ZHAW Winterthur
Institute of Applied Mathematics and Physics

Master Thesis

MC3D-TRECSIM: Multi-Camera 3D Trajectory Reconstruction for Enhanced Crowd Safety and Infrastructure Monitoring

Benjamin Heuberger

September 9, 2024

Supervisor
Dr Thomas Oskar Weinmann

Zusammenfassung

In dieser Arbeit wird die Entwicklung und Evaluierung des Systems ‘Multi-Camera 3D Trajectory Reconstruction for Enhanced Crowd Safety and Infrastructure Monitoring’ (MC3D-TRECSIM) vorgestellt, das die Sicherheit von Menschenmengen und die Überwachung von Infrastrukturen verbessern soll. Das System zielt darauf ab, Überwachungsaufgaben in überfüllten Umgebungen und automatisierten Infrastrukturen, wie z. B. öffentlichen Verkehrsmitteln, zu verbessern. Es nutzt bestehende Algorithmen zur zweidimensionalen (2D) Pose-Estimation, um räumliche Orientierungsdaten aus mehreren Kameraaufnahmen zu extrahieren, die dann integriert werden, um eine umfassende 3D-Darstellung von Personen, einschliesslich ihrer Posen und Positionen, zu rekonstruieren. Das Hauptziel besteht darin, ein robustes maschinelles Lernsystem zu schaffen, das in der Lage ist, die Bewegungsabläufe mehrerer Personen genau zu verfolgen und vorherzusagen, indem Daten aus verschiedenen Kameraquellen zusammengeführt werden.

Die Forschung konzentriert sich auf mehrere Schlüsselbereiche: Untersuchung von 2D-Pose-Estimation-Algorithmen, die für Multikamera-Setups geeignet sind, Entwicklung effektiver 2D-zu-3D-Datenfusionsmethoden und Entwurf von Algorithmen für die genaue Verfolgung und Vorhersage von Trajektorien. Die Robustheit des Systems wird durch die Analyse mehrerer Videosequenzen von Personen getestet, darunter auch Grenzfälle.

Die Ergebnisse zeigen, dass das System 3D-Trajektorien mit hoher Genauigkeit und Robustheit rekonstruieren kann, selbst in dynamischen Umgebungen. Das System erzielte präzise Schätzungen der Gliedmassenlänge und blieb in verschiedenen Testszenarien stabil. Allerdings ist das System noch nicht in der Lage, ungewöhnliche Körperausrichtungen und sich überschneidende Pfade von Personen zu verarbeiten. Die Leistungsanalyse zeigt, dass das System effizient arbeitet und in der Lage ist, Echtzeitdaten zu verarbeiten, wodurch es sich für praktische Überwachungs- und Infrastrukturüberwachungsanwendungen eignet.

Nach der Fertigstellung bietet das MC3D-TRECSIM-System eine funktionale Softwarelösung mit ausführlicher Dokumentation und Demonstrationsmaterial, um die Zugänglichkeit und Benutzerfreundlichkeit zu gewährleisten. Diese Arbeit trägt dazu bei, das Gebiet der 3D-Pose-Estimation voranzubringen und liefert wertvolle Erkenntnisse und praktische Anwendungen für die Überwachung, die Sicherheit und das Monitoring von Infrastrukturen in komplexen Umgebungen, wodurch die Sicherheit von überfüllten Räumen und automatisierten Infrastrukturen verbessert wird.

Der für die Analyse entwickelte und verwendete Quellcode sowie ein Beispielvideo sind in einem GitHub-Repository unter `https://github.com/Rothen/mc3d-trecsim` offen zugänglich.

Abstract

This thesis presents the development and evaluation of a multi-camera three-dimensional (3D) trajectory reconstruction system, Multi-Camera 3D Trajectory Reconstruction for Enhanced Crowd Safety and Infrastructure Monitoring (MC3D-TRECSIM), designed to enhance crowd safety and infrastructure monitoring. The system addresses the need to improve surveillance tasks in crowded environments and automated infrastructures, such as public transportation. It leverages existing two-dimensional (2D) pose estimation algorithms to extract spatial orientation data from multiple camera feeds, which are then integrated to reconstruct a comprehensive 3D representation of individuals, including their poses and positions. The core objective is to create a robust machine-learning framework capable of accurately tracking and predicting the trajectories of multiple individuals by fusing data from various camera sources.

The research focuses on several key areas: investigating 2D pose estimation algorithms suitable for multi-camera setups, developing effective 2D-to-3D data fusion methods, and designing algorithms for accurate trajectory tracking and prediction. The system's robustness is tested through analyzing multiple video sequences of people including edge case scenarios.

The results demonstrate that the system can effectively reconstruct 3D trajectories with high accuracy and robustness, even in dynamic environments. The system achieved precise limb length estimations and maintained stability across various test scenarios. However, the system is not yet able to handle unusual body orientations and overlapping paths of individuals. The performance analysis indicates that the system operates efficiently and is capable of real-time processing, making it suitable for practical surveillance and infrastructure monitoring applications.

Upon completion, the MC3D-TRECSIM system offers a functional software solution with detailed documentation and demonstration materials to ensure accessibility and ease of use. This thesis contributes to advancing the field of 3D pose estimation, providing valuable insights and practical applications for surveillance, security, and infrastructure monitoring in complex environments, thereby enhancing the safety and security of crowded spaces and automated infrastructures.

The source code developed and used for the analysis as well as an example video are openly available in a GitHub repository at `https://github.com/Rothen/mc3d-trecsim`.

Preface

The journey through the Master Program in Data Science has been one of discovery and deepening interest, particularly in the fields of computer vision and machine learning. These disciplines have fascinated me for their potential to transform vast amounts of visual data into meaningful insights, paving the way for innovative applications across various domains.

My fascination with computer vision was further ignited as I explored the integration of machine learning techniques, specifically neural networks, to enhance and expand its capabilities. The idea of combining these two powerful technologies became a central theme of my academic pursuits, inspiring me to delve deeper into the possibilities they offer when applied together.

The inspiration for my thesis topic came from a conversation with a friend and colleague, Martin Rejzek. During our discussions, Martin suggested an intriguing idea: tracking people exiting a ski lift in 3D using computer vision techniques. This concept resonated with me as it presented an opportunity to apply my academic knowledge to a real-world problem, offering practical solutions that could be directly implemented in a project at work.

This thesis represents the culmination of my studies and interests, as well as the application of neural networks in a challenging and practical context. It not only demonstrates the theoretical underpinnings of computer vision and machine learning but also their potential to solve complex problems in everyday scenarios. I hope that the findings and methodologies developed here will contribute to further advancements in the field and offer valuable insights for future projects.

Acknowledgements

I would like to express my deepest gratitude to all those who have supported me throughout this project, whether through emotional encouragement or practical assistance. I am particularly thankful to the following individuals:

To my supervisor, Dr Thomas Oskar Weinmann, for his invaluable guidance, unwavering support, and patience throughout the course of this project.

To my husband, Nemo Heuberger, for his constant encouragement and understanding during this demanding time.

To my colleagues and friends, Silvan Fluri, Carmen Frischknecht-Gruber, Dr Christian Hilbes, Martin Rejzek, David Stocker, and Preami Uthayavathanan, for believing in me, their willingness to be my test subjects, and for Martin's idea of tracking people exiting a ski lift, which became the foundation of this work.

To my friends, Duli, Guni, Karin, Max, Phipsi, Sandy, Sui, and Tami for offering enriching conversations, emotional support, and valuable feedback.

Finally, to my cats, Alma and Nancy, for their companionship and comfort during this journey.

Contents

1 Introduction

The demand for accurate and robust 3D pose estimation has significantly increased across various industries, particularly in applications such as surveillance, security, and infrastructure monitoring. These fields require precise motion-tracking capabilities to ensure safety, efficiency, and reliability in their operations. However, the current systems available for 3D pose estimation are often limited in their capabilities, creating barriers to their widespread adoption in real-world scenarios.

This thesis proposes developing a cost-effective, real-time 3D pose estimation system that can track multiple subjects without markers. By integrating multi-camera data and utilizing advanced machine learning techniques, this research aims to enhance the accuracy, reliability, and accessibility of 3D pose estimation, contributing to its broader adoption in critical industries.

1.1 Motivation

One of the primary challenges with existing 3D pose estimation systems is their dependence on markers, proprietary technology, and high operational costs. These factors limit their flexibility and accessibility, particularly for smaller organizations or projects with limited budgets. To address these limitations, there is a need for more accurate, robust, and cost-effective solutions that can be easily integrated into various applications without the need for specialized equipment or high-cost setups.

Three well-known systems in the realm of 3D motion capture are Vicon [1], Qualisys [2], and Anipose [3]. Vicon is recognized for its award-winning motion capture technology, widely used in fields such as biomechanics, sports science, and virtual reality. Qualisys offers high-precision motion capture systems known for their flexibility and accuracy. However, both Vicon and Qualisys systems are proprietary and closed-source, which restricts customization and integration with other technologies. Additionally, Vicon systems require the use of markers, which can be cumbersome in environments where unobtrusive monitoring is needed. While Qualisys has developed markerless solutions, they still rely on specialized and costly cameras, making them less accessible to users with limited resources.

Anipose, a toolkit designed for robust markerless 3D pose estimation in animals, offers an alternative approach. It provides a more accessible solution by eliminating the need for markers. However, Anipose has its own limitations, such as the inability to process in real-time and its restriction to tracking only one animal or person at a time. These drawbacks limit its applicability in scenarios where real-time tracking of multiple subjects is required.

In light of these challenges, there is a clear motivation to develop a better approach to 3D pose estimation that is both cost-effective and capable of operating in real-time across multiple subjects without the need for markers. Such a solution would significantly enhance the applicability of 3D motion capture technology, making it more accessible and practical for a wider range of

industries and applications. This thesis aims to contribute to the development of such a solution, addressing the current limitations and paving the way for broader adoption of 3D pose estimation in real-world scenarios.

1.2 Objectives

As described in the official task proposal in the appendix (see appendix A Master Thesis Proposal), this thesis aims to create and assess a robust machine learning system capable of simultaneously extracting 3D pose trajectories for any number of individuals by integrating data from multiple cameras. The overarching goal is to enhance the accuracy and reliability of 3D pose estimation in complex environments, particularly those involving multiple subjects and viewpoints.

To achieve the goals, we explored and implemented advanced methods for fusing 2D pose data from multiple cameras. This process involves the technical integration of data streams and the development of sophisticated algorithms capable of translating 2D information into accurate 3D reconstructions. This step is crucial, as it addresses the inherent challenges of depth perception and spatial orientation in multi-camera environments.

Furthermore, the thesis will design and refine algorithms for tracking and predicting the trajectories of individuals as they move within the monitored space. To ensure the system's robustness, we will also determine its limitations by identifying and testing edge cases. The testing includes creating scenarios that push the system to its boundaries, thereby revealing potential weaknesses and areas for improvement. The thesis aims to enhance the system's overall performance and reliability by systematically evaluating these edge cases.

Overall, this thesis seeks to advance the field of 3D pose estimation through innovative machine learning techniques, providing valuable insights and practical solutions for multi-camera systems.

2 Theoretical Framework

A multitude of different theoretical concepts and methods are required to tackle the objectives defined. This section provides an overview of the most important ones.

2.1 Projective Geometry

In this section, an overview of the basic concepts of projective geometry is provided, which form the foundation for understanding the mathematical principles underlying 3D pose estimation and camera calibration. Projective geometry is a branch of mathematics that deals with the properties of geometric figures under projection, a process that maps points from one space to another. This transformation is essential for capturing the relationship between 3D objects and their 2D representations in images, enabling the reconstruction of spatial information from visual data.

2.1.1 Homogeneous Coordinates

As described by OpenCV [4] and Anwar [5], homogeneous coordinates are a coordinate system widely utilized in projective geometry, which extends traditional Euclidean coordinates by incorporating a concept known as the point at infinity. This extension facilitates the representation of points at infinity and enables affine transformations to be expressed through matrix multiplications. In this system, a point in 2D space is represented as a 3D vector $(x, y, 1)^T$, while a point in 3D space is represented as a 4D vector $(x, y, z, 1)^T$.

The homogeneous vector $\overline{P}$ can be obtained by appending a 1 along an n-dimensional cartesian vector P; for example, a 3D cartesian vector can be mapped $P \to \overline{P}$ as follows:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \to \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{1}$$

For the inverse mapping $\overline{P} \to P$, all elements of the homogeneous vector get divided by its last element; for example, a 3D homogeneous vector gets its 2D cartesian counterpart as follows:

$$\begin{bmatrix} X \\ Y \\ W \end{bmatrix} \to \begin{bmatrix} X/W \\ Y/W \end{bmatrix}, \tag{2}$$

if $W \neq 0$.

Due to this mapping, all multiples $k\overline{P}$, for $k \neq 0$, of a homogeneous point represent the same euclidean point P. An intuitive understanding of this property is that all multiples of $\overline{P}$ are mapped to the same point under a projective transformation.

As an example for the use of homogeneous coordinates, consider the transformation of a point $P_0 = (X_0, Y_0, Z_0)^T$ in 3D space to a point $P_1 = (X_1, Y_1, Z_1)^T$ by a rotation matrix R and a translation vector t. Instead of applying the rotation and translation separately, the transformation can be combined into a single matrix multiplication:

$$P_1 = RP_0 + t \rightarrow \overline{P_1} = \left[\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right] \overline{P_0}. \tag{3}$$

These homogeneous coordinates are being used in the context of camera geometry to represent points in the camera's image plane and in the 3D world space, allowing for the transformation between these spaces.

2.1.2 Camera Geometry Basics

Camera geometry is essential in computer vision, as it defines how a 3D scene is projected onto a 2D image plane. This process is critical for understanding how cameras capture real-world scenes and is foundational for tasks such as 3D reconstruction and pose estimation. The transformation from 3D world coordinates to 2D image coordinates is governed by a series of matrices and vectors that represent the camera's internal and external parameters, as well as any distortions introduced by the lens. Key components in this transformation include the intrinsic camera matrix, the extrinsic camera matrix, and the distortion vector.

Intrinsic Camera Matrix

The intrinsic camera matrix encapsulates the internal characteristics of the camera, such as the focal length and the optical centre (principal point). It defines how the camera's lens maps 3D points in its coordinate system to 2D points in the image plane. Mathematically, the intrinsic camera matrix $A \in \mathbb{R}^{3\times3}$ is typically represented as

$$A = \begin{bmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

where f_x and f_y are the focal lengths along the x and y axes, respectively, and (c_x, c_y) represent the coordinates of the principal point and α represents the skew factor. The focal lengths are usually expressed in pixels, which scale the 3D points into the image plane. The principal point indicates where the optical axis intersects the image plane. The skew factor accounts for the

non-orthogonality between the image axes. A skew factor of zero means that the x and y axes are perpendicular. This is also the default value for most cameras, including the ones used in this thesis.

Extrinsic Camera Matrix

The extrinsic camera matrix, on the other hand, describes the camera's position and orientation in the world coordinate system. It captures how the camera is situated relative to the objects in the scene, effectively translating and rotating the 3D points from the world coordinate system to the camera's local coordinate system. The extrinsic matrix $[R|t] \in \mathbb{R}^{3\times4}$ is a matrix composed of a rotation matrix $R \in \mathbb{R}^{3\times3}$, which represents the camera's orientation in the 3D space, and a translation vector $t \in \mathbb{R}^3$, which represents the camera's position relative to the world origin, expressed as:

$$[R|t] = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{array}\right] \tag{5}$$

In order to be able to use the extrinsic camera matrix with homogeneous coordinates, a fourth row $(0, 0, 0, 1)$ is added to the matrix:

$$[R|t] \xrightarrow{\text{homogeneous}} \overline{[R|t]} = \left[\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{6}$$

Together, they allow for the transformation of world coordinates to the camera's coordinate system, which is then projected onto the image plane using the intrinsic matrix.

Distortion Vector

In real-world scenarios, lenses introduce distortions that cause deviations from the ideal pinhole camera model. These distortions can be radial, where points farther from the centre of the image are increasingly distorted, or tangential, where points are displaced due to the lens and sensor not being perfectly aligned. The distortion vector $d \in \mathbb{R}^5$ compensates for these imperfections and is usually represented by a set of parameters k_1, k_2, p_1, p_2, and sometimes - and in our case - higher-order terms like k_3. The radial distortion is modelled by k_1, k_2, and k_3, while the tangential distortion is modelled by p_1 and p_2. The distortion vector is represented as

$$d = (k_1, k_2, p_1, p_2, k_3)^T, \tag{7}$$

and figure 1 illustrates the effects of radial and tangential distortion on an image.

Figure 1: Illustration of radial and tangential distortion in a camera lens, from GeeksforGeeks [6].

