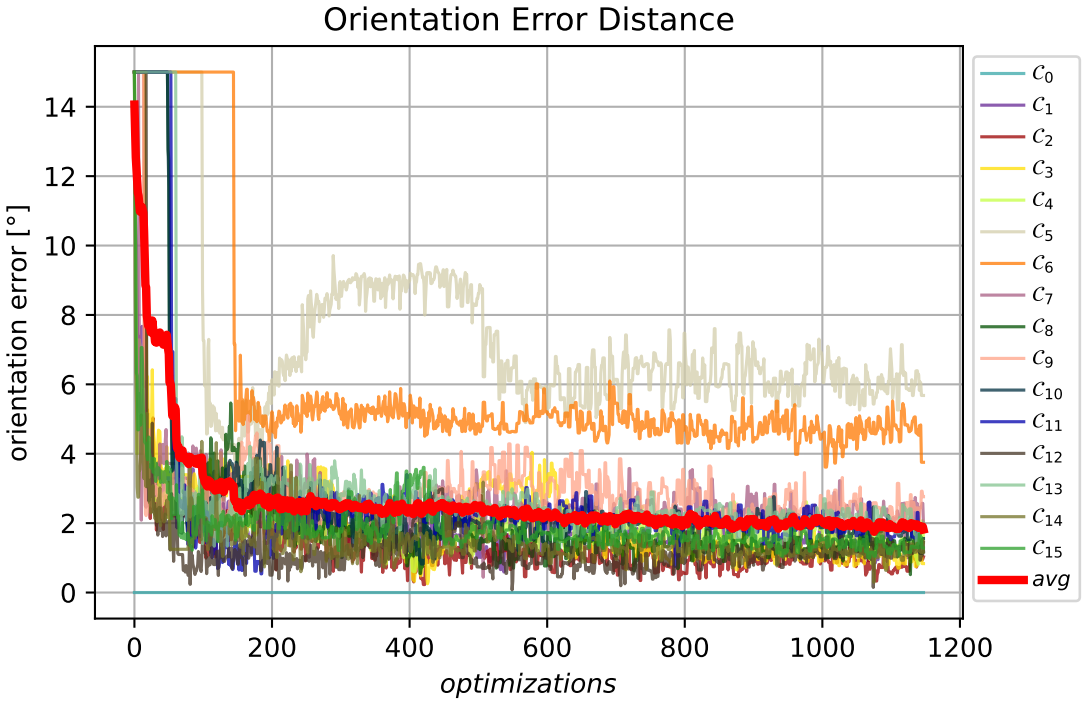

Figure 6.4: Orientation error toward the reference calibration for Experiment 1.

Fig. 6.4 shows the accompanying orientation error. Here, we clearly see the behavior we discussed in the previous paragraph. The orientation errors drop rapidly and reach 2.0° after 200 optimization cycles after which they stay roughly constant for the remaining calibration procedure. Here, the exceptions are $\mathcal{C}_5$ and $\mathcal{C}_6$, which stay between 6.0° and 8.0°. It is worth pointing out that both of them are directed at the same area of the room, which is largely occluded by desks and

monitors with little FoV overlaps w.r.t. other cameras (See Fig. 6.1).

When interpreting the orientation errors, we have to consider the corresponding position errors. As the position error is non-zero, the ideal orientation for such a position also has an error bigger than zero. From this one experiment, it seems like the orientations are close to ideal for the given camera positions. In other words, the hard part of solving the calibration problem seems to be about obtaining the camera positions, while the corresponding orientations are easily resolved and follow along.

So far, we have analyzed the raw errors of the obtained results toward the reference calibration. However, as we define $\mathcal{C}_0$ to be the origin of the coordinate system, we imply that $\mathcal{C}_0$ has an error of 0.0m and 0.0°. In order to make for a fairer comparison, we use Umeyama's method [36] to distribute the position error between all cameras. Fig. 6.5 shows the distributed position error for this experiment. With respect to the raw position error in Fig. 6.3, we see a similarly shaped curve for the average distributed position error, which now reaches a minimum of 0.12m. Furthermore, we see that the distributed position errors of all cameras are closer together. We observe that only $\mathcal{C}_5$ is a clear outlier, which we noticed before when comparing the orientation errors.

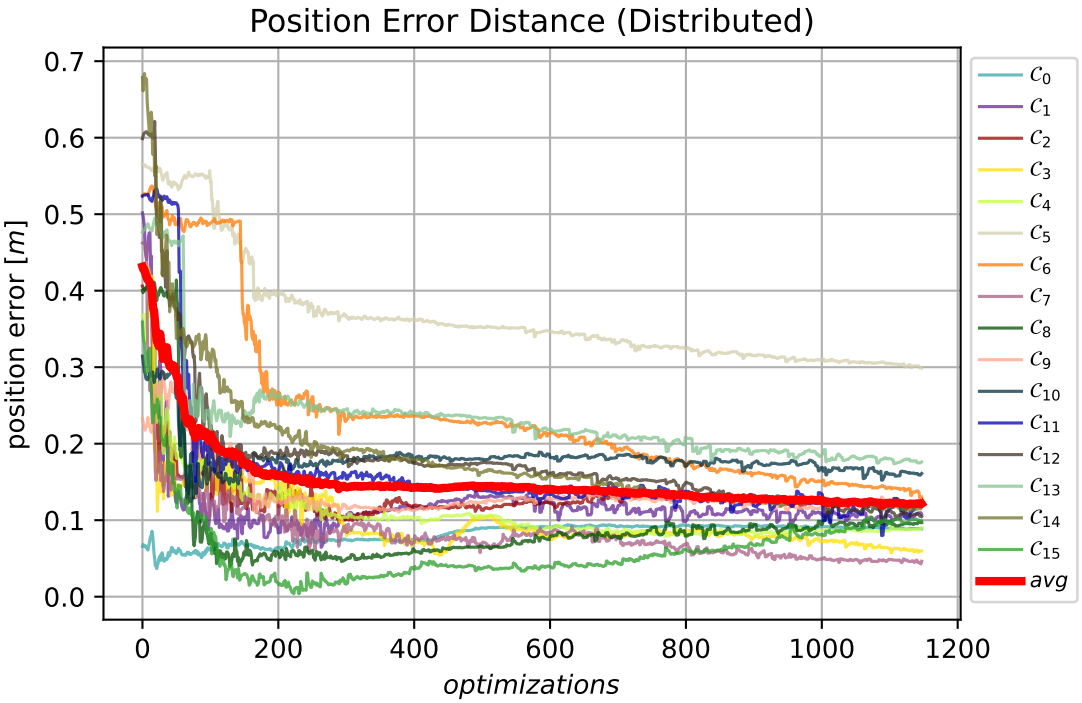

Figure 6.5: Distributed position error toward the reference calibration for Experiment 1.

Experiment 2

We continue Experiment 1 by using its result as the initial estimate of the extrinsic calibration and replaying the same recording of a single person walking across the room. However, we change two parameters to increase precision, while neglecting the reduced robustness of these parameter choices, as we knowingly use a relatively accurate initial estimate. In particular, we increase the number of selected person hypotheses from 10 to 70 and the optimization interval from 0.2s to 0.5s.

Fig. 6.6 illustrates that this further reduces the average distributed position error from 0.12m to 0.08m. From this figure, we can no longer identify a clear outlier as in Experiment 1. All cameras stabilize at distributed position errors between 0.027m and 0.124m. The optimal scaling factor obtained by Umeyama's method for the final camera positions is 0.994, where 1.0 would indicate a perfect resolve of scale, assuming that the reference calibration is free of scaling ambiguities.

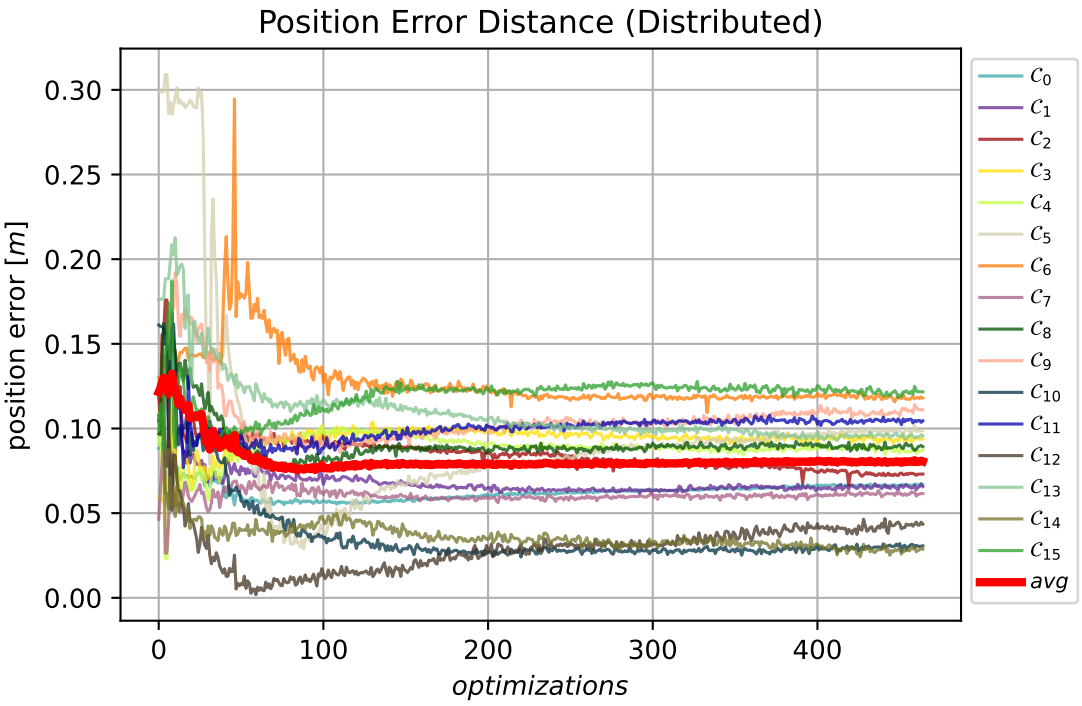

Figure 6.6: Distributed position error toward the reference calibration for Experiment 2.