Mathematically, the undistorted coordinates (x'', y'') can be calculated from the distorted coordinates (x, y) as follows:

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \\ y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y' \end{bmatrix}, \tag{8}$$

with

$$x' = \frac{x - c_x}{f_x} \tag{9}$$

$$y' = \frac{y - c_y}{f_y} \tag{10}$$

$$r^2 = x'^2 + y'^2 \tag{11}$$

2.1.3 Camera Projections Using the Pinhole Model

As described in the OpenCV documentation [4], the concept of camera projections allows for the mapping of real-world coordinates to their corresponding image coordinates. The projection process involves transforming a 3D point P_w in the world coordinate system to a 2D point p in the camera's image plane. This transformation is governed by the previously discussed intrinsic and extrinsic camera matrices of the camera.

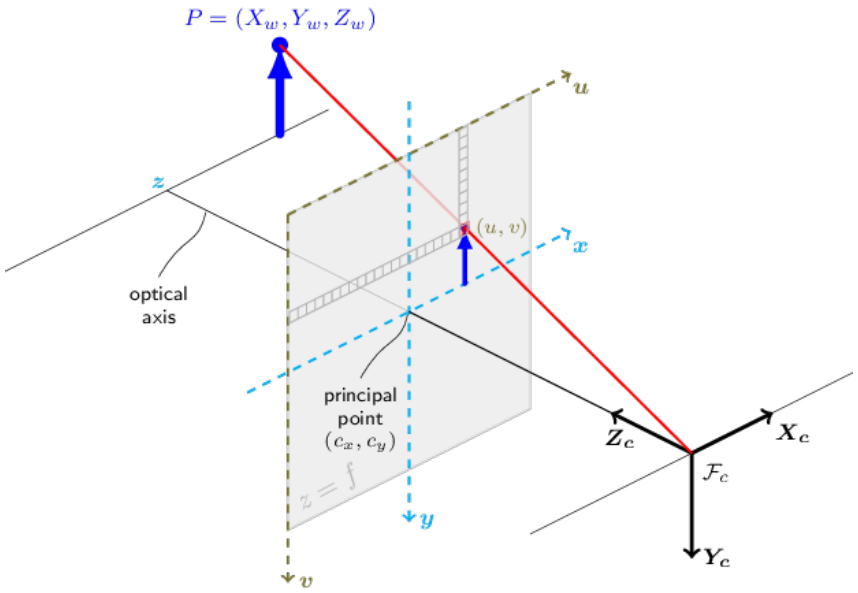


Figure 2: The pinhole camera model, illustrating the projection of a 3D point P_w to a 2D image point p, from the OpenCV documentation [4].

The mathematical relationship governing camera projections is given by:

$$\overline{P_C} = [R|t]\overline{P_w} \tag{12}$$

$$p = AP_C \tag{13}$$

Transforming World Coordinates to Camera Coordinates

The projection process begins by transforming the 3D world points $P_w = (X_w, Y_w, Z_w)^T$ from its world coordinate system to the camera coordinate system. This transformation is done because the camera's perspective depends on its orientation and position relative to the world. Mathematically, this transformation can be described as:

$$\overline{P_C} = [R|t]\overline{P_w}, \tag{14}$$

where $\overline{P_C}$ represents the 3D coordinates of the point in the camera's frame in the homogeneous form. This transformation can be expanded into matrix form, allowing us to see how the rotation and translation affect the point:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{15}$$

Here, X_c, Y_c, and Z_c are the coordinates of the point in the camera's reference frame, while X_w, Y_w, and Z_w are the corresponding coordinates in the world reference frame.

Normalizing Camera Coordinates

Once the point has been transformed into the camera's coordinate system, the next step is to normalize these coordinates by dividing by the depth component Z_c. This normalization step is essential for converting the 3D camera coordinates into 2D coordinates on the image plane:

$$P_C = \frac{\overline{P_C}}{Z_c} \tag{16}$$

This operation can be explicitly written as:

$$\begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \\ 1 \end{bmatrix} \tag{17}$$

Here, x_c and y_c represent the normalized 2D coordinates of the point in the camera's image plane. These coordinates are now independent of the depth Z_c, capturing only the point's relative position as seen by the camera.

Conversion to Pixel Coordinates

The normalized image plane coordinates must then be mapped to actual pixel coordinates on the 2D image. This mapping is achieved through the intrinsic camera matrix A. The relationship between the normalized coordinates and the pixel coordinates p_i is given by:

$$p = AP_C \tag{18}$$

Expanding this equation, we have:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} \tag{19}$$

This matrix multiplication results in the pixel coordinates x' and y', which directly correspond to the location of the point on the camera's image sensor. It is important to note that this transformation is valid only if $Z_c \neq 0$; if $Z_c = 0$, the point p is set to $(0, 0, 1)^T$ to avoid undefined behaviour.

Distortion Correction

Real-world cameras often introduce various types of distortions, mainly radial and tangential, due to lens imperfections. These distortions cause the captured image to deviate from the ideal pinhole camera model. To correct for such distortions and obtain accurate pixel coordinates, additional corrections are applied:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x x'' + c_x \\ f_y y'' + c_y \end{bmatrix} \tag{20}$$

In this correction step, u and v represent the final undistorted pixel coordinates, and f_x, f_y, c_x, and c_y are the camera's focal length and principal point coordinates respectively. The intermediate corrected coordinates x'' and y'' are derived using the following equations that account for the radial distortion parameters k_1, k_2, k_3, and tangential distortion parameters p_1, p_2:

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \\ y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y' \end{bmatrix} \tag{21}$$

Where the normalized coordinates x' and y' are calculated as:

$$x' = \frac{x - c_x}{f_x}, \quad y' = \frac{y - c_y}{f_y} \tag{22}$$

Moreover, the radial distance r^2 is given by:

$$r^2 = x'^2 + y'^2 \tag{23}$$

These equations systematically correct the distortions introduced by the lens, ensuring that the final pixel coordinates u and v are as close as possible to the actual location of the 3D point's projection onto the image plane.

2.2 Gaussian Mixture Models

As described by Weinmann [7], Gaussian Mixture Models (GMMs) are a versatile and widely used statistical tool for modelling complex distributions in data. They represent a mixture of multiple Gaussian distributions, each characterized by its mean, covariance, and weight. This approach allows GMMs to capture the underlying data structure that might arise from multiple data clusters, each with its Gaussian distribution. GMMs are particularly valuable in scenarios where data exhibits multimodality, generated from several different sources or processes, each contributing to the overall distribution.

A Gaussian Mixture Model is defined by a weighted sum of Gaussian distributions, expressed mathematically as:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k), \tag{24}$$

where $p(x)$ represents the probability density function of the data x, K is the number of Gaussian components in the mixture, π_k is the weight of the k^{th} Gaussian component (with $\sum_{k=1}^{K} \pi_k = 1$), μ_k is the mean vector, and Σ_k is the covariance matrix of the k^{th} Gaussian component. Each Gaussian component $\mathcal{N}(x|\mu_k, \Sigma_k)$ describes a cluster or subpopulation within the data, and the weights π_k indicate each component's relative importance.

GMMs are widely applied across various fields, including image processing, speech recognition, and anomaly detection. In image processing, for example, GMMs are used for background subtraction in video sequences, where they help to model and separate moving objects from the static background. In speech recognition, GMMs are employed to model the distribution of acoustic features in different phonetic classes. Their flexibility in modelling complex, multimodal distributions makes GMMs a powerful tool for data analysis and pattern recognition. Estimating the parameters of a GMM—namely the means, covariances, and weights—typically involves using the EM algorithm, which will be discussed in section 2.3 Expectation-Maximization Algorithm.

Gaussian Mixture Models provide a robust framework for modelling and analysing data with inherent variability and multiple underlying distributions. By combining multiple Gaussian components, GMMs can effectively capture and represent complex data structures, making them a valuable tool for various applications in machine learning and statistical analysis. Their ability to adapt to different data distributions through the EM algorithm further enhances their utility in practical scenarios.

2.3 Expectation-Maximization Algorithm

The Expectation-Maximization (EM) algorithm is a powerful iterative technique for estimating maximum likelihood parameters in statistical models, particularly when the model involves

hidden variables. It is employed in various fields, such as machine learning, statistics, and data mining, for problems where data is incomplete or has missing values. The EM algorithm handles these challenges by iteratively improving the parameter estimates until convergence is achieved.

The EM algorithm operates through two main steps: the Expectation (E) step and the Maximization (M) step. During the E-step, the algorithm computes the log-likelihood function's expected value concerning the parameters' current estimates and the distribution of the latent variables. This step involves calculating the probability distribution of the latent variables given the observed data and the current parameter estimates.

In the M-step, the algorithm maximizes this expected log-likelihood function with respect to the model parameters. The goal is to find parameter values that maximize the likelihood of the observed data, considering the expectations computed in the E-step. This results in updated parameter estimates used in the next iteration of the E-step. The process repeats, iterating between these two steps, until the changes in the log-likelihood or the parameter estimates fall below a predefined threshold, indicating convergence.

The EM algorithm is notably used in various applications, including GMMs, Hidden Markov Models (HMMs), and clustering problems. For instance, in GMMs, the EM algorithm helps estimate the Gaussian components' parameters that best fit the observed data. Similarly, in HMMs, it estimates transition and emission probabilities from sequences of observed symbols.

2.4 Log-Sum-Exp Trick

As the previously discussed concepts, specifically the GMMs and EMs (see sections 2.2 Gaussian Mixture Models and 2.3 Expectation-Maximization Algorithm), rely heavily on probabilities which can get exceptionally small using a computer, it is essential to understand the log-sum-exp trick. As described by Mao [8], the log-sum-exp trick is a numerical technique to enhance the stability and accuracy of computations involving logarithmic and exponential functions. In many optimisation and statistical problems, expressions involving the logarithm of a sum of exponentials, such as

$$\log\left(\sum_i e^{x_i}\right), \tag{25}$$

can lead to numerical instability, particularly when dealing with large or small values of x_i. This instability arises from the potential for overflow or underflow when exponentiating large values or the precision loss when dealing with very small ones. The log-sum-exp trick addresses this issue by transforming the expression into a more numerically stable form. Specifically, the trick involves rewriting equation 25 as

$$\max_i(x_i) + \log\left(\sum_i e^{x_i - \max_i(x_i)}\right). \tag{26}$$

Here, $\max_i(x_i)$ is the maximum value among x_i, which shifts the exponentials to avoid extreme values and mitigate numerical instability. This reformulation ensures that the exponential terms are computed relative to the largest value, reducing the risk of overflow or underflow and improving computational precision. The log-sum-exp trick is widely used in machine learning, where it is crucial for stabilizing calculations involving probabilistic models and large datasets.

2.5 Basis Splines

As described by Stingelin [9], Basis Splines, commonly known as B-Splines, are a family of piecewise-defined polynomials that are used in computational graphics, computer-aided design (CAD), and data interpolation due to their flexibility and numerical stability. They provide a robust framework for representing complex curves and surfaces in a smooth and controlled manner, making them essential tools in various applications that require precise and adaptable geometric modelling.

B-Splines are defined over a domain divided into intervals, and each interval is associated with a polynomial segment. These splines are constructed using a set of basis functions that are piecewise polynomials. This ensures that the overall spline curve is smooth at the points where polynomial segments meet, known as the knots. A B-Spline of degree p is represented as a linear combination of $p+1$ basis functions, each weighted by a set of control points. The general form of a B-Spline curve can be expressed as:

$$C(t) = \sum_{i=0}^{n} N_{i,p}(t) \cdot P_i, \tag{27}$$

where $C(t)$ is the B-Spline curve, $N_{i,p}(t)$ are the B-Spline basis functions of degree p at time t, P_i are the control points, and n is the number of control points minus one. The basis functions $N_{i,p}(t)$ are constructed recursively, ensuring that they are non-negative and sum to 1 over the entire domain, guaranteeing that the spline curve is always within the convex hull of its control points.

One of the key advantages of B-Splines is their local control property, meaning that adjustments to a single control point affect only a local segment of the spline curve rather than the entire curve. This local control makes B-Splines particularly useful for interactive design and modelling, where precise adjustments are necessary. Additionally, B-Splines are well-suited for approximating and interpolating data, as they can fit complex curves seamlessly while maintaining smoothness and continuity.

B-Splines find applications in various fields. In computer graphics and CAD, they are used to design and render smooth curves and surfaces with a high degree of control. In data fitting and interpolation, B-Splines provide a flexible approach to approximate datasets, ensuring that the resulting curves accurately represent the underlying trends in the data. In animation and motion capture, B-Splines generate smooth transitions between keyframes, allowing for natural and fluid animations.

Figure 3 illustrates the basis functions of a B-Spline curve of degree 3 and figure 4 shows an example of a B-Spline curve approximating noisy data points of the function $\sin(\alpha)$.

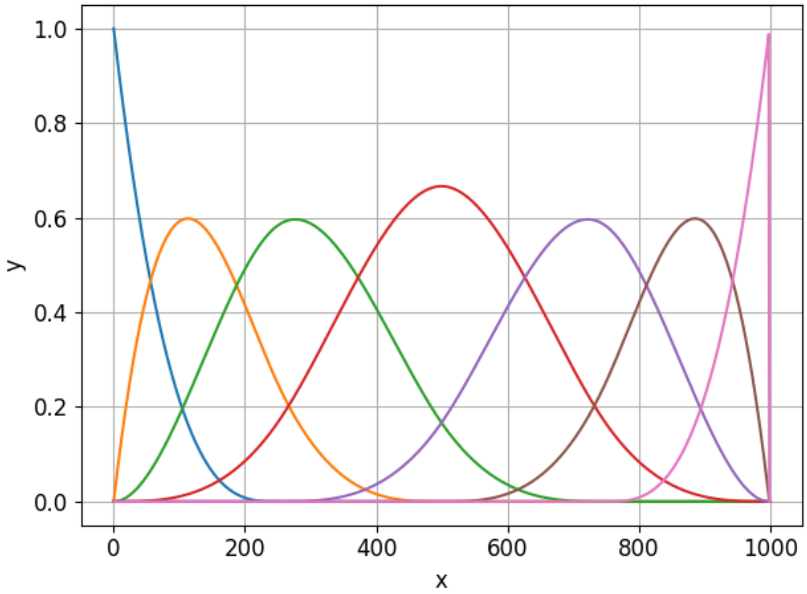


Figure 3: B-Spline basis functions of degree 3.

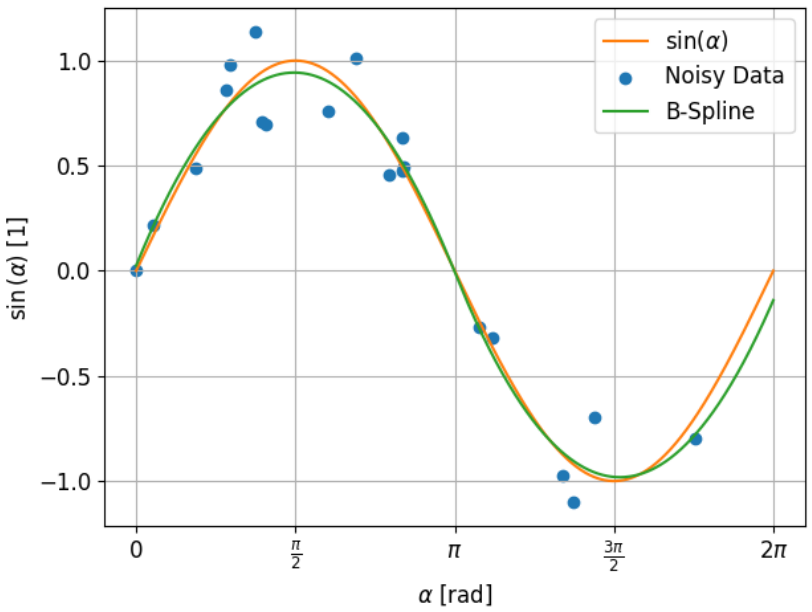


Figure 4: B-Spline curve approximating noisy data points of the function $\sin(\alpha)$.

3 Methodology

In this section, the methodology for developing and evaluating the MC3D-TRECSIM system is described. A structured approach is followed to ensure the system's accuracy and robustness in capturing and reconstructing 3D trajectories from multi-camera setups.

3.1 Key Libraries and Tools

The MC3D-TRECSIM system leverages several key libraries and tools to facilitate the development and implementation of the trajectory reconstruction and simulation framework. These libraries provide essential functionalities for camera calibration, 3D reconstruction, optimisation, and visualisation, enabling the system to process multi-camera data efficiently and accurately. The following subsections provide an overview of the libraries used in the system and their respective roles in supporting the trajectory reconstruction and simulation process. The part of the system that is responsible for the trajectory reconstruction is written in C++ while the visualisation is done using Python 3.12.

3.1.1 Eigen

Eigen [10] is a C++ library optimized for linear algebra operations, providing functionalities such as matrix and vector arithmetic, numerical solvers, and various matrix decompositions. Its template-based design supports efficient computation, making it suitable for applications in scientific computing, robotics, and machine learning. Eigen prioritizes performance while maintaining accuracy, utilizing features like expression templates and auto-vectorization to enhance computational efficiency. In the MC3D-TRECSIM system, Eigen is employed for linear algebra operations essential to the optimization algorithms, camera projections, and trajectory reconstruction, enabling the system to perform the necessary computations for real-time processing and analysis of multi-camera data.

The Eigen library was chosen specifically for its performance, flexibility, and ease of use, making it an ideal choice for handling the mathematical operations involved in the MC3D-TRECSIM system.

3.1.2 LBFGS++

LBFGS++ [11] is a header-only C++ library which provides an efficient implementation of the L-BFGS optimisation algorithm. LBFGS++ is designed to ease integration into existing C++ projects, offering a simple API for specifying the objective function to be optimised and the initial parameter values.

The Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimisation algorithm is a variant of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, tailored for high-dimensional optimisation problems where memory efficiency is critical. L-BFGS is particularly

well-suited for applications in machine learning and computer vision, where the number of parameters can be exceedingly large. Unlike the full BFGS algorithm, which requires storing and manipulating a dense approximation of the Hessian matrix, L-BFGS maintains a limited memory of the most recent updates to the gradient and position vectors. This allows the algorithm to approximate the inverse Hessian matrix using only a small subset of vectors, significantly reducing memory usage while still achieving effective optimisation. L-BFGS is an iterative algorithm that adjusts the search direction based on the history of the gradients, enabling it to converge efficiently even in complex, non-linear optimisation landscapes. This makes it an ideal choice for optimising the parameters of large-scale models, such as those encountered in training deep neural networks or fitting complex probabilistic models in computer vision tasks.

The L-BFGS algorithm, specifically the LBFGS++ library, was chosen because of the ready-to-use implementation and support for the Eigen library.

3.1.3 YOLOv7

2D pose estimation is a fundamental task in computer vision, aiming to detect the locations of key points or landmarks on the human body within a 2D image. These key points typically include joints such as the elbows, knees, and shoulders, which are necessary for understanding human posture and movement. The task has significant applications across various domains, including human-computer interaction, animation, sports analytics, and surveillance. The primary goal of 2D pose estimation is to accurately map these key points in the image plane, enabling the analysis of human activities in various environments.

A notable development in the realm of 2D pose estimation is the introduction of the YOLOv7 model, which is part of the YOLO family of real-time object detection systems. The YOLOv7 architecture, as detailed in the work of Wang et al. [12], builds upon previous versions by incorporating a series of innovations that improve both accuracy and efficiency. This includes a trainable "bag of freebies", which refers to techniques that enhance model performance without increasing inference time. YOLOv7 also introduces new network architecture optimisations and training strategies that significantly boost the model's ability to detect and localize key points for pose estimation.

Furthermore, Wong and Wong [13] have extended the capabilities of YOLOv7, focusing on enhancing the model's performance specifically in 2D pose estimation tasks. Their work demonstrates how the model can be fine-tuned to accurately predict human key points, even in challenging scenarios with occlusions and varied poses. The adaptability of YOLOv7 to 2D pose estimation tasks highlights its versatility and the potential for its application in real-world scenarios where both accuracy and speed are critical.

The advancements brought by YOLOv7 represent a significant leap forward in the 2D pose estimation field. By leveraging the strengths of deep learning and real-time object detection,

these models provide a robust solution for accurately mapping human key points in 2D space. Their code is available on GitHub [13] as an open-source project used in this thesis. Figure 5 shows an example of a 2D pose estimation result obtained from the YOLOv7 model.

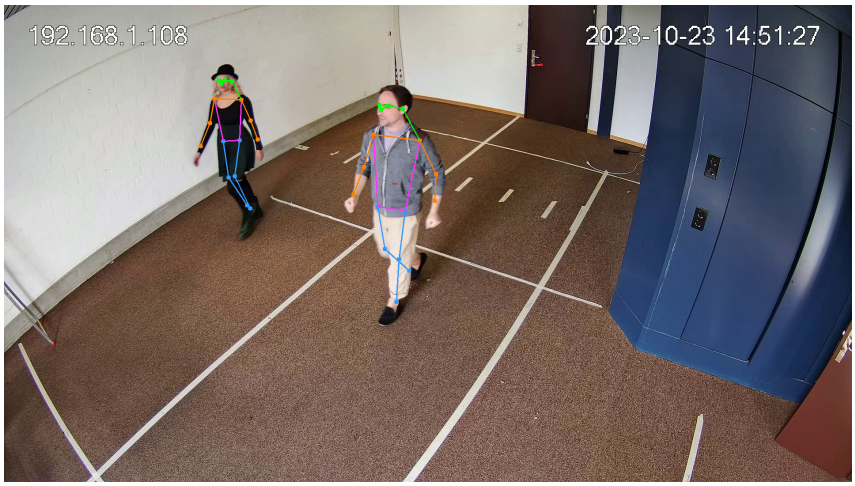


Figure 5: Skeletons of a human bodies with 17 key points obtained from YOLOv7.

3.1.4 Visualisation Software

This section discusses two essential tools instrumental in the visualisation pipeline: PyQtGraph for 3D visualisation and OpenCV for capturing, undistorting and visualising camera images.

PyQtGraph [14] is a Python-based library that excels in real-time visualisation, particularly for applications requiring 3D graphics. It leverages the power of OpenGL for rendering 3D scenes, making it highly suitable for visualising spatial relationships, such as the positions and orientations of cameras in a multi-camera setup or the tracking of people within a 3D environment.

In the context of camera networks, PyQtGraph allows for the real-time visualisation of camera fields of view, helping to understand how different cameras cover a scene. Additionally, PyQtGraph can represent people as points or skeletal models within this 3D space, enabling the visualisation of their movement trajectories relative to the cameras.

The interactive features of PyQtGraph, such as zooming, panning, and rotating the 3D scene, provide an intuitive way for users to explore and analyse the data, making it a powerful tool for both development and presentation purposes.

OpenCV [15] is a versatile library widely used in computer vision for capturing, processing, and visualising images. One of its core strengths lies in its ability to interface with various camera hardware, making it a preferred choice for capturing live video streams and images.

OpenCV provides a comprehensive set of tools for acquiring images from cameras, processing them in real-time, and displaying them with annotations such as bounding boxes, tracking paths, and other visual markers. Beyond visualisation, OpenCV is also used to undistorting images obtained from calibrated cameras.

3.2 Camera Calibration

The MC-Calib [16] [17] toolbox is used to calibrate the cameras in the multi-camera system, which is a critical step in ensuring the accuracy and reliability of a multi-camera system. MC-Calib is a specialised tool designed to calibrate multi-camera setups, where precise geometric relationships between cameras must be established to achieve correct 3D reconstructions and measurements.

The calibration process begins with the physical setup of the cameras, ensuring that they are securely mounted and oriented towards the area of interest. Ensuring the cameras have overlapping fields of view is crucial for effective stereo matching and 3D reconstruction. Once the cameras are in place, the calibration procedure can be initiated using a calibration object, typically a checkerboard pattern, which is moved through all cameras' shared field of view.

All cameras capture the checkerboard simultaneously in multiple positions and orientations. These images are then processed by MC-Calib to detect the corners of the checkerboard, which serve as reference points. The software uses these detected points to estimate the intrinsic matrix A of each camera individually, as well as the extrinsic matrix $\overline{[R|t]}$ that describes the spatial relationship between the cameras and the distortion vector d as described in section 2.1.2 Camera Geometry Basics.

After capturing and processing the necessary images, MC-Calib performs optimisation to minimise the reprojection error, which is the difference between the observed and projected positions of the checkerboard corners. This optimisation ensures that the estimated camera parameters accurately represent the physical setup. The final output includes the intrinsic parameters of each camera, the extrinsic parameters describing the relative positions and orientations of the cameras, and a set of rectification matrices that can be used to align the images from all cameras to a common reference frame.

Once the calibration is complete, it is essential to validate the results by checking the reprojection error and, if necessary, refining the calibration by capturing additional images or adjusting the camera setup. Proper calibration using MC-Calib ensures the multi-camera system is accurately configured, enabling precise 3D reconstructions and measurements in subsequent tasks.

3.3 Laboratory Setup

The laboratory setup for this research was established in a spacious room designed to accommodate a two-camera system for computer vision experiments. Two high-resolution Dahua

IPCHDBW3841RZSS2 with a capture rate of 20 frames per second (FPS) [18] cameras (see figure 6) were strategically positioned within the room to cover the area of interest with overlapping fields of view, ensuring comprehensive capture of the scene. These cameras were connected to a central computer with a powerful graphics card (NVIDIA RTX 4090), specifically chosen to support the computational demands of the YOLOv7 object detection framework, and a 13th Gen Intel Core i9 processor. This setup allowed for real-time processing and analysis of the captured video streams, enabling the efficient testing and evaluation of various computer vision algorithms within the laboratory environment.

Figure 6: Dahua IPCHDBW3841RZSS2 camera used in the laboratory setup.

After calibrating the cameras by taking 46 images of a checkerboard pattern on many different positions, the intrinsic and extrinsic parameters were estimated using the MC-Calib toolbox. The calibration frames of the cameras are shown in figure 7, and the 3D representation of the cameras in the laboratory setup is visualised in figure 8.

(a) Calibration frame of first camera

(b) Calibration frame of second camera

Figure 7: Calibration frames of the cameras used in the old laboratory setup.

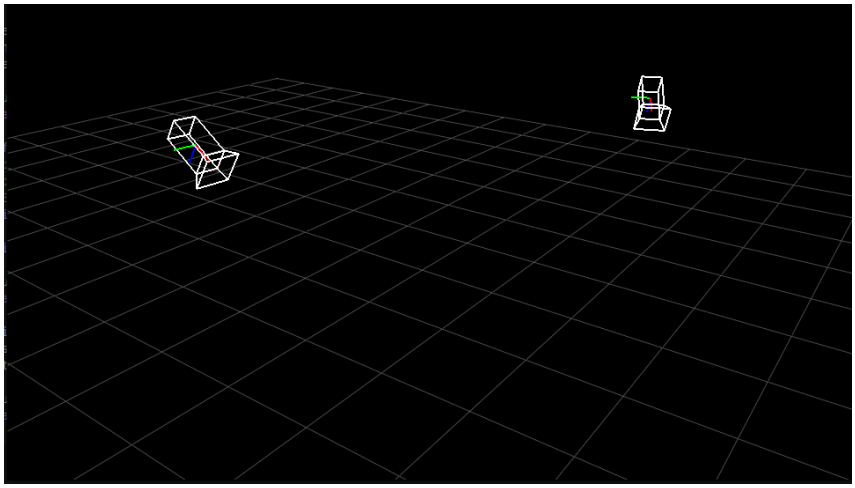


Figure 8: 3D reconstruction of the cameras in the laboratory setup using PyQtGraph [14].

Halfway through the project, the laboratory setup had to be changed to a new room (see figure 9) with the same camera setup because the building, the laboratory was located in, was being demolished to make way for a new building.

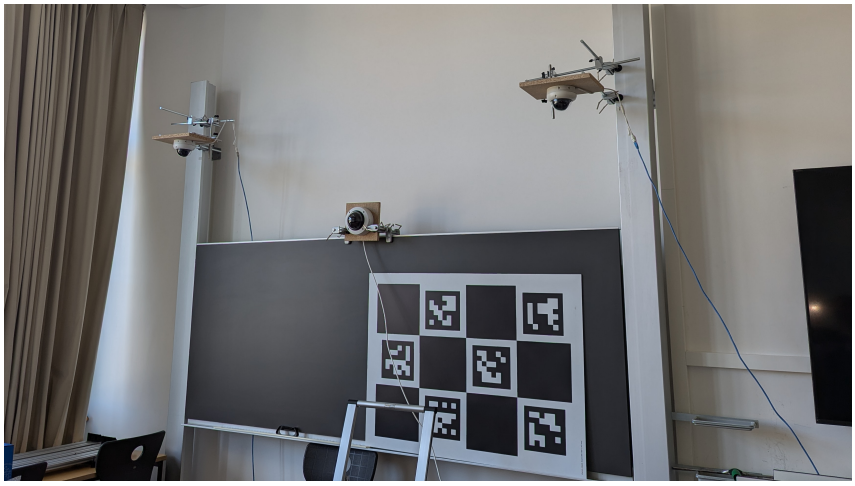
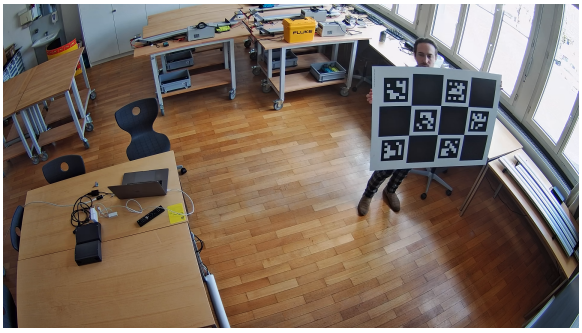


Figure 9: New laboratory room with the same camera setup. The centre camera is part of another setup.

After moving the cameras, the calibration process had to be repeated. The calibration frames of the new cameras are shown in figure 10, and the 3D representation of the cameras in the new laboratory setup is visualised in figure 11.

(a) Calibration frame of first camera



(b) Calibration frame of second camera

Figure 10: Calibration frames of the cameras used in the new laboratory setup.

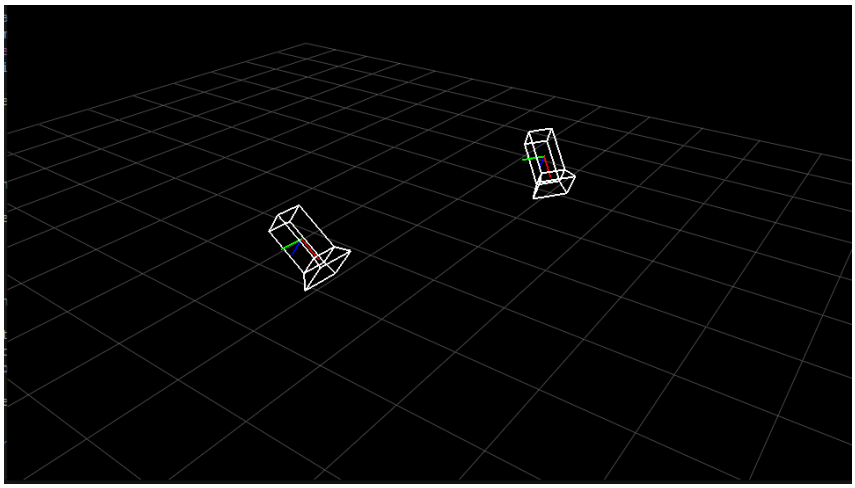


Figure 11: 3D reconstruction of the cameras in the laboratory setup using PyQtGraph [14].

3.4 Outline of Base Model

The process of reconstructing the trajectories of individuals begins by assuming that there are initially no people in the scene. The system then proceeds to retrieve frames from each camera in the network. This multi-camera setup allows for a comprehensive view of the scene from different angles, crucial for accurate 3D reconstruction and tracking.

Once the frames are captured they get undistorted using OpenCV and the system employs the YOLOv7 model to detect people within each frame. Upon detecting people in the frames, the model also identifies key points associated with each person, such as the nose, right shoulder, left shoulder, right hand, and other critical joints. These key points are essential for constructing a detailed 3D representation of each person in the scene. After detecting key points across all camera frames, the system estimates the number of people present.

For each identified key point type (e.g., nose, right shoulder), determining their accurate 3D positions begins. Initially, the system defines starting parameters for a B-Spline curve for each dimension (x, y, z) and for each person estimated to be in the scene. These 3D points, derived from the B-Spline parameters, are then projected onto the 2D image planes of the camera frames. This projection is a critical step as it connects the 3D model with the 2D observations, enabling the system to compare the hypothesised 3D positions with the actual detected key points in the 2D images.

To determine which detected key point corresponds to which projected 3D point, the system uses a GMM. Specifically, the GMM consists of one bivariate Gaussian distribution for each person in the scene. This model allows the system to estimate the likelihood that a particular detected key point belongs to a specific projected 3D point (see figure 12).

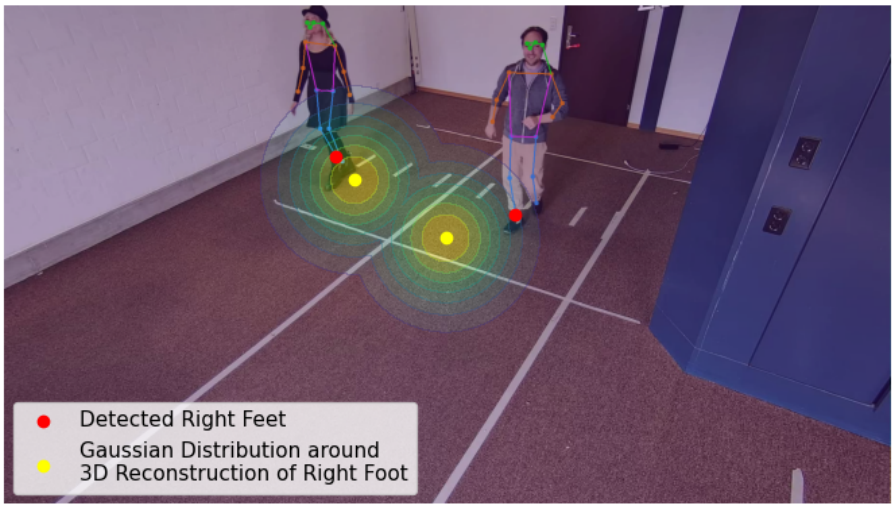


Figure 12: Gaussian responsibility distributions around the projection of 3D reconstructed right feet.