Fig. 6.7 shows that the average orientation error drops from 2.0° to 1.06°. Interestingly, we can identify $\mathcal{C}_6$ as an outlier, given the logarithmic scaling of the plot, which we earlier identified as one of two outliers in Fig. 6.4. This is easily explained by the fact that the camera mount of $\mathcal{C}_6$ loosened slightly between obtaining the reference calibration and performing Experiments 1 & 2, which we

tried to accommodate for by manually pushing the mount back in place to our best knowledge. Similar to Fig. 6.6, we see relatively stable errors from optimization cycle 200 onward. This indicates an optimal calibration w.r.t. to the gathered person hypotheses.

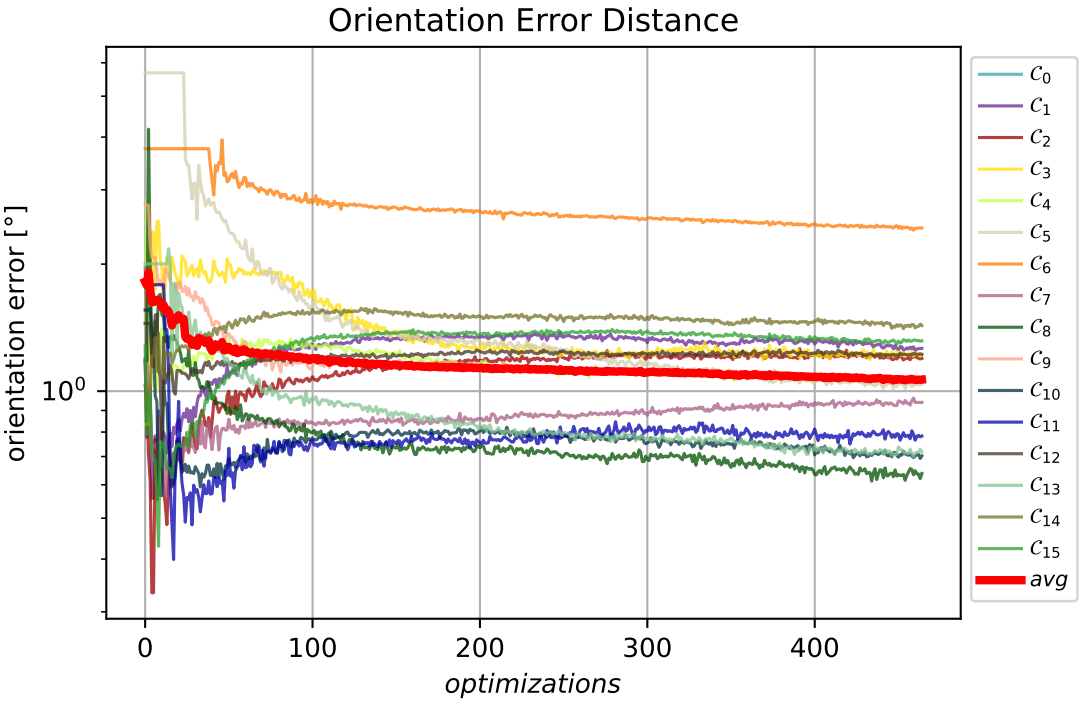

Figure 6.7: Orientation error toward the reference calibration for Experiment 2.

Experiment 3

We repeat Experiment 2 by using the same recording of one person crossing the entire room and the same set of parameters. However, instead of using an initial estimate that was obtained by a previous calibration procedure, we generate a new initial estimate with an error of exactly 0.15m and 15° in a random direction w.r.t. the reference calibration for all cameras $\mathcal{C}_i$ for $i > 0$. Note that this position error is similar to that in Experiment 2, while the orientation error is larger and matches that of Experiment 1.

Fig. 6.8 and Fig. 6.9 show the position and orientation errors for this experiment. All curves look similar to their counterparts in Experiment 2, but this time we achieve a lower average position error of 0.053m and a lower average orientation error of 0.86°. Repeating these experiments multiple times yields similar results and indicates that a systematic error obtained from a previous application of

our method is harder to resolve then random errors obtained from the reference calibration, given that their magnitude is the same. The optimal scaling factor for the shown result is 1.002.

We observe a similar distribution, or ranking, of the errors of specific cameras between the two experiments. Again, $\mathcal{C}_6$ is an outlier and calibrates with the largest errors w.r.t. to its position and orientation. This shows that the final errors are largely independent of the input distribution of the initial estimate. Instead, this behavior must inherit from either the reference calibration, the recording, or possibly, some systematic error within out method. As the cameras with larger errors repeatedly turn out to be ones for which we know the reference calibration not to be ideal, e.g $\mathcal{C}_6$, and this behavior is the same for different input detections, we have reason to believe that a substantial part of the reported errors actually inherits from the inaccuracy of the reference calibration.

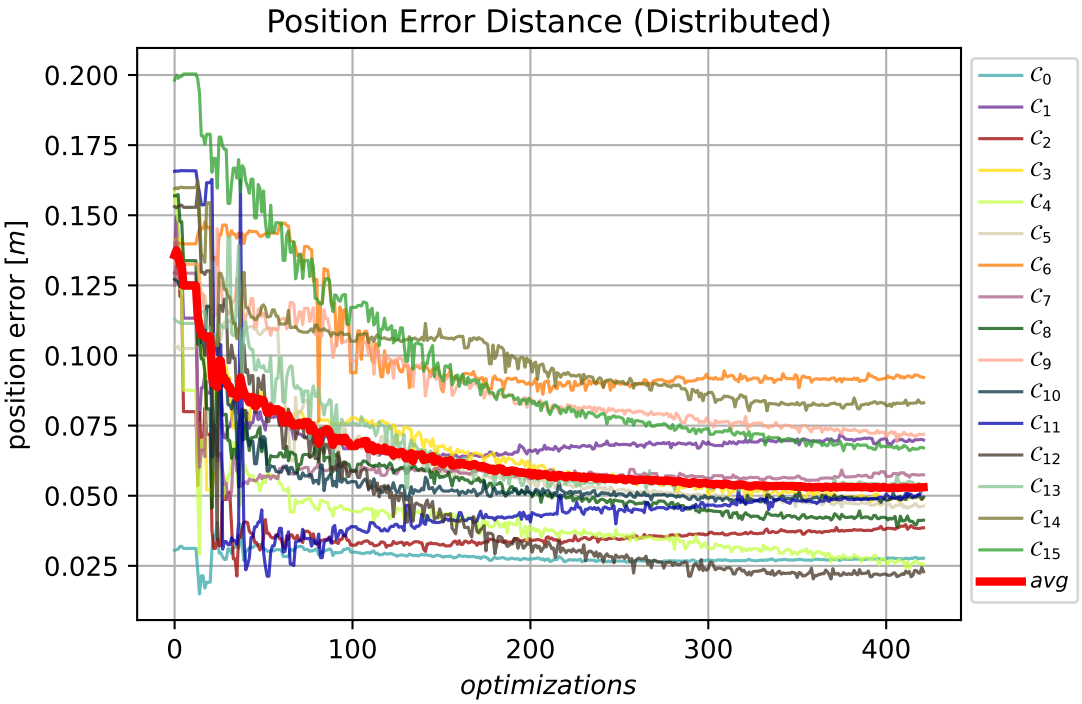

Figure 6.8: Distributed position error toward the reference calibration for Experiment 3.