To optimise the correspondence between detected key points and their projected 3D counterparts, the EM algorithm is employed. The EM algorithm iteratively adjusts the parameters of the GMM to maximise the likelihood, thereby refining the accuracy of the key point assignments. This process of projecting 3D points, assigning detected key points using the GMM, and optimising the parameters through EM is repeated until the system converges on a stable solution, where the key point assignments and 3D positions are accurate.

Finally, the system displays the results, which include the reconstructed 3D skeleton of each person at the current time. Additionally, the system visualises the paths of each key point, show-

ing their movement over a series of frames into both the past and the future. This continuous visualisation allows for real-time monitoring and analysis of the scene.

The entire process, from frame retrieval to result display, is repeated indefinitely. This continuous loop ensures that the system can adapt to changes in the scene, such as people entering or leaving, and maintain accurate tracking and reconstruction over time.

3.5 Mathematical Representation of Base Model

Let J represent the expected number of individuals to be tracked. For each person j, their movement over time t can be described by a function $h_j(t|\theta_j) : \mathbb{R} \to \mathbb{R}^3$, which returns their position in 3D space. The system captures measurements $m_n \in \mathbb{R}^2$, corresponding to key points in2D images taken from a specific camera k_n. The function $P_K : \mathbb{R}^3 \to \mathbb{R}^2$ then projects these 3D points onto the2D plane of the camera image.

3.5.1 Mixture Model

A mixture model is employed to model the observations, where $z = (z_1, \ldots, z_n)$ represents the hidden variables that indicate which person j corresponds to each measurement m_n. The covariance matrix for these measurements is given by $\Sigma = \nu \cdot I_2$, where ν is a scalar and I_2 is the 2×2 identity matrix. The probability vector π represents the likelihood of each assignment z_n. The mixture model is defined as follows:

$$\mathbb{P}(m_n|z_n = j, \theta) = \mathcal{N}(m_n|P_K(h_j(t_n|\theta_j)), \Sigma) \tag{28}$$

$$\mathbb{P}(z_n = j|\pi) = \pi_j \tag{29}$$

$$\mathbb{P}(m_n|\theta) = \sum_{j=1}^{m} \mathbb{P}(m_n|z_n = j, \theta) \cdot \mathbb{P}(z_n = j|\pi) \tag{30}$$

The total likelihood of the data D and hidden variables Z given the parameters θ and π is:

$$\mathbb{P}(D, Z|\theta, \pi) = \prod_{n=1}^{N} \mathbb{P}(m_n|z_n = j, \theta) \cdot \mathbb{P}(z_n = j|\pi) \tag{31}$$

Taking the logarithm of the likelihood function simplifies to:

$$\ln\left(\mathbb{P}(D, Z|\theta, \pi)\right) = \sum_{n=1}^{N} \ln\left(\mathbb{P}(m_n|z_n = j, \theta)\right) + \ln\left(\mathbb{P}(z_n = j|\pi)\right) \tag{32}$$

3.5.2 EM Algorithm

The EM algorithm is employed to iteratively estimate the parameters θ and π. During the expectation step, the responsibility $r_{n,j}$ of each measurement m_n being associated with person j is computed as:

$$r_{n,j} = \mathbb{P}(z_n = j | m_n, \theta_t, \pi_t) = \frac{\mathbb{P}(m_n | z_n = j, \theta_t) \cdot \mathbb{P}(z_n = j | \pi)}{\sum_{l=1}^{m} \mathbb{P}(m_n | z_n = l) \cdot \mathbb{P}(z_n = l | \pi)} \tag{33}$$

This leads to the expected log-likelihood function:

$$E_{\mathbb{P}(Z|D,\theta_t,\pi_t)} \ln(D, Z | \theta, \pi) = \sum_{n=1}^{N} \sum_{j=1}^{J} r_{n,j} (\ln(\mathbb{P}(m_n | z_n, \theta)) + \ln(\pi_j)) \tag{34}$$

In the maximisation step, this expected log-likelihood is maximised w.r.t. the parameters θ and π. For π, the update rule is:

$$\pi_{t+1,j} = \frac{r_j}{R} \tag{35}$$

with

$$r_j = \sum_{n=1}^{N} r_{n,j} \quad \text{and} \quad R = \sum_{j=1}^{J} r_j \tag{36}$$

For θ, the maximisation involves optimising

$$\underset{\theta_1,\ldots,\theta_m}{\arg\max} \sum_{n=1}^{N} \sum_{j=1}^{J} r_{n,j} \left(-\ln(2\pi) - \ln(\nu) - \frac{1}{2\nu} \| m_n - P_w \|^2 \right) \tag{37}$$

where $P_w = P_K(h_j(t_n | \theta_j))$ represents the projection of the estimated 3D position onto the2D image plane.

3.5.3 B-Spline Trajectory Representation

The trajectories of individuals are modeled using splines, where $X = (\Phi_1(t_{m_n}), \ldots, Phi_K(t_{m_n}))$ represents the row vector of spline basis functions at time t (for one hypothesis). This can be represented in matrix form:

$$A \cdot \theta = P_w \tag{38}$$

$$A \in \mathbb{R}^{M \times K} \quad \theta \in \mathbb{R}^{K \times 3J} \quad P_w \in \mathbb{R}^{M \times 3J}, \tag{39}$$

where M is the number of measurement points (key points extracted by YOLOv7), K is the number of B-Spline basis functions, and J is the number of hypotheses. Here shown is also the explicit form:

$$\underbrace{\begin{bmatrix} \Phi_1(t_{m_1}) & \Phi_2(t_{m_1}) & \dots & \Phi_K(t_{m_1}) \\ \Phi_1(t_{m_2}) & \Phi_2(t_{m_2}) & \dots & \Phi_K(t_{m_2}) \\ & \vdots & & \\ \Phi_1(t_{m_M}) & \Phi_2(t_{m_M}) & \dots & \Phi_K(t_{m_M}) \end{bmatrix}}_{\text{design matrix } A} \cdot \underbrace{\begin{bmatrix} \theta_{1,1}^X & \theta_{1,1}^Y & \theta_{1,1}^Z \\ \theta_{2,1}^X & \theta_{2,1}^Y & \theta_{2,1}^Z \\ & \vdots & \\ \theta_{K,1}^X & \theta_{K,1}^Y & \theta_{K,1}^Z \end{bmatrix}}_{\text{3D parameters } \theta} = \underbrace{\begin{bmatrix} P_{W_1}^X & P_{W_1}^Y & P_{W_1}^Z \\ P_{W_2}^X & P_{W_2}^Y & P_{W_2}^Z \\ & \vdots & \\ P_{W_M}^X & P_{W_M}^Y & P_{W_M}^Z \end{bmatrix}}_{\text{reconstructed 3D points } P_w} . \tag{40}$$

Each row of the design matrix represents a measurement point, each column of the 3D parameters matrix constitutes the parameters of the B-Spline for each dimension (x, y, z), and each row of the reconstructed 3D points stands for the (X, Y, Z) the reconstructed point in 3D space for measurement M. The derivative of the trajectory w.r.t. the parameters θ_j is then represented by

$$\frac{\partial h_j}{\partial \theta_j} = \begin{bmatrix} X & 0 & 0 \\ 0 & X & 0 \\ 0 & 0 & X \end{bmatrix} \tag{41}$$

3.5.4 Projection Function

As described in section 2.1.3 Camera Projections Using the Pinhole Model, the projection function P_K maps the 3D trajectory to the2D image plane. The derivative of this function w.r.t. the 3D coordinates are given by:

$$P_{W_{\text{inv}}} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R^T \cdot (P_w - t) \tag{42}$$

$$P_{C_{\text{inv}}} = \begin{bmatrix} 1/Z & 0 & -X/Z^2 \\ 0 & 1/Z & -Y/Z^2 \\ 0 & 0 & 0 \end{bmatrix} \tag{43}$$

$$P_C = A \cdot P_{C_{\text{inv}}} \cdot R^T \tag{44}$$

$$\frac{\partial P_K}{\partial h_j} = \begin{bmatrix} P_{C00} & P_{C01} & P_{C02} \\ P_{C10} & P_{C11} & P_{C12} \end{bmatrix} \tag{45}$$

This leads to the derivative of the projection function w.r.t. the parameters θ_j:

$$\frac{\partial P_K}{\partial \theta_j} = \frac{\partial P_K}{\partial h_j} \cdot \frac{\partial h_j}{\partial \theta_j}. \tag{46}$$

3.5.5 Derivatives for Optimisation

Finally, the derivative of the log-likelihood function w.r.t. the parameters θ_l is computed as:

$$\frac{\partial f}{\partial \theta_l} = \sum_{j=1}^{J} \sum_{n=1}^{N} r_{n,j} \cdot \nu^{-1} \cdot (m_n - P_K)^T \cdot \frac{\partial P_K}{\partial h_j} \cdot \frac{\partial h_j}{\partial \theta_j} \tag{47}$$

This derivative is used to update the parameters θ in the maximisation step using the L-BFGS algorithm, thereby refining the trajectory estimates of the tracked individuals.

3.6 Model Improvements

In this section, a series of targeted model improvements is explored to enhance the accuracy and stability of trajectory reconstruction and key point fitting in 3D modeling. These improvements address specific challenges encountered in the original modelling approach, such as the oscillations introduced by the B-Splines and the sensitivity of the EM algorithm to parameter initialisation. Regularization techniques are integrated, parameter initialisation methods refined, and responsibility-based support mechanisms implemented to mitigate issues such as noise sensitivity, local minima entrapment, and key point misalignment.

3.6.1 Parameter Management

In the initial approach to parameter management for B-spline fitting all parameters were reinitialised to random values. While this ensured that each fitting process started from a neutral state, it introduced significant computational overhead.

A more efficient parameter management strategy was implemented to optimise the fitting process. Instead of reinitialising the parameters before every fit, the previous fit's parameters are retained and used as the starting point for the subsequent fit. This method leverages the assumption that the parameters from the previous time step are likely close to the current optimal values, given the gradual nature of changes in the 3D positions being modelled.

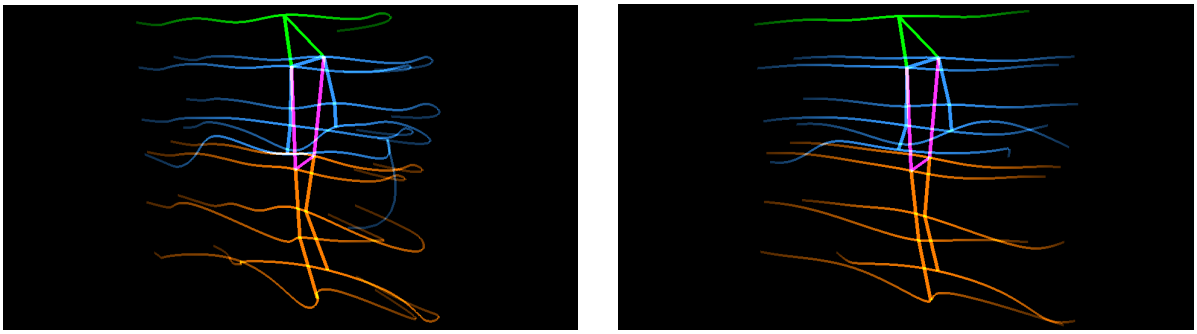
Moreover, when a new key point is added that extends beyond the current highest knot of the B-Spline, the parameter matrix of equation 40 is padded with the latest available parameters. This process is illustrated in equation 48, where the parameter matrix is extended by adding a row corresponding to the most recent parameter values.

$$\theta_{t+1} = \left[\begin{array}{c} \theta_t \\ \hline \theta^X_{t\,K,1} \quad \theta^Y_{t\,K,1} \quad \theta^Z_{t\,K,1} \end{array}\right] \tag{48}$$

This method of managing the parameters ensures they are already near the optimal values, thereby reducing the number of iterations required for convergence in each fit. Consequently, this significantly reduces the overall computation time, making the fitting process more efficient without sacrificing accuracy.

3.6.2 B-Spline Smoothing

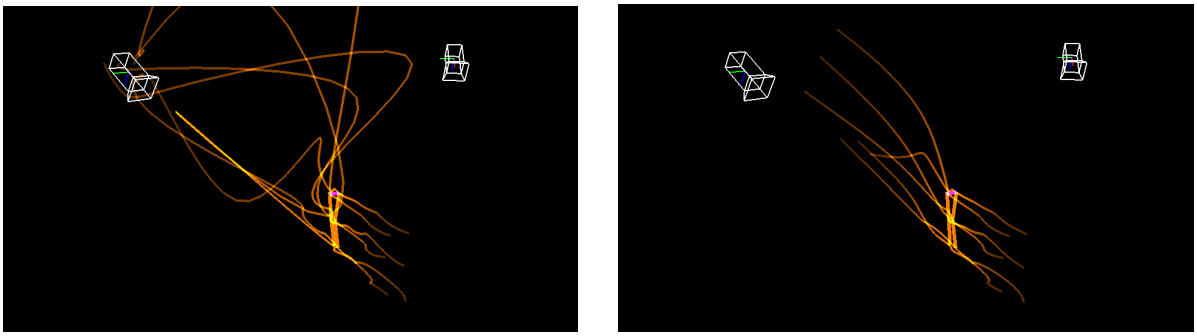
B-spline smoothing is a powerful tool for reconstructing trajectories from noisy data, but it can also introduce challenges, as described by Dahmen and Reusken [19]. One significant issue is that B-splines can lead to high oscillations in the reconstructed trajectory. These oscillations can be caused by several factors, including noise in the data, the presence of outliers, and the high degrees of freedom associated with the B-spline model. Such oscillations often result in a trajectory that is not smooth and reacts strongly to outliers. This issue becomes problematic if the YOLOv7 network, which provides the initial predictions, is not accurate. Inaccuracies in the network's predictions can lead to unrealistic movements in the trajectory, especially during the early stages of reconstruction or when a person exits the scene. For example, as shown in figure 13, the beginning of the trajectory may exhibit squiggly lines, and wobbly bumps may appear during movement.



(a) Without smoothing (b) With smoothing

Figure 13: Comparison of 3D trajectory reconstruction without and with B-Spline smoothing.

Additionally, as depicted in figure 14, the trajectory may "explode" unnaturally when a person leaves the scene, in contrast to the expected smooth collapse.



(a) Without smoothing (b) With smoothing

Figure 14: Comparison of 3D trajectory reconstruction without and with B-Spline smoothing at exit.

To address these issues, a regularisation term can be added to the design matrix of the B-Spline. This regularisation term effectively limits the movement speed within the reconstructed trajectory, thereby reducing the trajectory's sensitivity to noise and outliers. The regularization matrix, denoted as B_T, is defined as follows:

$$B_T = \begin{bmatrix} b_{3,1} & b_{3,2} & b_{3,3} & & & \dots & 0 \\ 0 & b_{4,2} & b_{4,3} & b_{4,4} & & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & b_{p-1,p-3} & b_{p-1,p-2} & b_{p-1,p-1} & 0 \\ 0 & \dots & & & b_{p,p-2} & b_{p,p-1} & b_{p,p} \end{bmatrix} \in \mathbb{R}^{(p-2)\times(p)}, \tag{49}$$

with

$$p = k + l \tag{50}$$

$$d_j = \frac{(k-1)(k-2)||N_{j,k-2}||_2}{t_{j+k-2} - t_j} \tag{51}$$

$$b_{j,j-2} = \frac{d_j}{t_{j+k-2} - t_{j-1}} \tag{52}$$

$$b_{j,j} = \frac{d_j}{t_{j+k-1} - t_j} \tag{53}$$

$$b_{j,j-1} = -(b_{j,j-2} + b_{j,j}) \tag{54}$$

$$||N_{j,k-2}||_2 : \quad \text{Integral of Basis Function } j \text{ and degree } k-2 \tag{55}$$

The elements of the regularisation matrix are derived based on the degree of the B-spline and the intervals between the knots. By appending the regularisation matrix B_T to the B-spline design matrix, the optimisation problem is modified to minimise the 2-norm of the difference between the design matrix and the function values:

$$\left|\left| \begin{bmatrix} A_T \\ \lambda B_T \end{bmatrix} \cdot c - \begin{bmatrix} f \\ 0 \end{bmatrix} \right|\right|_2 \to \min \tag{56}$$

Here, $\lambda \in \mathbb{R}$ represents the regularisation factor, and $f \in \mathbb{R}^n$ is the vector of function values. The regularisation factor λ controls the strength of the regularisation. Figure 15 shows a comparison of different smoothing factors λ for a B-Spline approximation of the function $y = x^2$.

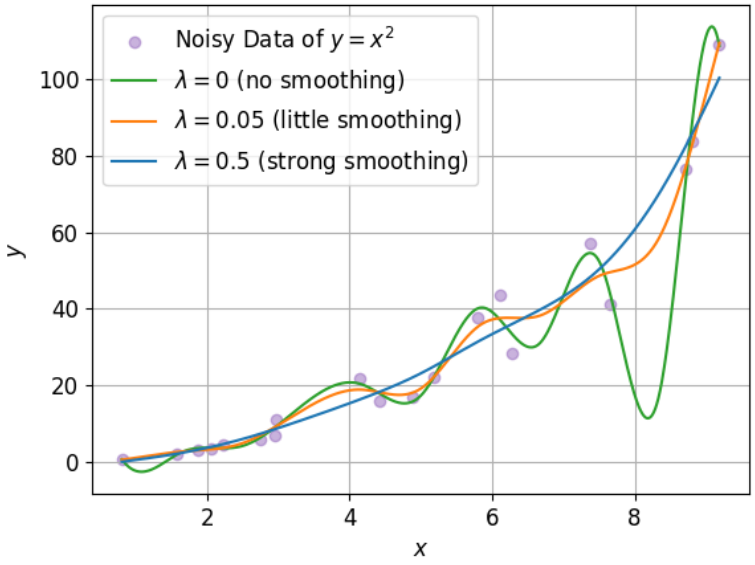


Figure 15: Comparison of different smoothing factors λ for a B-Spline approximation of the function $y = x^2$.

The new objective of the optimisation is to maximise the base model while minimising the smoothing penalty; thus, the equation 37 gets extended to

$$\underset{\theta_1,\ldots,\theta_m}{\arg\max} \underbrace{\sum_{n=1}^{N}\sum_{j=1}^{J} r_{n,j}\left(-\ln(2\pi) - \ln(\nu) - \frac{1}{2\nu}\|m_n - P_w\|^2\right)}_{\text{Base Model}} \underbrace{- \sum_{n=1}^{B-2}\sum_{j=1}^{J} ||P_w||^2}_{\text{Smoothing Penalty}}, \tag{57}$$

while the derivative of the objective function w.r.t. the parameters θ_l from equation 47 takes the form of

$$\frac{\partial f}{\partial \theta_l} = \underbrace{\sum_{j=1}^{J}\sum_{n=1}^{N} r_{n,j} \cdot \nu^{-1} \cdot (m_n - P_K)^T \cdot \frac{\partial P_K}{\partial h_j} \cdot \frac{\partial h_j}{\partial \theta_j}}_{\text{Base Model}} \underbrace{- \sum_{n=1}^{B-2}\sum_{j=1}^{J} 2 \cdot P_w^T \cdot \frac{\partial h_j}{\partial \theta_j}}_{\text{Smoothing Penalty}} \tag{58}$$

By incorporating the regularisation term, the optimisation process ensures that the reconstructed trajectory remains smooth and realistic, even when dealing with noisy data or inaccuracies in the initial predictions. This approach balances the fidelity to the data with the need for smoothness, leading to more stable and plausible trajectory reconstructions.

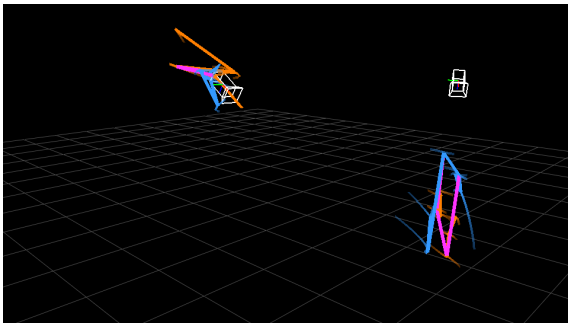
3.6.3 Mean Key Point Initialization

The performance and results of the EM algorithm are highly susceptible to the initialisation of parameters, especially when dealing with complex models or high-dimensional data. In particular, the EM algorithm's reliance on initial parameter values means that the quality of these starting points can significantly impact the algorithm's ability to converge to the global optimum.

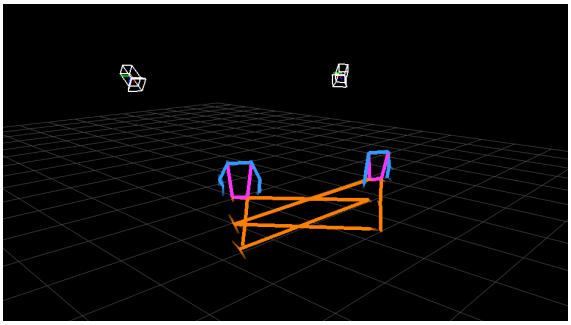
Initially, the parameters for a new hypothesis within the EM algorithm have been initialised randomly. While this approach is straightforward, it comes with substantial risks. Random initialisation can lead to several critical issues: the algorithm may become trapped in local minima, as illustrated in figure 16a, or it might incorrectly assign key points, leading to the phenomenon of "switched key points", where important data points are associated with the wrong hypotheses (see figure 16b). These problems, especially when combined, can severely degrade the performance of the EM algorithm, resulting in suboptimal or erroneous outcomes, as depicted in figure 16c.

To mitigate these challenges, a more structured initialisation strategy has been adopted. Instead of random initialisation, the parameters of the key points for any new hypothesis are now initialised by centring them around the mean of all existing key points and projectinng them a parameterised distance away from the camera. This new method leverages the existing data distribution to provide a more informed starting point, significantly reducing the likelihood of the algorithm getting stuck in local minima or incorrectly switching key points.

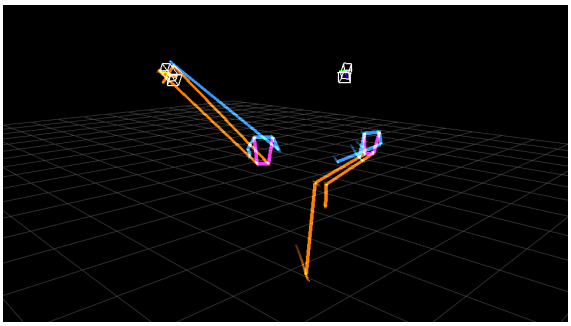
Empirical results demonstrate that this mean-based initialisation approach leads to a more stable and reliable performance of the EM algorithm. By anchoring the initial parameters closer to the actual data distribution, the algorithm is less prone to the issues associated with random initialisation, as evidenced by the improved outcomes shown in figure 16d. This improvement enhances the accuracy of the algorithm's results and increases the method's robustness in diverse scenarios.



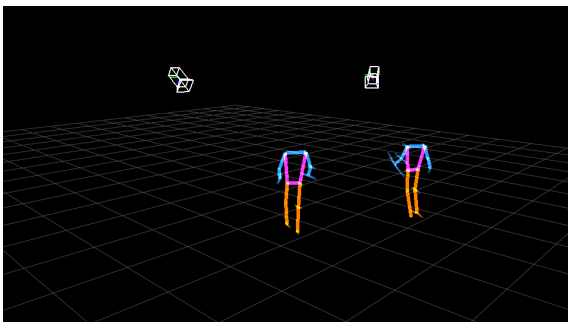
(a) EM stuck in local minima



(b) EM with switched key points



(c) EM with both problems



(d) EM with mean initialisation

Figure 16: Comparison between randomly initialised parameters and initialisation to the mean of the key points.

3.6.4 Key Point Support by Responsibility Analysis

One of the critical functions of the EM algorithm is to calculate the "responsibility" that each hypothesis bears for each key point. This responsibility is represented as a probability, indicating how likely it is that a given key point is associated with a particular hypothesis. By calculating these responsibilities, the algorithm can effectively distribute the observed data points across the different hypotheses, thus forming the basis for further probabilistic reasoning.

These responsibilities can then be leveraged to determine which key points are supported by which hypotheses. In this context, "supported" refers to the likelihood that a key point is associated with a specific hypothesis across multiple camera frames, exceeding a predefined threshold. If the likelihood that a key point belongs to a given hypothesis in at least a certain number of cameras surpasses this threshold, the key point is considered supported by that hypothesis. This probabilistic approach allows for a more nuanced understanding of the data, enabling the algorithm to make informed decisions about which key points are reliably associated with specific hypotheses.

The notion of "support" is instrumental in determining the limits of the algorithm's fitting capabilities. For instance, key points that cannot be fitted by the algorithm—meaning their likelihood of belonging to any hypothesis does not meet the required threshold—can be identified and

treated accordingly. This concept is further explored in Section 3.6.6 Managing Unsupported Hypothesis Key Points, where the handling of unsupported key points is discussed in detail.

Moreover, the same notion of support can be applied to hypothesis management within the algorithm. Hypotheses that fail to maintain sufficient support from key points across multiple frames may be deemed unnecessary and thus can be removed ("killed") from consideration. This process of hypothesis elimination is crucial for optimising the algorithm's performance by focusing computational resources on more promising hypotheses. This topic is further elaborated in Section 3.6.7 Better Hypothesis Management, where strategies for effective hypothesis management are presented.

Using the EM algorithm's calculated responsibilities and the support concept, the overall process becomes more robust and efficient, leading to better hypothesis estimation and, ultimately, more accurate model fitting.

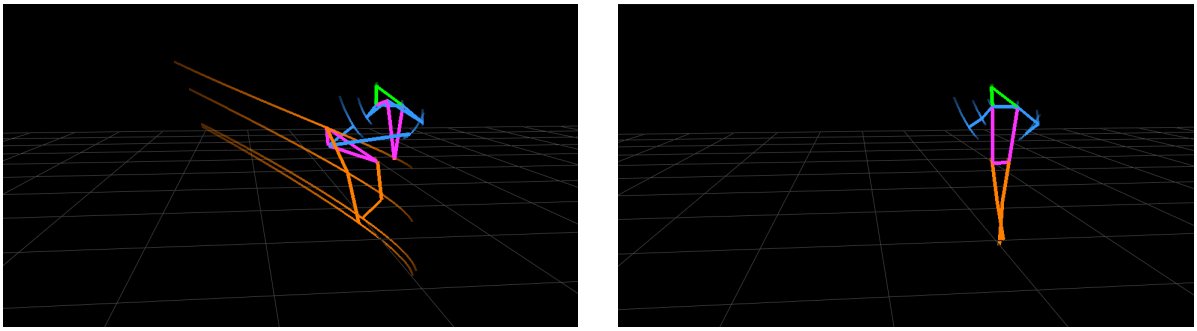
3.6.5 Prefiltering of Key Points

In detecting key points using the YOLOv7 network, it is observed that not all key points are accurate. As demonstrated in figure 17, some key points detected by the network may be misplaced or incorrectly identified, leading to errors in subsequent steps. These inaccuracies can significantly distort the hypotheses generated during the reconstruction process, as illustrated in figure 18a, where the resulting 3D skeleton reconstruction is visibly compromised.

The output of the YOLOv7 network is a set of key points for each detected person consisting of the key point's coordinates and a confidence score. Using this confidence score, a prefiltering step is introduced to mitigate the impact of erroneous key points before the detected key points are utilised in the algorithm. This prefiltering process involves setting a threshold for the minimum number of key points with a confidence score above a certain level, as determined by the YOLOv7 network. Specifically, only key points that exceed this confidence threshold are retained for further processing. Moreover, if the number of valid key points in a set does not meet the minimum threshold, all key points in that set are discarded.

This prefiltering approach eliminates unreliable data, resulting in a more stable fitting of the key points during the 3D skeleton reconstruction process. The benefits of this method are evident in figure 18b, where the prefiltered key points lead to a more accurate and stable reconstruction of the 3D skeleton. By discarding low-confidence key point sets, the overall robustness and accuracy of the algorithm are significantly improved, reducing the likelihood of distortions in the final output.

Figure 17: Detected key points by YOLOv7 with stray key points on the lower left chair.



(a) Without key point prefiltering.

(b) With key point prefiltering.

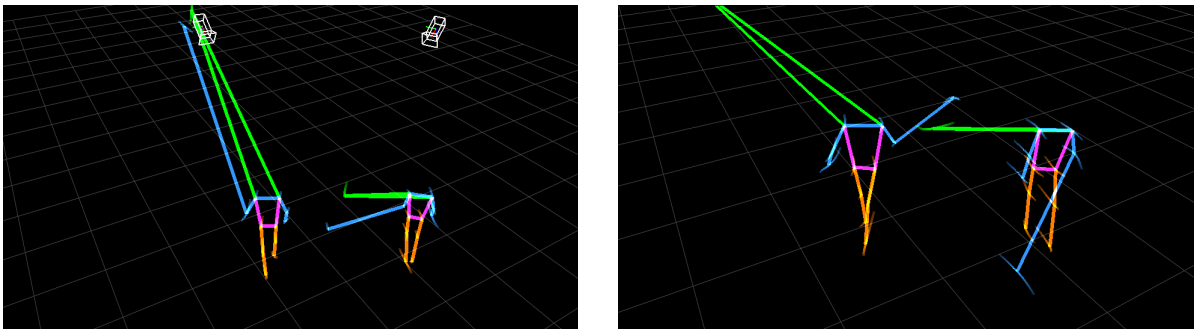
Figure 18: Comparison of 3D reconstruction effect of key point prefiltering.

3.6.6 Managing Unsupported Hypothesis Key Points

The concept of support (described in section 3.6.4 Key Point Support by Responsibility Analysis) plays a crucial role in refining the algorithm for fitting key points in a 3D skeleton reconstruction. Initially, the algorithm was designed to fit all key points, regardless of whether they had sufficient support from any hypothesis. This approach often led to suboptimal outcomes, as unsupported key points could cause the algorithm to become trapped in local minima or result in erratic switching of key points, a phenomenon thoroughly discussed in section 3.6.3 Mean Key Point Initialization.

To mitigate these issues, the algorithm was modified to avoid fitting key points that lacked support from any hypothesis. However, this modification introduced a new challenge: once ignored by the algorithm, unsupported key points could become effectively "stuck in the air". Since these points were not being fitted, their support would never increase, especially in dynamic scenarios

where the subject moves away from the initial position of the key point, as illustrated in figure 19. This phenomenon resulted in a persistent mismatch between the key points and the underlying skeletal structure.

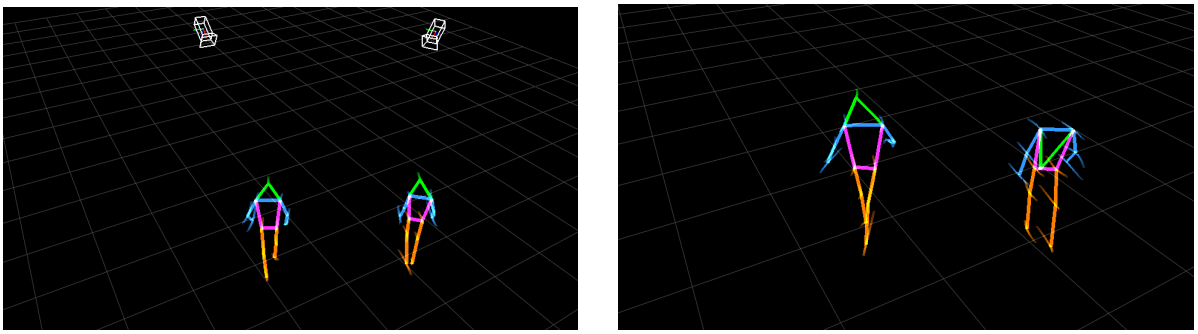


(a) Right at the beginning of the video sequence.

(b) Couple of frames later.

Figure 19: Result of not fitting unsupported key points.

A more nuanced approach was implemented to address this issue. While the algorithm still refrains from fitting unsupported key points directly, it now carries the mean position of the fitted key points as a parameter to the unsupported hypothesis. This adjustment allows the unsupported key point to get "dragged along" with the movement of the supported structure, gradually increasing its support until the algorithm can reliably fit it. The result is a more stable and accurate reconstruction of the 3D skeleton, as demonstrated in figure 20. Figure 20b shows that even though the nose key point remains unsupported and is not directly fitted, the mean position of the surrounding fitted key points is used to guide its eventual stabilisation. This method significantly improves the overall coherence and stability of the skeleton fitting process, ensuring that key points remain aligned with the dynamic movements of the subject.



(a) Right at the beginning of the video sequence.

(b) Couple of frames later.

Figure 20: Result after dragging along the mean of the unsupported key points.

3.6.7 Better Hypothesis Management

In the early stages of the algorithm's development, it was required that the number of people present in the scene was entered manually. While functional, this manual process was limited and impractical for dynamic environments where the number of people could change frequently. The need for a more sophisticated approach became evident, leading to the development of an automated hypothesis management system that could accurately estimate the number of people in the scene based on detected key points.

The central idea behind this improved hypothesis management is to enable the algorithm to automatically infer the number of people present at any given time, eliminating the need for manual input. The algorithm achieves this by analysing the key points detected across multiple frames and cameras. Given that the number of people in a scene can fluctuate—due to individuals entering or leaving the scene or due to inaccuracies in the YOLOv7 detection system, such as failing to detect a person or mistakenly identifying an object as a person—the hypothesis management system needed to be both adaptive and robust.

The initial strategy for estimating the number of people was to take the maximum number of detections from any single camera in the scene. This approach effectively ensured that new people were quickly recognised and added to the scene, a process referred to as "birthing". However, this rapid "birthing" of new hypotheses also introduced potential issues, particularly creating hypotheses for people who did not exist.