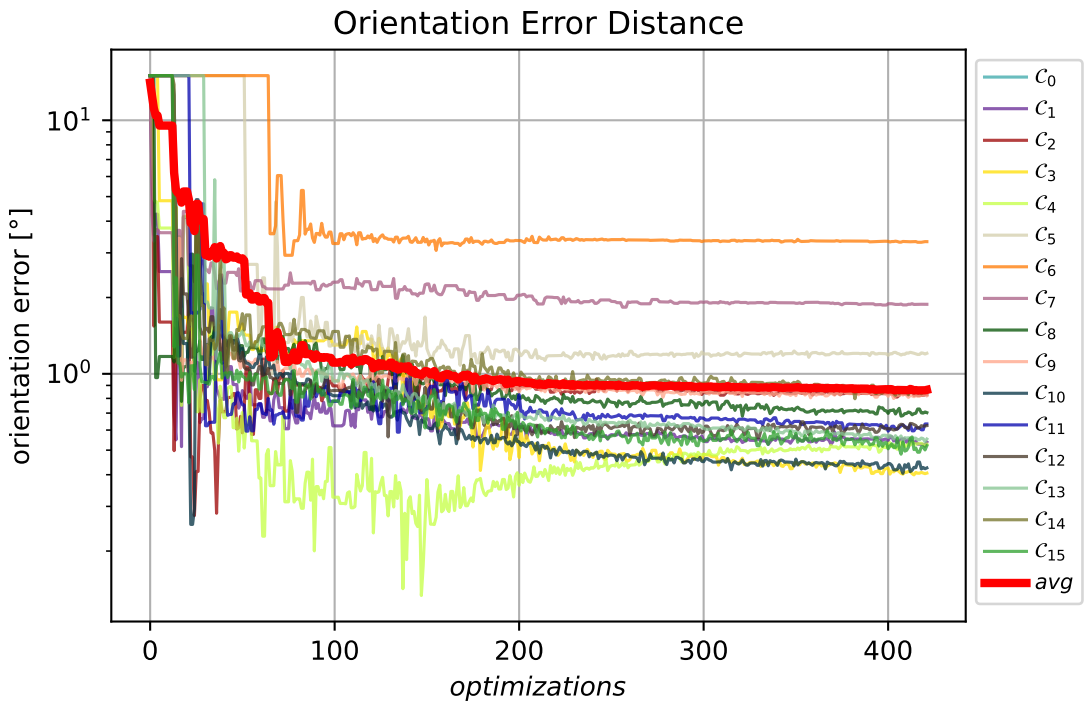

Figure 6.9: Orientation error toward the reference calibration for Experiment 3.

Experiment 4

In this experiment, we switch to Scenario 2 and use live inputs from the sensors. During this experiment, two persons cross the room trying to generate detections in all cameras while one other person is sitting at a desk working. Additionally, a coat stand and two humanoid robots repeatedly generate false detections. We use default parameters and an initial error of exactly 0.20m and 10° in a random direction w.r.t. the reference calibration for all cameras $\mathcal{C}_i$ for $i > 0$.

Fig. 6.10 and Fig. 6.11 show the distributed position error and the orientation error for this experiment. The final average distributed position error is 0.0583m and the corresponding orientation error is 0.4289°. The final position errors range from 0.0107m for $\mathcal{C}_0$ to 0.1197m for $\mathcal{C}_{12}$. Interestingly, we can identify two groups of cameras, where one group centers tightly around a position error of 0.03m, while the other (smaller) group centers around 0.09m with a wider spread. The majority of the optimization takes place in the first 200 optimization cycles or approximately 20 seconds. Afterward, the errors seem to be stable. When looking closely, we can see sinusoidal oscillations with magnitudes of about 0.01m between the cameras. We observed this behavior during other experiments as well. Generally, the frequency of these oscillations increases and their amplitude decreases.

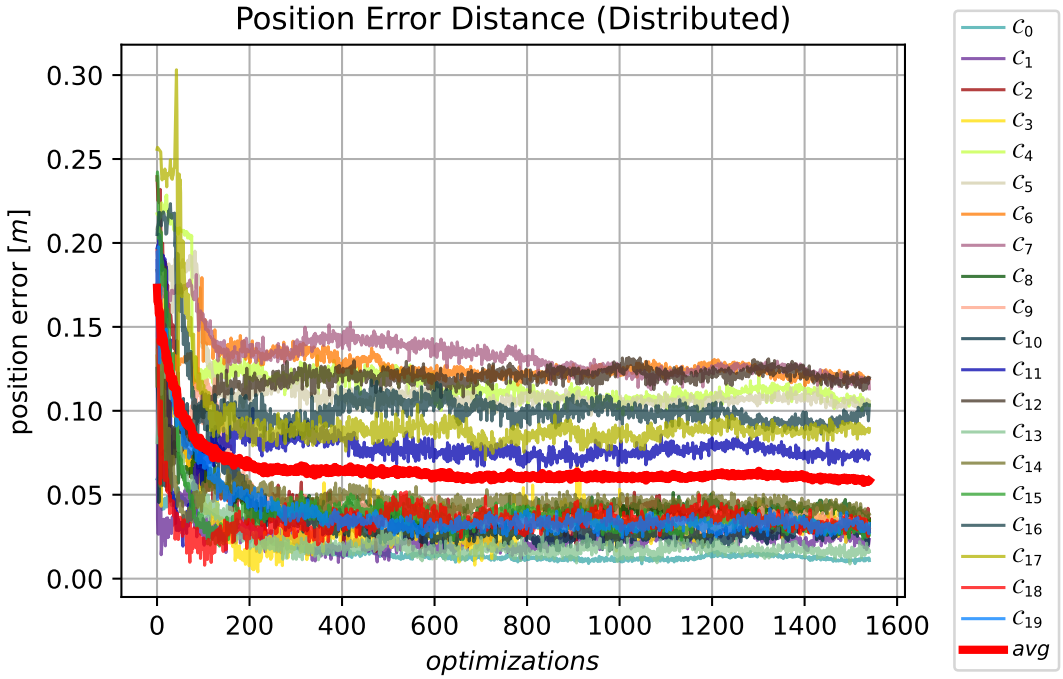

Figure 6.10: Position error toward the reference calibration for Experiment 4.

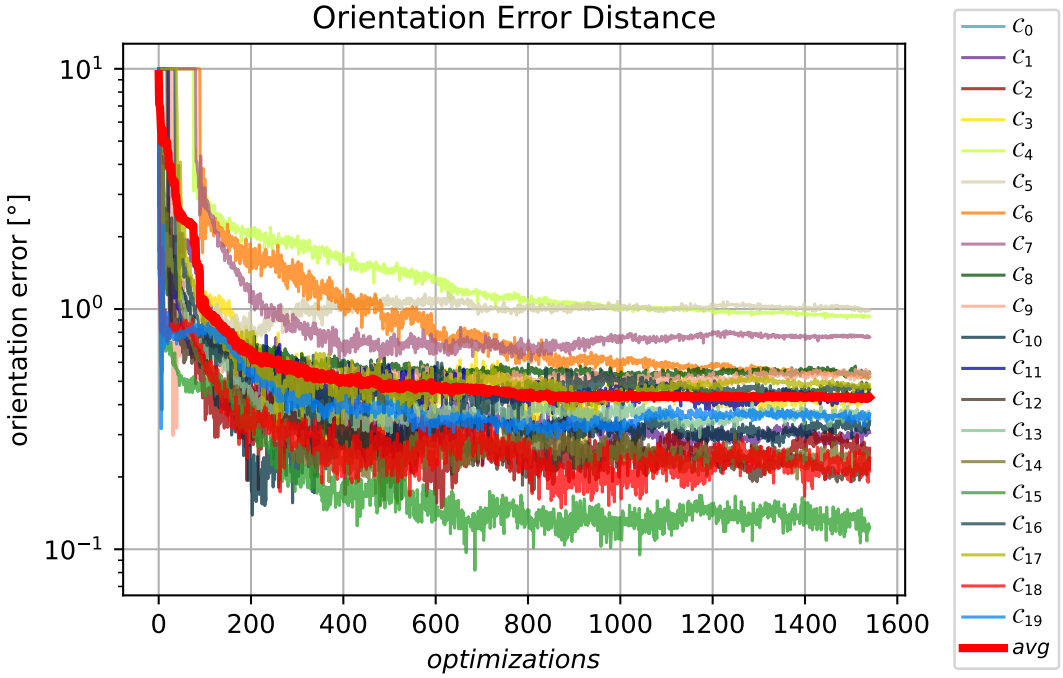

Figure 6.11: Orientation error toward the reference calibration for Experiment 4.

While we only show the results of one calibration procedure here, due to the live nature of the experiment, we did repeat the experiment about five times. Again, we notice that the same cameras repeatedly converge toward the same magnitudes of error. As the errors are relatively small, we compare the accuracy of our calibration against the reference calibration by measuring the reprojection error for an unrelated recording of multiple persons within the same scenario. The same recording was used by Bultmann *et al.* [47] to report reprojection errors. We point out that our recording does not contain any semantic feedback.

Table 6.1: Comparison of the reprojection errors per joint between our method and the reference calibration for Experiment 4.