To counterbalance this, the algorithm was designed with a mechanism to quickly "kill" any hypotheses not sufficiently supported by the detected key points. Specifically, a hypothesis would be terminated if the number of key points associated with it fell below a certain threshold. This threshold acts as a filter, ensuring that only hypotheses with strong evidence — i.e., a sufficient number of key points — are maintained. At the same time, those based on weak or spurious data are discarded.

This improved hypothesis management system has significantly enhanced the algorithm's performance. By automating the estimation of the number of people in the scene and dynamically adjusting hypotheses based on the strength of detected key points, the algorithm is now better equipped to handle the complexities of real-world scenarios.

3.7 Evaluation of Robustness, Accuracy and Speed

A comprehensive series of tests will be conducted to evaluate the robustness and accuracy of the MC3D-TRECSIM algorithm thoroughly. These tests will assess the algorithm's performance under typical conditions and challenging edge cases. The tests are designed to identify the algorithm's strengths and weaknesses, providing a clear understanding of its capabilities and areas that may require further improvement.

3.7.1 General Robustness Testing

The first set of tests will focus on evaluating the general robustness of the algorithm under typical conditions. These conditions include scenarios where:

1. **Individuals are well-recognised by the YOLOv7 algorithm**: The accuracy of the algorithm will be tested when YOLOv7 reliably detects and identifies key points in individuals.

2. **People enter the scene from positions not directly in front of the cameras**: The algorithm will be tested to see how well it reconstructs individuals who enter from various angles, ensuring that initial detection and tracking are accurate across different entry points.

3. **People do not cross paths**: Scenarios will be set up where individuals move independently without intersecting paths, allowing the algorithm to demonstrate its ability to track each person without interference accurately.

4. **People do not adopt uncommon poses**: The algorithm's ability to maintain accurate reconstructions during typical movements, such as walking, turning around, or raising arms, will be tested.

These tests will involve visual inspections and quantitative analysis, including comparing limb length estimations to ground-truth measurements. The algorithm's performance under these conditions will provide a baseline for its reliability in standard environments.

3.7.2 Limb Length Estimation Accuracy and Consistency

The second set of tests was specifically targeting the accuracy of limb length estimations, a critical aspect of the algorithm's functionality. These tests will measure:

- **Mean accuracy**: The average estimated limb lengths over a whole video sequence will be compared to the real world measurements, with particular attention to how closely the algorithm approximates the actual measurements.

- **Standard deviation**: The consistency of limb length estimates will be analysed by calculating the standard deviation across multiple measurements.

Special attention was given to cases where discrepancies are expected, such as with specific subjects, where YOLOv7 has shown inaccuracies. The tests identified which aspects of the algorithm may need refinement by comparing the results across different body measurements. The limbs and body lengths which have been compared are defined in figure 21 and the accompanying table 1.

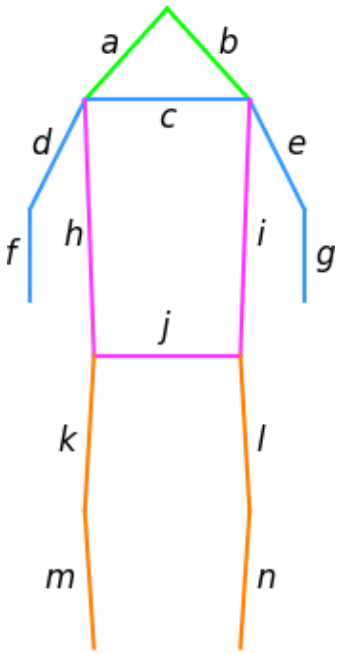


Figure 21: Limbs

a	Nose - Right Shoulder
b	Nose - Left Shoulder
c	Shoulder Line
d	Right Brachium
e	Left Brachium
f	Right Antebrachium
g	Left Antebrachium
h	Right Flank
i	Left Flank
j	Waist
k	Right Thigh
l	Left Thigh
m	Right Shin
n	Left Shin

Table 1: Limb names

To see how the approximated limb lengths vary over time, they will be primarily plotted in a graph as seen in figure 22, where each colour represents a different hypothesis, and the approximated limb lengths are plotted against the frame number. If available, the approximated limb lengths will be compared to the measurements. These plots will be zoomed in on the relevant frames, and outliers will be hidden.

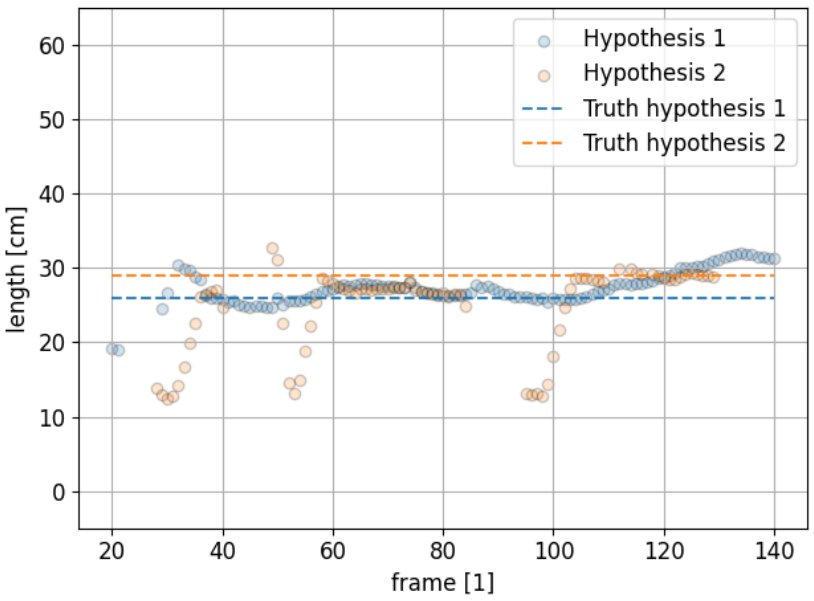
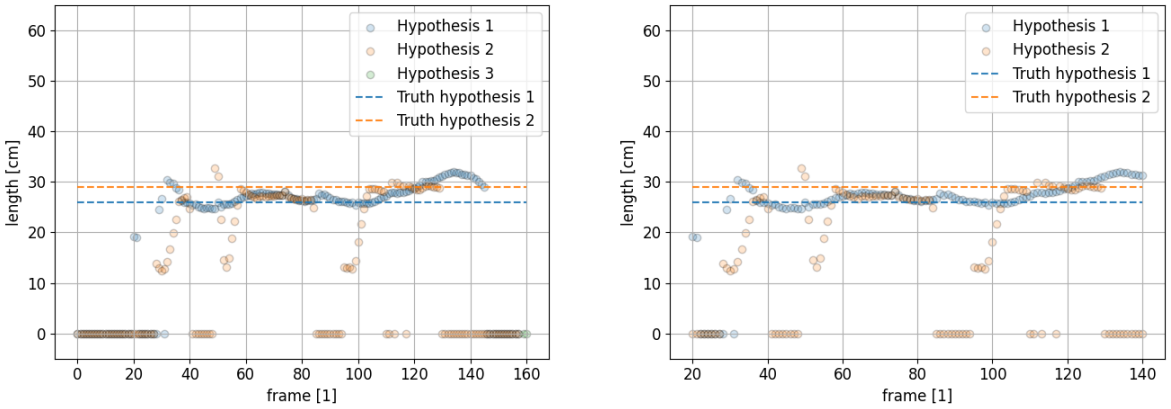


Figure 22: Comparison of the approximated and the measured length of the line between the nose and the right shoulder (line a in figure 21) for two people excluding unsupported key points.

If plots with outliers or not zoomed-in data exist (as seen in figure 23), they will be mentioned in the text and shown in the appendix.



(a) Plot over the whole video sequence.

(b) Plot zoomed in on relevant frames (20 - 140).

Figure 23: Comparison of the approximated and the measured length of the line between the nose and the right shoulder for two people, including unsupported key points.

3.7.3 Edge Case Testing

A series of edge case tests will be conducted to challenge the algorithm further. These scenarios are designed to push the algorithm beyond typical conditions and explore its limitations:

1. **People crossing paths**: The algorithm's ability to differentiate and accurately track individuals as they intersect will be tested. This will help identify potential issues with key point assignment and reconstruction accuracy during complex interactions.

2. **Multiple people entering the scene close to the cameras**: The tests will examine how the algorithm handles individuals entering from unexpected locations, particularly close to the cameras, where initial detection and positioning may be more challenging.

3. **Uncommon body orientations**: The algorithm will be tested on its ability to accurately reconstruct individuals in less common postures, such as lying down. This will include assessing the accuracy of limb length measurements in these positions and identifying any misalignments or errors caused by the YOLOv7 algorithm.

Each edge case will involve both qualitative visual assessments and quantitative measurements, such as analysing the mean and standard deviation of limb length estimations. The results from these tests will help pinpoint specific scenarios where the algorithm struggles, providing a roadmap for future improvements.

3.7.4 Speed Analysis

In addition to robustness and accuracy, the speed of the MC3D-TRECSIM algorithm is a critical factor, especially for real-time applications. Speed analysis will evaluate the algorithm's computational efficiency and ability to maintain performance under varying conditions. The following aspects will be analysed:

- **Frame processing time**: The average time to process each frame will be measured across different test scenarios. This metric will help determine if the algorithm can operate in real-time environments and where optimisations may be needed.

- **Scalability with multiple individuals**: The algorithm's performance will be tested with varying numbers of people in the scene. The goal is to assess how the processing time scales as the number of detected individuals increases, which is crucial for environments with high foot traffic.

The speed analysis will involve time-based metrics and comparing the algorithm's performance under different conditions. By analysing the time taken for each algorithm step, from detection to reconstruction, areas that require optimisation can be identified. This evaluation will ensure that the MC3D-TRECSIM algorithm efficiently provides accurate and robust results, making it

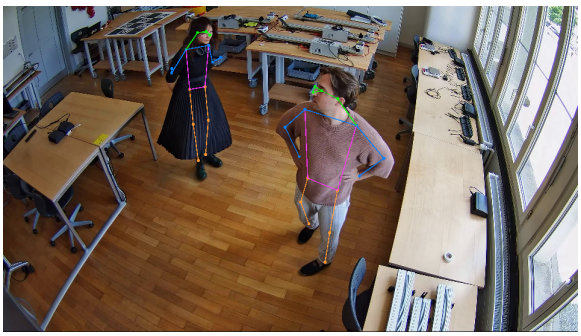
suitable for real-time applications. All speed tests are going to be conducted on the computer in the laboratory (see section 3.3 Laboratory Setup) using one thread.

4 Results

This section presents how well the MC3D-TRECSIM algorithm is able to reconstruct individuals in different cases. The evaluation is divided into four main sections: general robustness, limb length estimations, edge cases, and speed analysis. The general robustness section assesses the algorithm's performance under typical conditions, such as tracking individuals entering the scene, raising their arms, and turning around. The Limb Length Estimations Accuracy and Consistency section examines the accuracy and consistency of the algorithm's limb length estimations, comparing the approximated limb lengths to the measured lengths. The edge cases section examines scenarios where the algorithm struggles to track individuals accurately due to various factors, such as occlusions, body orientation, and people crossing paths. And the speed analysis section evaluates the algorithm's computational efficiency and real-time performance.

4.1 General Robustness

Assessing the accuracy and robustness of the algorithm through visual inspection reveals reliable performance under typical conditions. These typical conditions, as illustrated in figures 24 and 25, include scenarios where individuals are well-recognized by the YOLOv7 algorithm, enter the scene from positions not directly in front of the cameras, do not cross paths, and avoid uncommon poses. In such scenarios, the algorithm demonstrates a solid capability to accurately track and reconstruct individuals in 3D space.



(a) Frame of camera 1.

(b) Frame of camera 2.

Figure 24: Frames of two people.

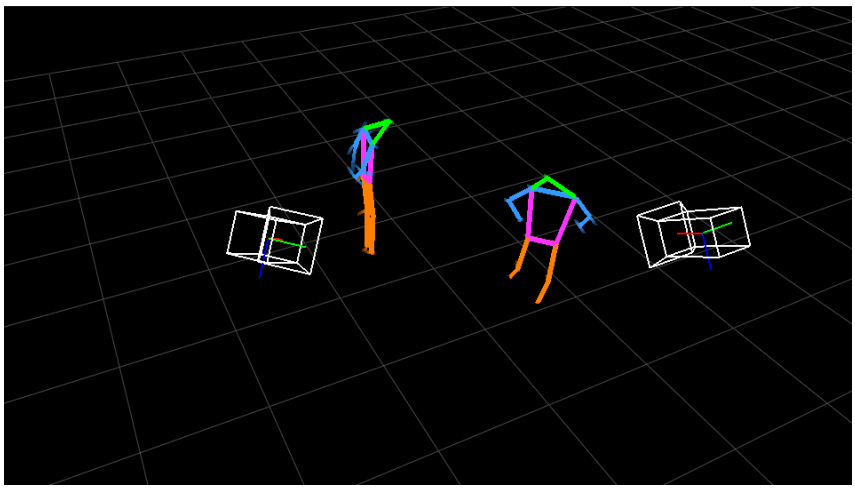
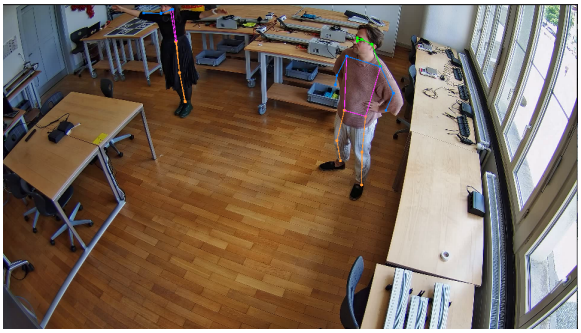


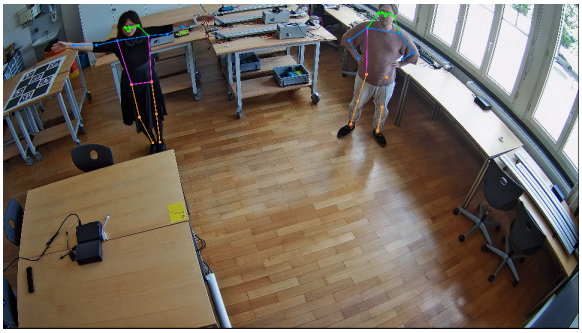
Figure 25: 3D reconstruction of two people.

Under these conditions, the algorithm effectively handles even relatively complex movements, such as raising arms or turning around. The consistent performance in these cases highlights the algorithm's reliability in standard operational environments. The ability to maintain accuracy during these actions suggests a level of robustness essential for practical applications where various human movements are expected. Examples for these cases can be found in the appendix C Plots For Visual Robustness Inspection and Plots For Visual Robustness Inspection.

However, it is essential to note that some glitches in the 3D reconstruction can occur, primarily due to inaccuracies in the YOLOv7 algorithm's initial detections. Figures 26 and 27 show such a glitch, where the person's right arm to the left gets elongated, and the person appears to be two-dimensional and mostly appear due to inconsistencies in the 2D pose estimation. These glitches are generally minor and do not significantly detract from the system's overall performance. They serve as a reminder of the dependency on the underlying detection algorithm, where any inaccuracies in key point identification can propagate through the reconstruction process and should be appropriately dealt with.



(a) Frame of camera 1.



(b) Frame of camera 2.

Figure 26: Glitchy frames of two people.

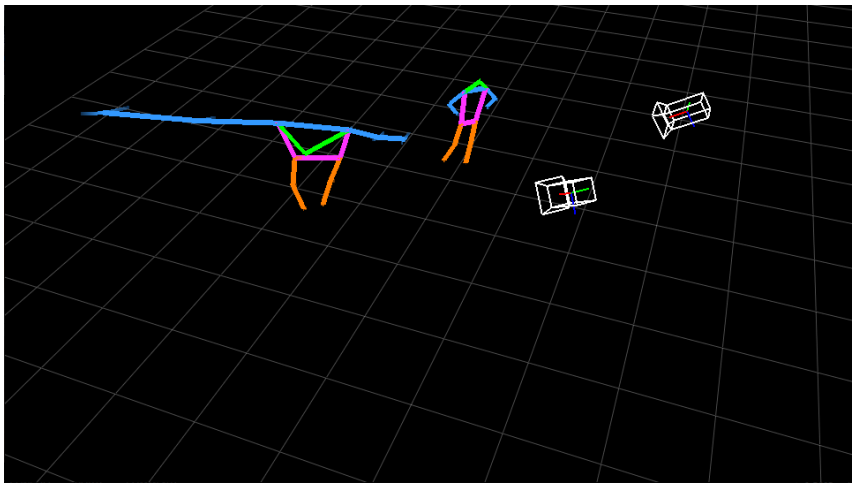
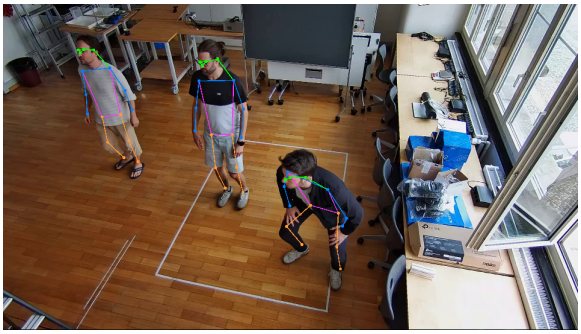
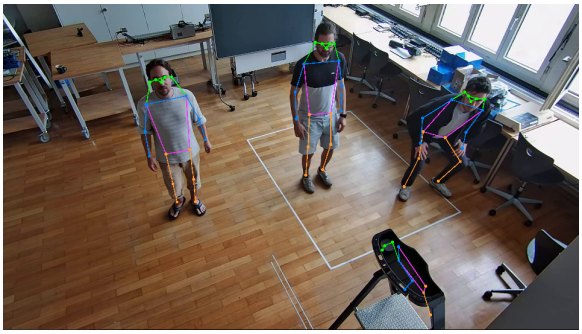


Figure 27: Glitchy 3D reconstruction of two people.