Source	**Head**	**Hips**	**Knees**	**Ankles**	**Shoulders**	**Elbows**	**Wrists**	**Mean**
Reference	4.17px	5.40px	5.27px	6.50px	3.835px	4.88px	6.46px	**5.01px**
Method	3.80px	4.70px	4.77px	6.13px	3.534px	4.51px	6.18px	**4.60px**

The measured reprojection error per joint is shown in Tab. 6.1. We achieve lower reprojection errors in all categories, implying that our calibration is more precise than the reference calibration. Repetitions of this experiment reliably score between 4.5 and 4.9 pixels. As one would expect, the reprojection error is generally larger for faster moving joints like ankles and wrists, while it is smaller for more stable joints. It is worth noting that the measured reprojection does not exclusively originate from the provided extrinsic calibration, but also from other factors, e.g. the approach for triangulation and pose estimation.

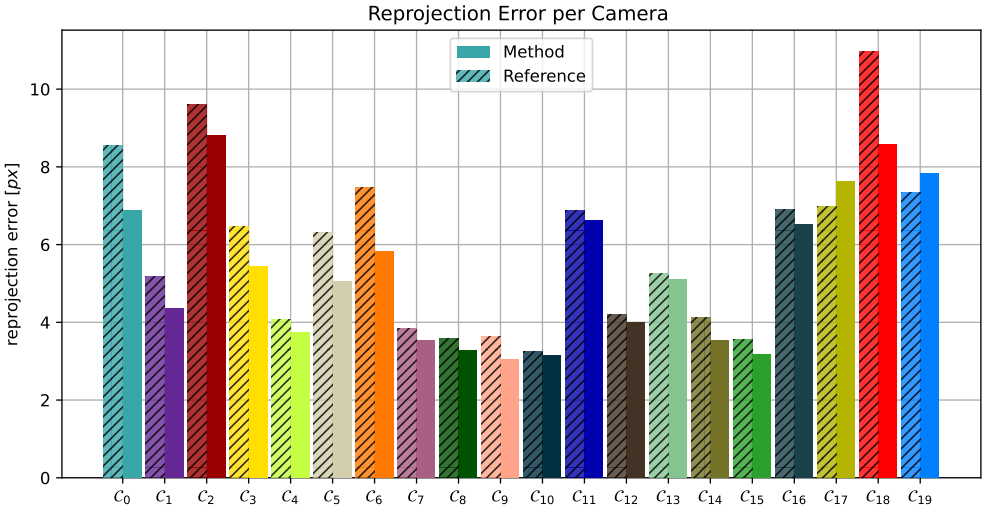

Figure 6.12: Comparison of the reprojection error per camera between our method and the reference calibration for Experiment 4.

Fig. 6.12 shows the reprojection errors per camera. We see that, with our calibration the reprojection error decreases for all cameras, except $\mathcal{C}_{17}$ and $\mathcal{C}_{19}$ for which it increases slightly.

Experiment 5

To validate our deduction from Experiment 4 which is that our method is capable of achieving higher accuracy of the extrinsic calibration compared to our reference calibration, we repeat the experiment by starting with the reference calibration as the initial estimate. All parameters remain the default ones. The input for the sensors is different in that only one person is crossing the room and two persons are sitting at their desk. We let the calibration procedure run for less than one minute.

Fig. 6.13 shows the distributed position error of this experiment. We observe that all cameras converge toward roughly the same error distance w.r.t. Fig. 6.10.

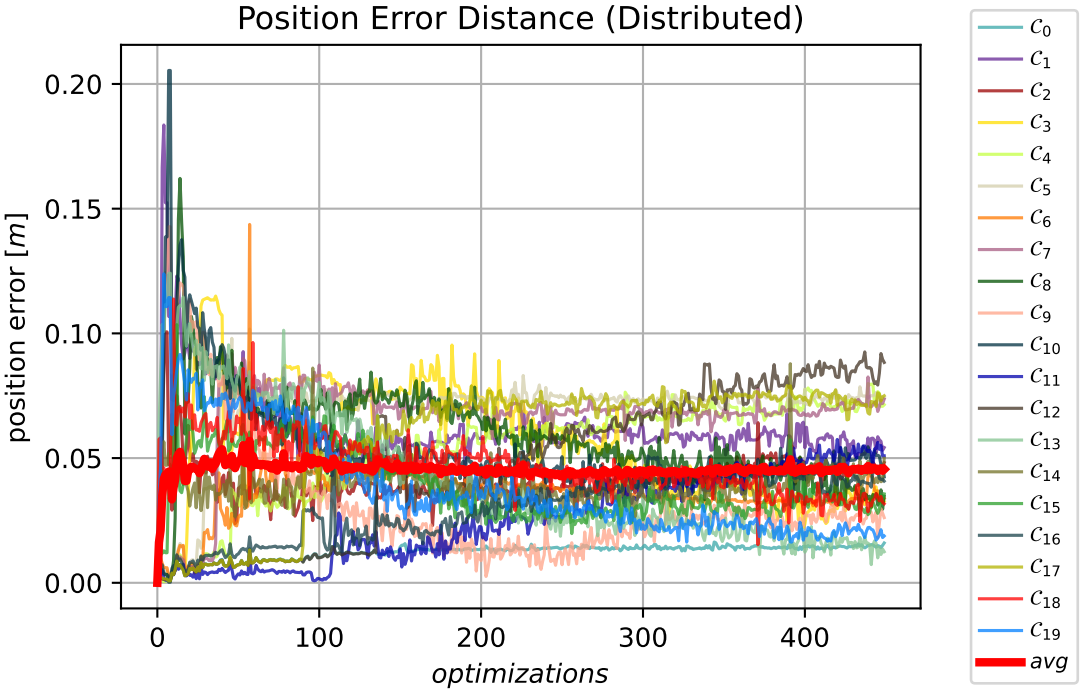

Figure 6.13: Distributed position error toward the reference calibration for Experiment 5.

As in Experiment 4, we compare the reprojection error of the obtained calibration against the reference calibration (See Tab. 6.2). Again, our method achieves lower reprojection errors (4.68 pixels on average) than the reference calibration. However, these are slightly higher than in the previous experiment (4.60 pixels on

average). Presumably, this is caused by the shorter duration of the calibration procedure.

Table 6.2: Comparison of the reprojection errors per joint between our method and the reference calibration for Experiment 5.

Source	Head	Hips	Knees	Ankles	Shoulders	Elbows	Wrists	Mean
Reference	4.17px	5.40px	5.27px	6.50px	3.835px	4.88px	6.46px	**5.01px**
Method	3.89px	4.85px	4.86px	6.17px	3.581px	4.58px	6.21px	**4.68px**

Analogously to Fig. 6.12, Fig. 6.14 shows the reprojection errors per camera for this experiment. Both figures look similar. For example, we can observe that the reprojection error for $\mathcal{C}_{17}$ and $\mathcal{C}_{19}$ slightly increases and that the largest improvement is achieved for $\mathcal{C}_{18}$.

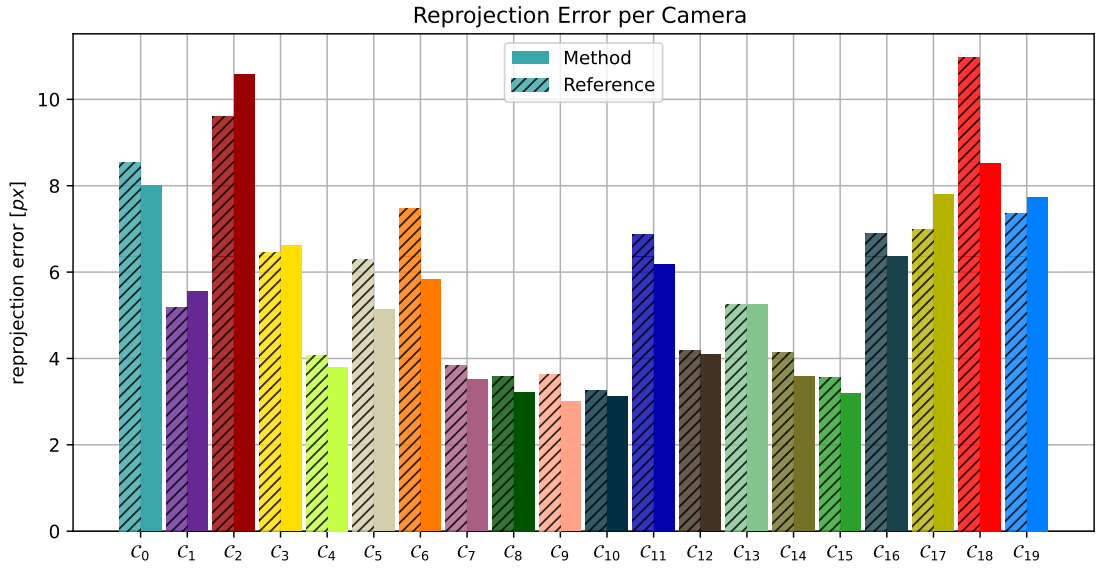

Figure 6.14: Comparison of the reprojection error per camera between our method and the reference calibration for Experiment 5.

Experiment 6

In this experiment, we evaluate the sensitivity of our method toward the intrinsic parameters which are assumed to be known for each camera. We use a recording of two persons crossing the room for two minutes within Scenario 1. We use the default parameters, but increase the optimization interval to one second in order to get more consistent results by reducing the computational load for the system and to avoid scheduling conflicts.

We capture a baseline by using the reference calibration matrix $\mathbf{K}_i$ and distortion coefficients dist_i for each camera i. Then, we omit the distortion coefficients by not undistorting the received detections. Finally, we keep omitting the distortion coefficients, but randomly permute the set of calibration matrices such that no camera uses its correct calibration matrix. Note that in Scenario 1, all cameras are of the same type.

We repeat all three setups ten times. To further improve consistency, we use the same initial estimate with an error of exactly 15cm and 15° in a random direction w.r.t. the reference calibration for all cameras $\mathcal{C}_i$ for $i > 0$. Tab. 6.3 lists the results of this experiment.

Table 6.3: Average distributed unscaled position errors for different configurations of the intrinsic calibration over multiple runs for Experiment 6.

	Baseline	**Omitted dist$_i$**	**Permuted $\mathbf{K}_i$**
Max.	0.0916m	0.1000m	0.1336m
Min.	0.0911m	0.0968m	0.1107m
Avg.	0.0914m	0.0984m	0.1201m

When comparing the baseline setup against the setup with omitted distortion coefficients, we measure a slight increase in error of 0.007m on average. Note that the maximum error of the former is below the minimum error of the latter. With the permuted calibration matrices, we observe a more drastic increase in error against both other setups. Here, we notice an increased amount of failed optimizations in the beginning of the calibration procedures. Not only does the average error increase, but the calibration results also become less consistent, which is reflected in the larger span between the minimum and maximum errors. Nonetheless, the method still works as intended while the errors increase by 0.029m on average w.r.t. to the baseline setup.

Experiment 7

In this experiment, we analyze the scaling ambiguity of the method that originates from not knowing the height of the detected persons. For this, we scale the position components of the reference calibration by a scaling factor s and use it as the initial estimate, in which we directly adopt the reference orientations. We use the same recording as in Experiment 6 with two persons crossing the room for two minutes within Scenario 1. We use default parameters. For each scaling factor, we repeat the experiment ten times and report average results. For $s = 0.5$, we scale

the minimum and maximum person height for the depth estimation accordingly, otherwise all detections would be rejected during data association.

Tab. 6.4 shows the results of this experiment. All reported errors are distributed by Umeyama's method. In the fourth column, *"Scaled"* refers to the distributed and scaled error, which equals zero in the beginning of the calibration procedure. In the fifth column, *"Factor"* refers the corresponding scaling factor used to minimize the error, which equals $\frac{1}{s}$ in the beginning of the calibration procedure.

Table 6.4: Average distributed position error for a scaled reference calibration by the factor s over multiple runs.

s	**Initial**	**Final**	**Scaled**	**Factor**
0.50	2.94687m	2.93073m	0.06251m	1.989293
0.97	0.17681m	0.17871m	0.05789m	1.028238
1.03	0.17681m	0.21166m	0.05745m	0.967419

We observe that the average distributed position error does not change significantly for an initialization with scaled reference positions. This demonstrates that the method has no knowledge about the scale of the observations and cannot resolve it. However, we see that the method is capable of maintaining some initialized scale without drifting away from it. Furthermore, when looking at the scaled error, we observe similar values as in the other experiments for Scenario 1, affirming our suspicion that the reference calibration is not optimal.

6.3. Results

This section summarizes the results of our experiments and describes further observations we encountered during the process of experimentation.

We can report that our implementation is successfully integrated into the existing system of smart edge sensors and that our method works as intended, in that the calibration error reduces over time, w.r.t. the initial estimate of the extrinsic calibration and the reference calibration. Given sufficient detection inputs and a feasible initial estimate, we have shown that the cameras converge quickly ($\lesssim 30$ seconds) to an almost optimal pose and then converge slowly ($\lesssim 5$ minutes) to their optimal pose where they finally stabilize. We deduce that resolving the correct camera positions is the central challenge within this method, while the camera orientations are easily resolved and follow along by matching the incorrect camera positions. This is not surprising, as the orientation of a camera is independent of the depth of a detection and does not require fusing corresponding detections from multiple views.

During experimentation we notice that the cameras reliably end up in roughly the same poses (for each scenario), between multiple calibration procedures with varying initial estimates and different detection inputs. At the same time, we reliably achieve average distributed position errors below 0.1m and average orientation errors below 1.0°. To assess the quality of our calibration w.r.t. the reference calibration, we calculate their reprojection errors and find that our calibrations achieve slightly lower reprojection errors. Considering that the reference calibration is obtained by an offline method utilizing a traditional calibration target with known correspondences, this might be considered surprising.

The above statements hold for feasible initial estimates, with an average position error between ~ 0.05m and ~ 0.30m toward the reference calibration, which is an accuracy that can be achieved by manual means (See App. A). With less accurate initial estimates, we observe that a single calibration run does not always obtain optimal camera poses. However, multiple (2) runs can resolve this problem, where the first runs serve to find a better initial estimate, which is then utilized in the following runs. When experimenting with more extreme initial estimates, with average position errors of several meters and orientations errors above 90°, we find that the method becomes unstable. However, through trial and error over multiple calibration runs, it is still possible to improve the calibration until an accuracy is reached that allows for normal operation. In this context, initial pose estimates in which the cameras untruly look away from the scene are the most problematic, as they can easily cause the failure of an optimizations cycle.

During most experiments, we use the default parameters presented in Sec. 5.3, which seem to be optimal for a wide range of errors, w.r.t. the initial estimate of the extrinsic calibration, and for a variety of different detection inputs, regarding the number of persons present in the scene and their movement during the calibration procedure. As most parameters interact with each other, it can be difficult to isolate and validate the effect of a single parameter. Under ideal conditions, selecting person hypotheses at random achieves similar results as selecting them according to our proposed algorithm (See Alg. 1). However, during calibration procedures where the majority of detections are generated by single cameras and the overall movement in the scene is little, we can clearly see that our algorithm prevents excessive selection of person hypotheses with a similar localization and encourages the calibration of as many cameras as possible w.r.t. the provided detections.

The adaptive noise models (See Alg. 3) used in the prior factors for the camera nodes are used conservatively with little effect. Choosing optimal values for a given experiment achieves similar results as using the adaptive noise models with generic starting values. We find that their main purpose is to simplify the application of

the method by mitigating the requirement to fine-tune them.

The temporal smoothing algorithm (See Alg. 2) does significantly improve the results in that it increases the reliability of our method. Deactivating this algorithm does in rare cases produce similar results, compared to using it, but single person hypotheses with wrong data association, large timing offsets or other suboptimal properties can easily distort an entire calibration and prohibit the overall convergence.

In Experiment 6, we show that our method is robust against using wrong intrinsic parameters and not compensating for lens distortions. However, we can measure a decrease in the accuracy of the resulting calibration.

In Experiment 7, we analyze the behavior of our method w.r.t. scaling ambiguities in the calibration. We show our method maintains the scale of the calibration that is inherent in the initial estimate. However, given an initial estimate with an improper scale, our method cannot resolve the calibration to the correct scale.

During most experiments, we notice various false detections of humanoid robots and a coat stand. It turns out that our approach for data association is effective in filtering such detections, while simultaneously correctly associating the present persons. For feasible initial estimates, the presence of multiple persons in the scene is unproblematic in that we do not notice any wrong associations. This statement holds in the case of two persons standing very close to each other in which both persons are reliably rejected.

7. Conclusion

Throughout this thesis, we introduce the topic of extrinsic camera calibration by discussing related works and establishing the relevant theoretical concepts. We then develop an online method to perform extrinsic camera calibration on a system of smart edge sensors, relying solely on the use of person keypoint detections.

The person keypoint detections get fused at a central backend by means of synchronization, filtering, and data association. Then, the obtained person hypotheses repeatedly get encoded in factor graphs to constrain the camera poses. Knowledge obtained about the camera poses by the optimization of one factor graph is forwarded and used during the construction of the next factor graph, enabling the accumulation of knowledge and provoking the convergence of all cameras toward an optimal pose. Lastly, the convergence behavior is improved by various refinement schemes based on temporal smoothing and error estimation.

Our method is designed to be robust against false or sparse sets of detections. Most of its internal parameters are adaptive w.r.t. the statistical distribution of the properties of the received person keypoint detections, eliminating the need for extensive fine-tuning of parameters. Compared to similar methods, our approach is free of many of the typical assumptions. It can cope with and exploit the detections of multiple persons being present in the scene simultaneously. Their poses toward the cameras can be arbitrary, as long as their shoulders and hips are detected. Lastly, the approach supports a wide variety of scene layouts w.r.t. curvatures of the ground surface, occlusions from objects or the sensor topology.