The algorithm demonstrates robust performance under typical conditions when three people enter the scene simultaneously. This is evident in the accurate 2D key point detections shown in figure 28 and the corresponding 3D reconstruction in figure 29. The algorithm can successfully handle multiple individuals, maintaining stable tracking and reconstruction across frames even if some stray key points have been detected by the YOLOv7 network (as seen in the lower right part of the frame in figure 28b).



(a) Frame of camera 1.



(b) Frame of camera 2.

Figure 28: Frames of three people.

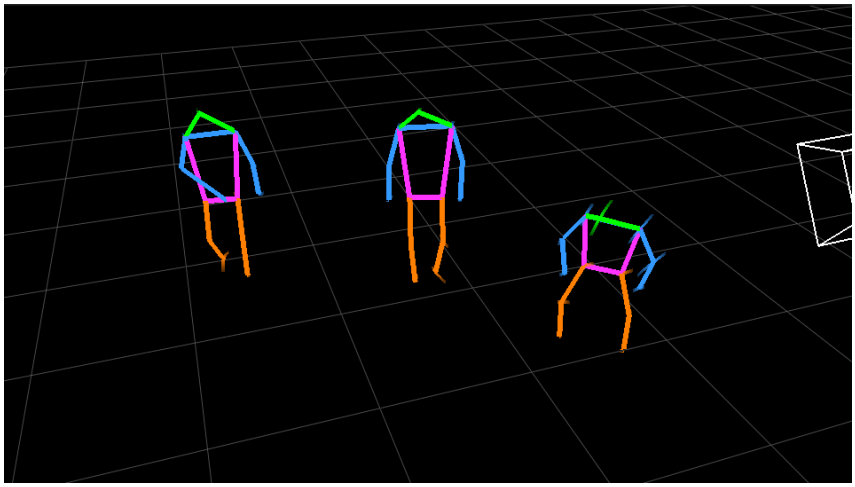
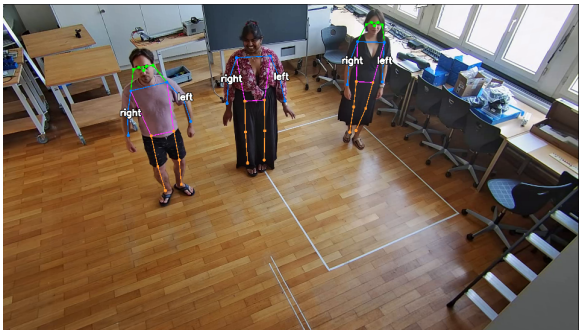


Figure 29: 3D reconstruction three people.

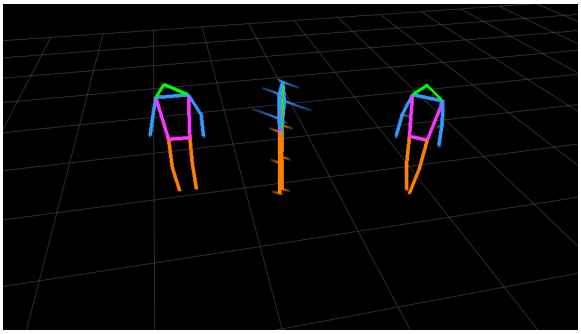
However, the algorithm's reliability heavily depends on the precise identification of key points by the YOLOv7 network. When YOLOv7 fails to accurately recognize key points, the quality of the 3D reconstruction deteriorates significantly. This issue is particularly noticeable in figures 30 and 31, where an error in key point detection causes a mix-up between the left and right sides of the person in the centre of the scene, especially in the frame from camera 1. This misidentification leads to a distorted 3D reconstruction, highlighting the critical role of accurate key point detection for the overall success of the algorithm.

(a) Correct detections on frame of camera 1.

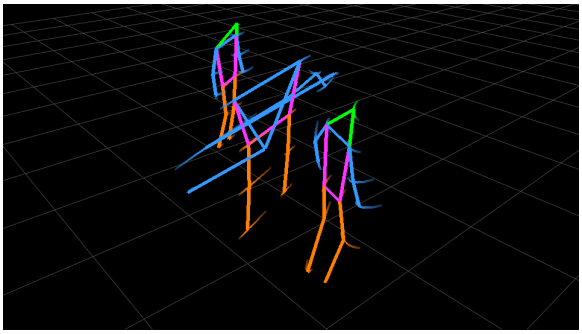


(b) Faulty detections on frame of camera 1.

Figure 30: Frames of three people with annotated sides detected by the YOLOv7 network.



(a) Center view



(b) Side view

Figure 31: Distorted 3D reconstruction of centre person because of faulty key point detection.

A more extended sequence video example can be found on the project's GitHub [20]. Additionally, more detailed examples and analyses are provided in the appendix C Plots For Visual Robustness Inspection. These additional examples reinforce the algorithm's overall strength while highlighting areas where future improvements could be made.

4.2 Limb Length Estimation Accuracy and Consistency

Besides the overall robustness, the accuracy of the reconstructed limbs is also important. Table 2 shows, that the algorithm's limb length estimations are generally quite accurate, demonstrating a strong correlation with the ground-truth measurements. On average, the approximated limb lengths closely align with the actual measurements, and the standard deviation across these estimations is notably low. This indicates a consistent performance of the algorithm in estimating limb lengths, with most approximations falling within a narrow margin of error. However, there are some exceptions. Particularly the shoulder line and the antebrachia which show a high deviation from actual measurement in subject 1 and subject 2. These deviations are likely due to

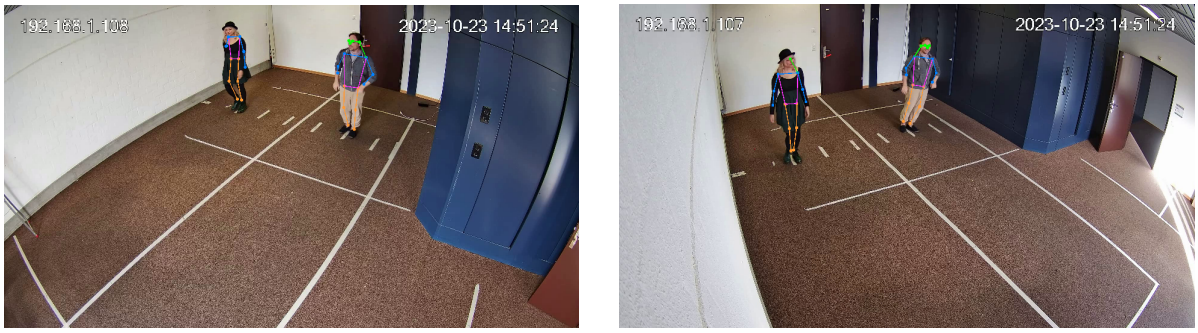
inaccuracies in the YOLOv7 algorithm's key point detection, as shown in figure 32b where the antebrachia of subject 1 (right) are detected to be shorter than they really are.

	Subject 1			**Subject 2**		
Limb	*Truth*	*Mean* $\pm$ *Std.*	Δ	*Truth*	*Mean* $\pm$ *Std.*	Δ
Nose-R. Shoulder	29	27.49 ± 1.96	-1.51	26	25.67 ± 4.98	-0.33
Nose-L. Shoulder	28	28.09 ± 2.15	$+0.09$	26	25.26 ± 3.29	-0.74
Shoulder Line	40	33.34 ± 0.57	-6.66	33	29.38 ± 4.39	-3.62
R. Brachium	27	25.54 ± 1.10	-1.46	30	24.34 ± 2.19	-5.66
L. Brachium	27	25.42 ± 1.81	-1.58	31	24.49 ± 2.01	-6.51
R. Antebrachium	27	19.59 ± 2.06	-7.41	30	21.39 ± 2.88	-8.61
L. Antebrachium	27	18.18 ± 2.20	-8.82	30	21.75 ± 2.96	-8.25
R. Flank	40	48.66 ± 0.79	$+8.66$	38	40.50 ± 1.37	$+2.50$
L. Flank	40	48.30 ± 1.21	$+8.30$	39	41.75 ± 1.56	$+2.75$
Waist	32	21.24 ± 0.55	-10.76	30	19.28 ± 3.04	-10.72
R. Thigh	47	35.83 ± 1.31	-11.17	50	37.31 ± 1.08	-12.69
L. Thigh	47	36.54 ± 1.52	-10.46	47	36.81 ± 1.06	-10.19
R. Shin	42	37.96 ± 1.52	-4.04	46	43.16 ± 2.62	-2.84
L. Shin	42	37.10 ± 2.29	-4.90	43	42.78 ± 1.77	-0.22

Table 2: Comparison of the measured and the approximated lengths (in cm) for the limbs of two subjects.

Also, the estimations for the flanks, waist, and thighs, tend to be less accurate than other limbs with very high deviations from actual measurements. The thighs, in particular, are estimated to be larger, while the flanks are estimated to be smaller than the actual measurements. This variation could be due to the YOLOv7 algorithm predicting the waist position differently from how it was measured in real life. The misalignment in waist detection has a cascading effect on the accuracy of related limb measurements, leading to these specific inaccuracies. Despite these issues, the algorithm's overall performance in limb length estimation remains strong, with only a few areas needing further refinement.

Figure 32 also shows that the YOLOv7 algorithm struggles to identify parts around the face of subject 2 (left) and later in the video sequence also flips subject 2 around in some frames. This leads to the overall accuracy and consistency of the reconstruction to be a bit better for subject 1 than subject 2. The fluctuating limb length estimations of the shoulder line shown in figure 33 also indicate a possible problem with YOLOv7 identifying subject 2.



(a) Right ear, left ear, nose and right hand of left subject not recognized.

(b) Right ear and nose of left subject not recognized.

Figure 32: Faulty key point detection by YOLOv7.

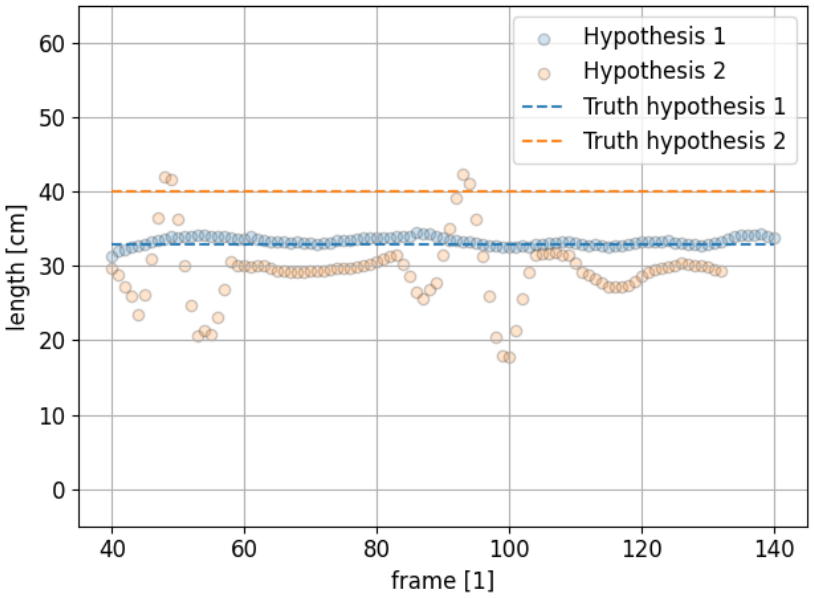
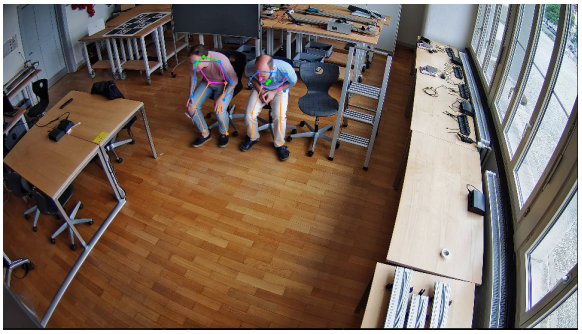


Figure 33: Estimated length of shoulder line over video sequence.

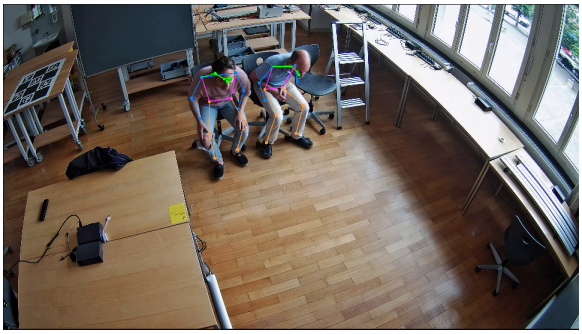
More limb length comparison plots across this sequence and a table containing the measurement comparisons of a fit with less smoothing can be found in the appendix D.1 Limb Length Estimation Accuracy and Consistency.

4.2.1 Limb Length Consistency while Standing Up

Analyzing the sequence of two people standing up, as shown in figures 34 and 35, reveals some notable inaccuracies in the 3D reconstruction, indicating challenges that the algorithm faces in accurately tracking individuals during this transition.



(a) Frame of camera 1.



(b) Frame of camera 2.

Figure 34: Frames of two people standing up.

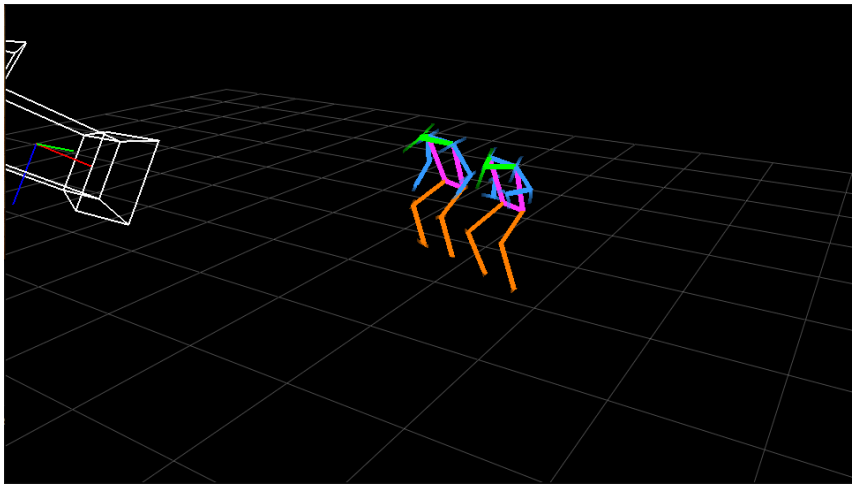


Figure 35: 3D reconstruction of two people standing up.

The algorithm's performance declined in this scenario, as evidenced by an increased standard deviation in the limb length measurements, detailed in table 3. This increase in variability suggests that the algorithm struggles to maintain consistency in limb length estimation when individuals move from a seated to a standing position.

	Subject 1	**Subject 2**
Limb	*Mean* $\pm$ *Std.*	*Mean* $\pm$ *Std.*
Nose-R. Shoulder	24.09 ± 1.68	27.21 ± 3.21
Nose-L. Shoulder	26.89 ± 3.49	28.92 ± 1.50
Shoulder Line	31.74 ± 1.96	32.20 ± 1.18
R. Brachium	29.60 ± 3.30	27.35 ± 3.62
L. Brachium	27.90 ± 1.89	26.54 ± 1.35
R. Antebrachium	20.86 ± 4.38	21.46 ± 1.85
L. Antebrachium	24.28 ± 1.76	22.44 ± 2.65
R. Flank	48.83 ± 5.69	46.33 ± 5.33
L. Flank	49.19 ± 6.51	47.49 ± 4.79
Waist	21.60 ± 1.16	21.30 ± 1.04
R. Thigh	38.53 ± 4.82	36.23 ± 4.61
L. Thigh	37.96 ± 5.03	34.80 ± 4.54
R. Shin	45.37 ± 3.42	36.70 ± 4.21
L. Shin	43.57 ± 4.81	39.56 ± 3.73

Table 3: Limb length estimations for people standing up.