After implementing the proposed method and integrating it into an existing system of smart edge sensors, we evaluate the proposed method and our implementation in a series of experiments. We demonstrate that each component of our method works as intended. We compare our calibration results to a reference calibration obtained by an offline calibration method based on using traditional calibration targets. Fortunately, we can show that our calibration results are more accurate than the reference calibration by reliably achieving lower reprojection errors while obtaining similar camera poses.

Not only proves our method to be a quick and easy-to-use calibration utility, but it also achieves state-of-the-art accuracy. The general approach of our method seems to be a promising foundation for further work in this area.

7.1. Discussion & Future Work

We conclude this thesis by discussing elements that could be improved within our method and by suggesting further components that could be added to it.

We show that our method is capable of maintaining the scale that is implied by the initial estimate of the extrinsic calibration and that it can improve the calibration within this scale, but it certainly lacks a mechanism to correctly resolve the scale in case the uncertainty in the scale of the initial estimate is high. One straightforward way of achieving this is the use additional of factor nodes inside the factor graphs, which constrain the distance between the landmark nodes of each person according to some model of the human anatomy. A similar approach is used by Bultmann *et al.* [24]. However, this could easily introduce a wrong scale into the resulting calibration in case the scale of the initial estimate is mostly correct. For a calibration procedure with a single person, one would need to specify the exact dimensions of the considered person in order to resolve the scale correctly. For calibration procedures with a few persons, this approach would likely induce an incorrect scaling, as multiple persons have different dimensions and we do not use any identification mechanism that finds the correspondence between an observed person and its dimensions in some database. This scaling error might be negligible when enough persons are present during the calibration procedure, so that the assumption of some average anatomy becomes applicable. However, multiple components of our pipeline, starting with the smart edge sensors, cannot handle the required amount of persons.

Another solution to the problem of scaling ambiguity would be to equip each person in the scene with a unique fiducial marker of known dimensions. The extracted features of the fiducial markers could be encoded in the factor graphs, similar to the approach of Reinke *et al.* [11], to resolve scale. Furthermore, the markers could be used to replace the current approach for depth estimation and data association. This approach would probably require the combination with a tracking approach, in order to keep using detections of persons where the marker is currently not visible but was visible before.

Consistent 2D and 3D tracking could help improve data association. Furthermore, this would allow the exploitation of the fact that the dimensions of each person are constant during the calibration procedure by adding additional factors into the factor graphs that constrain the dimensions of a person, i.e. the distance between the respective joints, once enough measurements of the person's dimensions are taken.

Data association could be improved for inaccurate initial estimates of the extrinsic calibration by using visual descriptors as proposed by Hermans *et al.* [48]. In particular, we could compare the distances between the visual descriptors of person detections from multiple views in a similar way we compare the distances between line segments. It would be possible to use this alongside the existing geometric solution. In fact, the extraction of such visual descriptors is currently being implemented by colleagues on the sensors of type B. The computational resources available on sensor type A do not suffice for this approach.

Due to fact that a large amount of the performed computations with this method are distributed over the entire system of sensors, the approach could be easily scaled for multi-camera systems of arbitrary size. In particular, one would need to introduce a divide and conquer strategy that divides all cameras into clusters of cameras with overlapping FoVs where each cluster is running our proposed method independently. Then, the relative pose between two neighboring calibrated clusters could be obtained by first identifying one camera in each cluster with an overlapping FoV between them and finally constructing a factor graph for the respective two-camera systems. For example, this strategy could be used for calibrating cameras spread over neighboring rooms where at least one camera in each room is facing the door toward the neighboring room.

As the time our method requires to converge toward an accurate calibration is short, one might be tempted to adapt the method for non-static cameras. This would require a different architecture for the construction of the factor graphs. In particular, instead of connecting landmark nodes that originate from joint detections of varying points in time to one set of camera nodes, we would need to instantiate a unique set of camera nodes for each point in time and connect the landmark nodes accordingly. To evade the problem of under-determination for such sparse graphs, we would probably be required to build larger graphs where each set of camera nodes is connected to a sufficient amount of landmark nodes. Lastly, we would need to add additional constraints between the camera nodes that represent the instances of the same camera at different points in time. For example, this could be achieved by using inertial measurement units.

A. Manual Calibration

Figure A.1: RGB images from all cameras within Scenario 2 where each image center is highlighted by a red cross.

We try to find the extrinsic calibration of the cameras within Scenario 2 (See Sec. 6.1) by using a floor plan of the building, an RGB image of each camera and a tape measure. First, we visually estimate the camera positions and draw them onto the floor plan. Then, we find the center points within the RGB images of all cameras (See Fig. A.1), which serve as an estimate of the principle points. We try to find the location where the principle axis of each camera intersects with the ground plane and draw them into the map. We conclude the preparation of the floor plan by choosing a location on the ground plane, which serves as an intermediate origin of the coordinate system in which we measure all camera poses. Here, the z-axis points toward the ceiling, while the x- and y-axes point to the right and top within the map (See Fig. A.2).

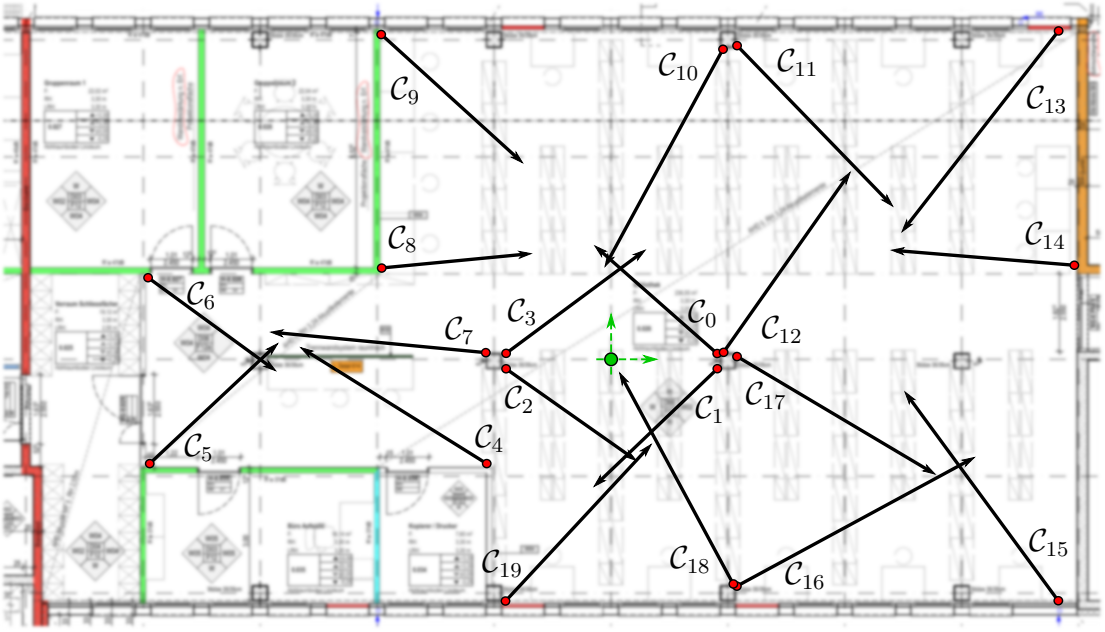

Figure A.2: Visual estimation of the camera poses for Scenario 2 within the floor plan of the building. Arrows indicate where the principle axes intersect the ground plane. An arbitrary pose aligned with the room geometry serves as origin of the coordinate system (green).

Inside the room, we measure the height of all cameras as well as the distance between the two inner pillars of the room, which serves as a ratio to convert between pixels and meters. Now, we can simply measure the x- and y-components of the position of each camera on the map by converting the pixel-distances to meters and use our height measurements for the z-components.

Similarly, we can measure the yaw- and pitch-components of the rotation of each camera. For the roll-angles, we simply assume 0° for all cameras that are mounted as intended and 180° for all cameras that are mounted upside-down ($\mathcal{C}_0$ to $\mathcal{C}_3$). Finally, we can transform the position and orientation components of all cameras into the coordinate system of camera $\mathcal{C}_0$ by applying the inverse of the obtained transformation between the room coordinate system and $\mathcal{C}_0$ to all cameras $\mathcal{C}_0$ to $\mathcal{C}_{19}$.

The resulting calibration of this experiment has an average error of 0.1737m and 11.14° for cameras $\mathcal{C}_1$ to $\mathcal{C}_{19}$ toward our reference calibration (See Sec. 6.1).

B. Reference Calibration

Table B.1: Reference extrinsic calibration for Scenario 1 with 16 cameras.

Camera	**Position**	**Orientation**
$\mathcal{C}_0$	$(+\ 0.00\text{m}, +\ 0.00\text{m}, +\ 0.00\text{m})^\mathsf{T}$	$(+\ 0.00°, +\ 0.00°, +\ 0.00°)^\mathsf{T}$
$\mathcal{C}_1$	$(-\ 2.63\text{m}, -\ 2.92\text{m}, +\ 3.90\text{m})^\mathsf{T}$	$(+\ 92.95°, +\ 38.87°, +142.93°)^\mathsf{T}$
$\mathcal{C}_2$	$(+\ 0.30\text{m}, -\ 3.67\text{m}, +\ 5.34\text{m})^\mathsf{T}$	$(+109.04°, +\ 17.10°, +177.49°)^\mathsf{T}$
$\mathcal{C}_3$	$(+\ 3.08\text{m}, -\ 0.74\text{m}, +\ 1.61\text{m})^\mathsf{T}$	$(+\ 14.16°, -\ 46.99°, -\ 31.41°)^\mathsf{T}$
$\mathcal{C}_4$	$(+\ 6.11\text{m}, +\ 0.88\text{m}, +\ 0.29\text{m})^\mathsf{T}$	$(+\ 29.51°, -\ 47.01°, -\ 39.68°)^\mathsf{T}$
$\mathcal{C}_5$	$(+\ 6.31\text{m}, +\ 0.87\text{m}, +\ 0.33\text{m})^\mathsf{T}$	$(+\ 13.87°, +\ 21.69°, +\ 24.80°)^\mathsf{T}$
$\mathcal{C}_6$	$(+\ 3.67\text{m}, -\ 2.05\text{m}, +\ 4.00\text{m})^\mathsf{T}$	$(+\ 65.48°, +\ 55.72°, +\ 97.00°)^\mathsf{T}$
$\mathcal{C}_7$	$(+\ 3.64\text{m}, -\ 1.87\text{m}, +\ 3.72\text{m})^\mathsf{T}$	$(+115.78°, -\ 16.18°, -156.50°)^\mathsf{T}$
$\mathcal{C}_8$	$(+\ 9.52\text{m}, -\ 5.04\text{m}, +\ 8.94\text{m})^\mathsf{T}$	$(+\ 93.20°, -\ 47.12°, -126.69°)^\mathsf{T}$
$\mathcal{C}_9$	$(+\ 13.02\text{m}, -\ 0.62\text{m}, +\ 3.38\text{m})^\mathsf{T}$	$(+\ 37.79°, -\ 52.87°, -\ 49.14°)^\mathsf{T}$
$\mathcal{C}_{10}$	$(+\ 6.32\text{m}, -\ 7.81\text{m}, +\ 12.73\text{m})^\mathsf{T}$	$(+119.00°, -\ 4.57°, -169.25°)^\mathsf{T}$
$\mathcal{C}_{11}$	$(+\ 3.34\text{m}, -\ 2.08\text{m}, +\ 4.00\text{m})^\mathsf{T}$	$(+\ 4.78°, +\ 0.16°, +\ 2.01°)^\mathsf{T}$
$\mathcal{C}_{12}$	$(-\ 4.68\text{m}, -\ 1.30\text{m}, +\ 1.80\text{m})^\mathsf{T}$	$(+\ 18.76°, +\ 37.00°, +\ 39.43°)^\mathsf{T}$
$\mathcal{C}_{13}$	$(+\ 0.05\text{m}, -\ 5.91\text{m}, +\ 9.06\text{m})^\mathsf{T}$	$(+\ 82.71°, +\ 52.01°, +120.76°)^\mathsf{T}$
$\mathcal{C}_{14}$	$(-\ 0.25\text{m}, -\ 5.83\text{m}, +\ 8.82\text{m})^\mathsf{T}$	$(+115.17°, +\ 2.35°, -178.45°)^\mathsf{T}$
$\mathcal{C}_{15}$	$(-\ 7.31\text{m}, -\ 4.27\text{m}, +\ 5.45\text{m})^\mathsf{T}$	$(+\ 74.98°, +\ 55.33°, +112.67°)^\mathsf{T}$

Table B.2: Reference extrinsic calibration for Scenario 2 with 20 cameras.

Camera	**Position**	**Orientation**
$\mathcal{C}_0$	$(+\ 0.00\text{m}, +\ 0.00\text{m}, +\ 0.00\text{m})^\mathsf{T}$	$(+\ 0.00°, +\ 0.00°, +\ 0.00°)^\mathsf{T}$
$\mathcal{C}_1$	$(+\ 0.28\text{m}, -\ 0.12\text{m}, -\ 0.23\text{m})^\mathsf{T}$	$(-\ 49.64°, +\ 53.35°, -\ 78.31°)^\mathsf{T}$
$\mathcal{C}_2$	$(+\ 3.51\text{m}, +\ 1.91\text{m}, +\ 2.93\text{m})^\mathsf{T}$	$(-115.31°, -\ 6.65°, -178.24°)^\mathsf{T}$
$\mathcal{C}_3$	$(+\ 3.22\text{m}, +\ 2.06\text{m}, +\ 3.16\text{m})^\mathsf{T}$	$(-\ 75.23°, -\ 55.08°, +111.34°)^\mathsf{T}$
$\mathcal{C}_4$	$(+\ 5.52\text{m}, +\ 1.38\text{m}, +\ 1.96\text{m})^\mathsf{T}$	$(+\ 4.80°, -\ 4.11°, +172.46°)^\mathsf{T}$
$\mathcal{C}_5$	$(+\ 10.67\text{m}, +\ 4.62\text{m}, +\ 6.99\text{m})^\mathsf{T}$	$(+\ 66.52°, +\ 55.75°, -\ 81.48°)^\mathsf{T}$
$\mathcal{C}_6$	$(+\ 7.42\text{m}, +\ 6.20\text{m}, +\ 9.47\text{m})^\mathsf{T}$	$(+113.51°, +\ 8.06°, -\ 0.51°)^\mathsf{T}$
$\mathcal{C}_7$	$(+\ 3.57\text{m}, +\ 2.28\text{m}, +\ 3.46\text{m})^\mathsf{T}$	$(+\ 8.31°, -\ 28.12°, +161.49°)^\mathsf{T}$
$\mathcal{C}_8$	$(+\ 3.57\text{m}, +\ 3.93\text{m}, +\ 6.14\text{m})^\mathsf{T}$	$(+106.14°, +\ 39.38°, -\ 32.98°)^\mathsf{T}$
$\mathcal{C}_9$	$(-\ 0.54\text{m}, +\ 5.89\text{m}, +\ 9.07\text{m})^\mathsf{T}$	$(+118.11°, -\ 0.12°, +\ 2.06°)^\mathsf{T}$
$\mathcal{C}_{10}$	$(-\ 5.55\text{m}, +\ 2.42\text{m}, +\ 3.84\text{m})^\mathsf{T}$	$(+\ 78.52°, -\ 57.34°, +\ 67.82°)^\mathsf{T}$
$\mathcal{C}_{11}$	$(-\ 5.80\text{m}, +\ 2.24\text{m}, +\ 3.67\text{m})^\mathsf{T}$	$(+119.82°, -\ 5.84°, +\ 3.54°)^\mathsf{T}$
$\mathcal{C}_{12}$	$(-\ 0.19\text{m}, -\ 0.06\text{m}, -\ 0.16\text{m})^\mathsf{T}$	$(+\ 42.49°, +\ 59.35°, -109.62°)^\mathsf{T}$
$\mathcal{C}_{13}$	$(-\ 10.97\text{m}, -\ 0.73\text{m}, -\ 0.84\text{m})^\mathsf{T}$	$(+\ 65.45°, -\ 55.96°, +\ 88.44°)^\mathsf{T}$
$\mathcal{C}_{14}$	$(-\ 7.03\text{m}, -\ 2.70\text{m}, -\ 4.37\text{m})^\mathsf{T}$	$(+\ 6.96°, -\ 30.15°, +160.10°)^\mathsf{T}$
$\mathcal{C}_{15}$	$(-\ 0.77\text{m}, -\ 5.34\text{m}, -\ 8.38\text{m})^\mathsf{T}$	$(+\ 8.99°, +\ 8.69°, -170.14°)^\mathsf{T}$
$\mathcal{C}_{16}$	$(+\ 3.85\text{m}, -\ 2.13\text{m}, -\ 3.39\text{m})^\mathsf{T}$	$(+\ 92.41°, +\ 50.63°, -\ 56.31°)^\mathsf{T}$
$\mathcal{C}_{17}$	$(-\ 0.24\text{m}, -\ 0.23\text{m}, -\ 0.43\text{m})^\mathsf{T}$	$(+122.16°, +\ 6.42°, -\ 2.08°)^\mathsf{T}$
$\mathcal{C}_{18}$	$(+\ 3.90\text{m}, -\ 2.02\text{m}, -\ 3.22\text{m})^\mathsf{T}$	$(+\ 8.51°, +\ 16.67°, -166.89°)^\mathsf{T}$
$\mathcal{C}_{19}$	$(+\ 7.63\text{m}, +\ 0.08\text{m}, -\ 0.19\text{m})^\mathsf{T}$	$(+\ 59.85°, +\ 56.54°, -\ 91.23°)^\mathsf{T}$

List of Figures

List of Tables

Bibliography

[1] Jérôme Maye, Paul Furgale, and Roland Siegwart. "Self-supervised calibration for robotic systems". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2013, pp. 473–480. DOI: 10.1109/IVS.2013.6629513.

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

[3] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4304–4311. DOI: 10.1109/ICRA.2016.7487628.

[4] Edwin Olson. "AprilTag: A robust and flexible visual fiducial system". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 3400–3407. DOI: 10.1109/ICRA.2011.5979561.

[5] Przemysław Komorowski Jacekand Rokita. "Extrinsic camera calibration method and its performance evaluation". In: *Computer Vision and Graphics*. 2012, pp. 129–138. DOI: 10.1007/978-3-642-33564-8_16.

[6] D.G. Lowe. "Object recognition from local scale-invariant features". In: *IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[7] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24.6 (1981), pp. 381–395. DOI: 10.1145/358669.358692.

[8] Romil Bhardwaj, Gopi Krishna Tummala, Ganesan Ramalingam, Ramachandran Ramjee, and Prasun Sinha. "AutoCalib: Automatic traffic camera calibration at scale". In: *ACM Transactions on Sensor Networks (TOSN)* 14.3-4 (2018), pp. 1–27. DOI: https://doi.org/10.1145/3199667.

[9] Junzhi Guan, Francis Deboeverie, Maarten Slembrouck, Dirk Van Haerenborgh, Dimitri Van Cauwelaert, Peter Veelaert, and Wilfried Philips. "Extrinsic calibration of camera networks based on pedestrians". In: *Sensors* 16.5 (2016). DOI: 10.3390/s16050654.

[10] Anh Minh Truong, Wilfried Philips, Junzhi Guan, Nikos Deligiannis, and Lusine Abrahamyan. "Automatic extrinsic calibration of camera networks based on pedestrians". In: *IEEE International Conference on Distributed Smart Cameras (ICDSC)*. Association for Computing Machinery, 2019. DOI: 10.1145/3349801.3349802.

[11] Andrzej Reinke, Marco Camurri, and Claudio Semini. "A factor graph approach to multi-camera extrinsic calibration on legged robots". In: *IEEE International Conference on Robotic Computing (IRC)*. 2019, pp. 391–394. DOI: 10.1109/IRC.2019.00071.

[12] Frank Dellaert and Michael Kaess. "Factor graphs for robot perception". In: *Foundations and Trends in Robotics (FNT)* 6.1-2 (2017). http://dx.doi.org/10.1561/2300000043, pp. 1–139.

[13] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. USA, 2005, pp. 886–893. DOI: 10.1109/CVPR.2005.177.

[14] M.A. Fischler and R.A. Elschlager. "The representation and matching of pictorial structures". In: *IEEE Transactions on Computers* C-22.1 (1973), pp. 67–92. DOI: 10.1109/T-C.1973.223602.

[15] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Pictorial structures for object recognition". In: *International Journal of Computer Vision (IJCV)* 61.1 (2005), pp. 55–79. DOI: 10.1023/B:VISI.0000042934.15159.49. URL: https://doi.org/10.1023/B:VISI.0000042934.15159.49.

[16] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. "OpenPose: Realtime multi-person 2D pose estimation using part affinity fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 43.1 (2021), pp. 172–186. DOI: 10.1109/TPAMI.2019.2929257.

[17] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. "RMPE: Regional multi-person pose estimation". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2353–2362. DOI: 10.1109/ICCV.2017.256.

[18] Jiefeng Li, Can Wang, Hao Zhu, Yihuan Mao, Hao-Shu Fang, and Cewu Lu. "CrowdPose: Efficient crowded scenes pose estimation and a new benchmark". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10855–10864. DOI: 10.1109/CVPR.2019.01112.

[19] Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and Cewu Lu. "Pose Flow: Efficient online pose tracking". In: *British Machine Vision Conference (BMVC)*. 2018. URL: http://www.bmva.org/bmvc/2018/index.html.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. *Microsoft COCO: Common objects in context*. 2015. arXiv: 1405.0312.

[21] Mykhaylo, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. "2D human pose estimation: New benchmark and state of the art analysis". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014. DOI: 10.1109/CVPR.2014.471.

[22] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic. "3D pictorial structures revisited: Multiple human pose estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 38.10 (2016), pp. 1929–1942. DOI: 10.1109/TPAMI.2015.2509986.

[23] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 36.7 (2014), pp. 1325–1339. DOI: 10.1109/TPAMI.2013.248.

[24] Simon Bultmann and Sven Behnke. "Real-time multi-view 3D humanion using semantic feedback to smart edge sensors". In: *Robotics: Science and Systems XVII*. 2021. DOI: 10.15607/rss.2021.xvii.040.

[25] Julian Tanke and Juergen Gall. "Iterative greedy matching for 3D human pose tracking from multiple views". In: *German conference on pattern recognition*. 2019. DOI: 10.1007/978-3-030-33676-9_38.

[26] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. 2nd ed. USA: Cambridge University Press, 2003. ISBN: 0521540518.

[27] Zhongwei Tang, Rafael Grompone von Gioi, Pascal Monasse, and Jean-Michel Morel. "A precision analysis of camera distortion models". In: *IEEE Transactions on Image Processing* 26.6 (2017), pp. 2694–2704. DOI: 10.1109/TIP.2017.2686001.

[28] Daphne Koller and Nir Friedman. *Probabilistic graphical models: Principles and techniques*. Jan. 2009. ISBN: 978-0-262-01319-2.

[29] Brendan J. Frey. "Extending factor graphs so as to unify directed and undirected graphical models". In: *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI)*. UAI'03. Acapulco, Mexico: Morgan Kaufmann Publishers Inc., 2002, pp. 257–264. DOI: 10.48550/ARXIV.1212.2486.

[30] Frank Dellaert. *Factor graphs and GTSAM: A hands-on introduction*. Tech. rep. GT-RIM-CP&R-2012-002. GT RIM, 2012. URL: https://research.cc.gatech.edu/borg/sites/edu.borg/files/downloads/gtsam.pdf.

[31] Yan-Bin Jia. "Quaternions and rotations". In: *Com S* 477.577 (2013).

[32] Jack B Kuipers. "Quaternions and rotation sequences". In: *Proceedings of the International Conference on Geometry, Integrability and Quantization*. 2000, pp. 127–143. DOI: doi:10.7546/giq-1-2000-127-143.

[33] Jack B Kuipers. *Quaternions and rotation sequences: A primer with applications to orbits, aerospace, and virtual reality*. Princeton university press, 1999. ISBN: 9780691102986.

[34] James Diebel. "Representing attitude: Euler angles, unit quaternions, and rotation vectors". In: *Matrix* 58.15-16 (2006), pp. 1–35.

[35] Du Q Huynh. "Metrics for 3D rotations: Comparison and analysis". In: *Journal of Mathematical Imaging and Vision* 35.2 (2009), pp. 155–164.

[36] S. Umeyama. "Least-squares estimation of transformation parameters between two point patterns". In: *IEEE Transactions on Pattern Analysis and Machine intelligence (PAMI)* 13.04 (1991), pp. 376–380. DOI: 10.1109/34.88573.

[37] Andrew Howard, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, Yukun Zhu, Ruoming Pang, Hartwig Adam, and Quoc Le. "Searching for MobileNetV3". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 1314–1324. DOI: 10.1109/ICCV.2019.00140.

[38] Josh Faust. *Approximate time synchronizer*. Accessed: 2022-02-17. 2010. URL: http://wiki.ros.org/message_filters/ApproximateTime.

[39] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Ng. "ROS: An open-source robot operating system". In: vol. 3. 2009, p. 5.

[40] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[41] Stefan Wirtz and Dietrich Paulus. "Evaluation of established line segment distance functions". In: *Pattern Recognition and Image Analysis* 26 (2016), pp. 354–359. DOI: 10.1134/S1054661816020267.

[42] Landis Markley, Yang Cheng, John Crassidis, and Yaakov Oshman. "Averaging quaternions". In: *Journal of Guidance, Control, and Dynamics* 30 (2007), pp. 1193–1196. DOI: 10.2514/1.28949.

[43] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[44] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[45] Google. *EdgeTPU dev board*. Accessed: 2022-03-25. 2020. URL: https://coral.ai/docs/dev-board/datasheet.

[46] NVIDIA. *Nvidia Jetson Xavier NX Developer Kit*. Accessed: 2022-03-25. 2020. URL: https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit.

[47] Simon Bultmann and Sven Behnke. "3D semantic scene perception using distributed smart edge sensors". In: *Accepted for IEEE International Conference on Intelligent Autonomous Systems (IAS)*. 2022.

[48] Alexander Hermans, Lucas Beyer, and Bastian Leibe. "In defense of the triplet loss for person re-identification". In: (2017). URL: http://arxiv.org/abs/1703.07737.