One specific issue highlighted in figure 36 is the apparent increase in the length of the right flank as the people stand up. This unexpected change in limb length is puzzling and points to a potential flaw in the reconstruction process. The exact reasons behind this phenomenon are not entirely clear. However, it is plausible that inaccuracies in the YOLOv7 algorithm - particularly in detecting and tracking body parts during the dynamic motion of standing up - may be contributing to these errors.

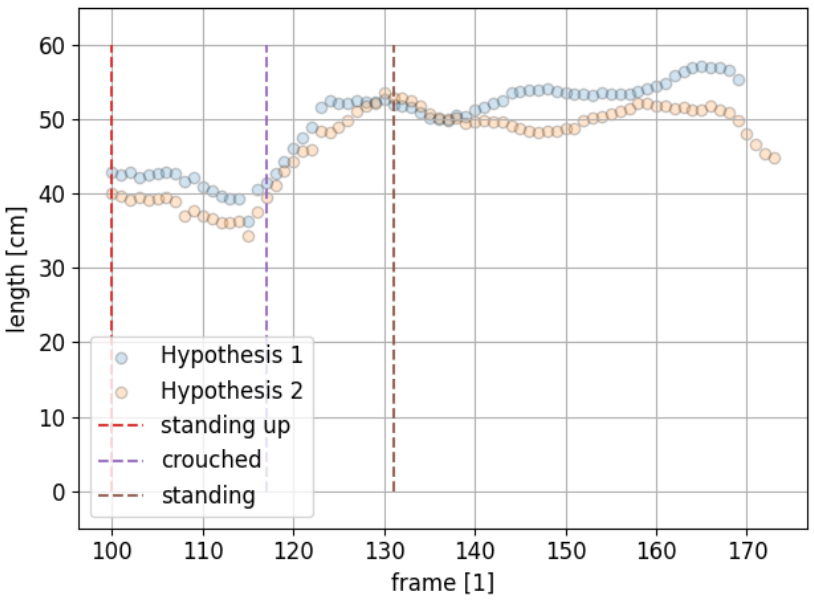


Figure 36: Approximated limb length of right flank over the whole sequence of two people standing up.

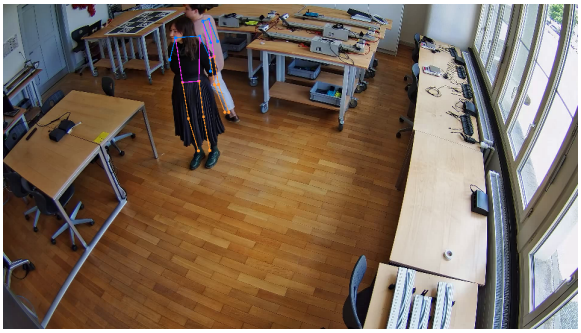
Further analysis of limb length measurements across this sequence can be found in the appendix D.2 Limb Length Comparison Plots Of Standing Up. Moreover, more frames from the sequence, which offer a visual representation of the reconstruction process during the standing-up motion, are available in the appendix E.1 Reconstruction of Sequence of People Standing Up.

4.3 Edge Cases

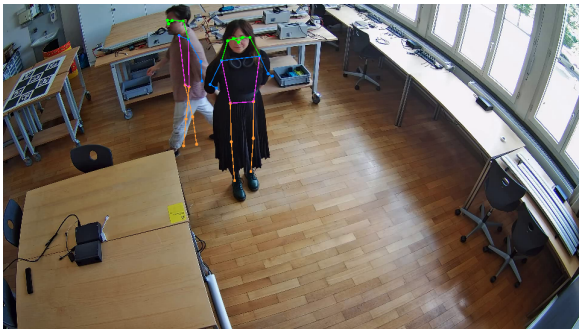
In this section, some edge cases demonstrate the limitations of the current implementation of MC3D-TRECSIM. These cases highlight scenarios in which the algorithm struggles to track individuals accurately due to various factors, such as occlusions and body orientation.

4.3.1 People Crossing Paths

When two people cross paths, the reconstruction quality significantly diminishes, presenting a clear challenge for the algorithm. The primary issue arises from the algorithm's difficulty differentiating between the two individuals as they intersect. During this interaction, both hypotheses have high responsibilities for the key points identified by YOLOv7, which confuses the system. As a result, the algorithm struggles to assign the correct key points to the corresponding individual, leading to distorted and inaccurate reconstructions (see figures 37 and 38).



(a) Frame of camera 1.



(b) Frame of camera 2.

Figure 37: Camera frames of two people crossing paths.

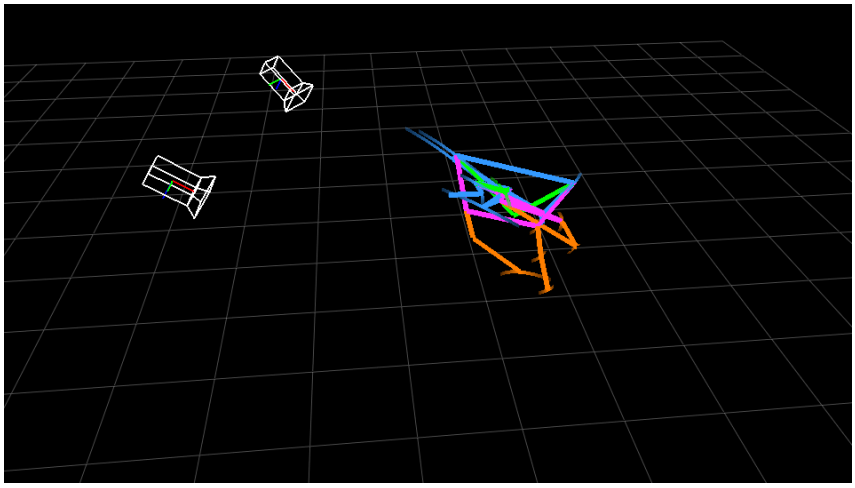


Figure 38: 3D reconstruction of two people crossing paths.

The problem persists even after the individuals have crossed paths. The algorithm's confusion during the intersection often carries over, resulting in continued poor reconstruction quality (more frames of the sequence can be found in appendix E.2 Reconstruction of Sequence of People Crossing Paths). The aftermath of the crossing event leaves the system in a state where it is unable to accurately re-establish the correct identities and positions of the individuals, further compounding the errors initiated during the crossing.

Interestingly, the situation only marginally improves when the individuals are far apart or when one hypothesis, representing one of the individuals, is removed from the analysis. However, these conditions are not always feasible in practical scenarios, where people may frequently cross paths at varying distances. Therefore, while distance or hypothesis removal can somewhat mitigate the issue, they do not fully resolve the inherent challenges posed by such close interactions. This highlights the need for further refinement in the algorithm's ability to manage key point

assignments and identity tracking during and after path-crossing events. One possible solution could be to implement a more sophisticated base model where the length of the limbs is taken into account to differentiate between individuals.

4.3.2 People Entering the Scene Close to The Cameras

When multiple people enter the scene close to the cameras, the reconstruction quality significantly deteriorates. This issue arises primarily because the initialisation process is based on the assumption that people will enter from the back of the room, rather than from a position close to the cameras. This assumption leads to incorrect initial positioning of individuals in the scene, affecting the accuracy of their subsequent tracking and reconstruction.

The initialisation parameters, critical for accurate reconstruction, are calculated under the presumption that the person is standing several meters from the cameras. This approach, detailed in section 3.6.3 Mean Key Point Initialization, works well when the subjects enter the scene from a distance, aligning with the expected initialisation conditions. However, when people enter the scene closer to the cameras, this assumption fails, resulting in poor alignment and inaccurate reconstructions.

The discrepancy between the assumed entry point and the actual entry point causes the system to misinterpret individuals' spatial positioning, leading to errors in their 3D reconstruction. These errors are particularly evident when multiple people enter the scene simultaneously, worsening the problem and making it difficult for the system to differentiate and track each person accurately.

Figures 39 and 40 show the frames and 3D reconstruction of two people reentering the scene. The algorithm struggles to accurately track the individuals due to the proximity to the cameras, resulting in distorted and inaccurate reconstructions.

(a) Reentering camera 1.

(b) Reentering camera 2.

Figure 39: Frames of two people entering the scene close to the cameras.

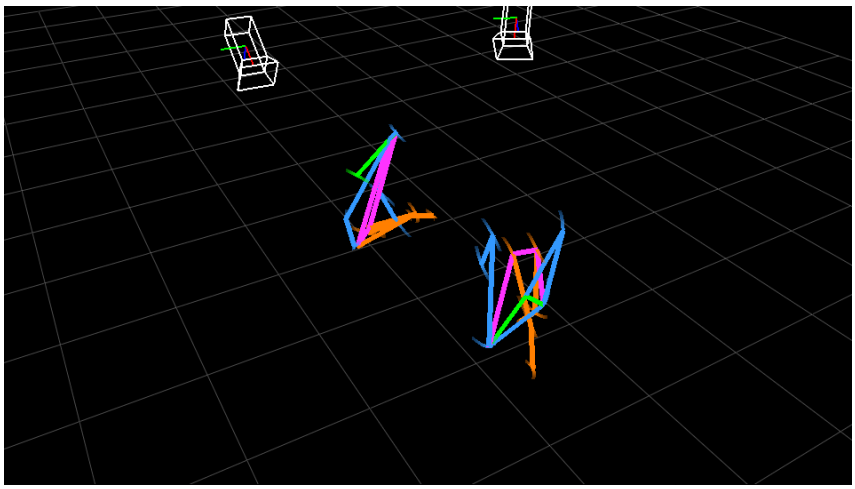
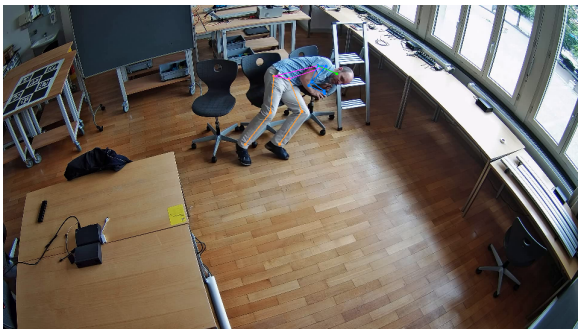


Figure 40: 3D reconstruction of two people entering the scene close to the cameras.

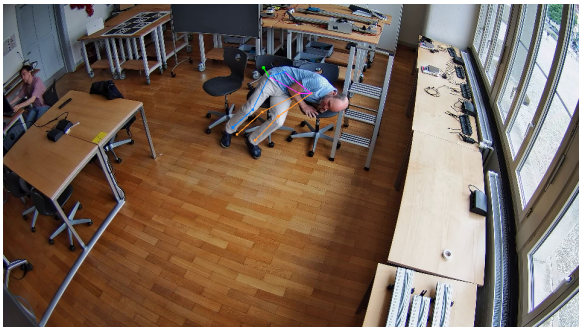
4.3.3 Uncommon Body Orientations

The results of measuring the limbs of a person lying down have proven to be significantly less accurate than those obtained when the person is standing. This discrepancy highlights a limitation in the current implementation of YOLOv7, which struggles with accurately recognizing limbs when the subject is not upright.

One of the primary issues observed is that YOLOv7 tends to incorrectly switch the positions of the head and hips while keeping the feet in the same position. The figures 41a and 41b illustrate this problem. They show the frame of the subject lying down on the two cameras. YOLOv7 does recognize the person and its position correctly in figure 41a but struggles with estimating the correct position on the second camera frame (figure 41b).



(a) Camera Frame 01.



(b) Camera Frame 01.

Figure 41: Frames of subject lying on chairs.

This misidentification disrupts the algorithm's ability to correctly interpret the body's orientation, leading to errors in limb detection and, consequently, inaccurate limb length measurements. Figure 42 shows how MC3D-TRECSIM distorts a person's limbs while lying down because of the faulty detection of the YOLOv7. The algorithm is unable to accurately track the limbs, as the head and hips are misaligned.

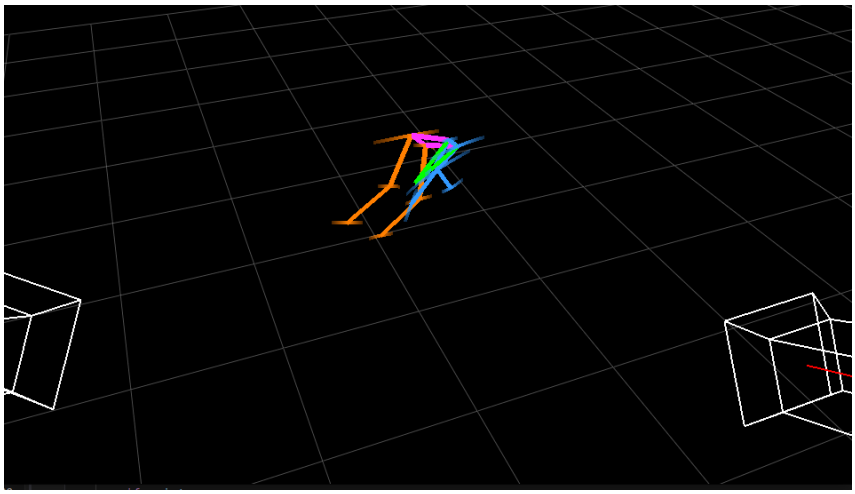


Figure 42: Two people standing up.

This misalignment in recognition causes the algorithm to be "thrown off", as it cannot reliably track and measure the limbs in this altered position. The impact of these errors is quantifiable, as shown by the length standard deviation in the limb length measurements (see table 4). The elevated variability indicates that YOLOv7's current configuration is insufficient for accurate limb measurement in non-standard postures, mainly when the subject is lying down. This finding suggests a need for additional training on diverse body orientations or potential adjustments to the algorithm to accommodate different postures better.

Limb	*Mean* $\pm$ *Std.*
Nose-R. Shoulder	25.45 ± 2.76
Nose-L. Shoulder	24.26 ± 7.22
Shoulder Line	27.96 ± 7.18
R. Brachium	27.24 ± 2.41
L. Brachium	28.23 ± 3.27
R. Antebrachium	23.18 ± 4.04
L. Antebrachium	21.04 ± 4.19
R. Flank	49.63 ± 9.92
L. Flank	47.92 ± 6.37
Waist	22.14 ± 3.50
R. Thigh	37.84 ± 2.99
L. Thigh	39.02 ± 4.46
R. Shin	46.11 ± 1.91
L. Shin	43.84 ± 3.17

Table 4: Results of the limb length measurements for a person lying down.

Figure 43 shows the nose - left shoulder line length estimations for the subject lying down over the whole video sequence. The red dotted line marks the frame at which the person starts lying down. The purple dotted line marks the frame, at which YOLOv7 misoriented the person. The brown dotted line marks the frame at which the person is done sitting up again. More measurements can be found in the appendix D.3 Limb Length Comparison Plots Of Lying Down. The unsupported key points are included to show how the key points lose their support as soon as YOLOv7 misaligns the person.

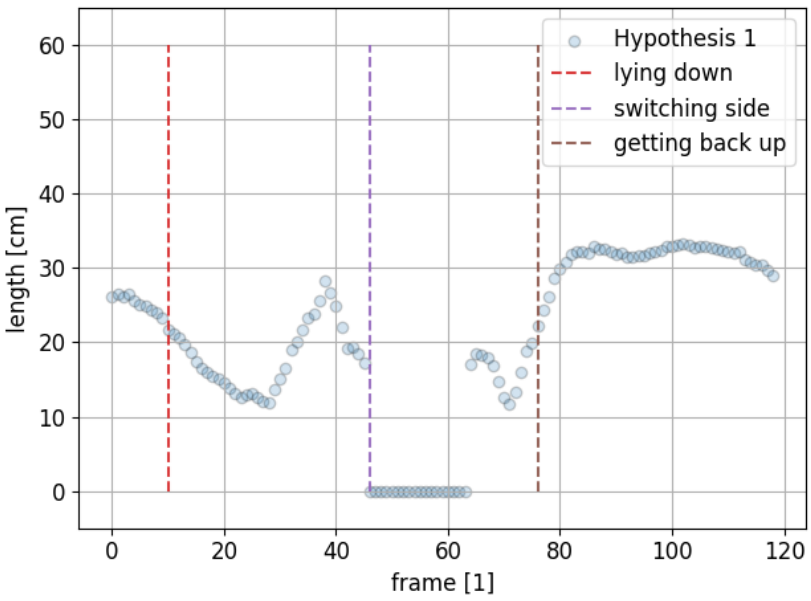


Figure 43: Length of nose - left shoulder line length estimations for a person lying down including unsupported key points.

4.4 Speed Analysis

Speed is a critical factor for fitting and inference, as processes that are too slow can be detrimental, especially when handling real-time data requirements. Overall, the algorithm demonstrates high speed, with the time required for the fitting process being relatively low across different scenarios. However, there is a noticeable discrepancy between the time needed for the fitting and inference processes, with the latter being significantly more time-consuming.

4.4.1 Fitting Time Analysis

The fitting time analysis has been done with the default parameters described in the GitHub repository's ReadMe [20]. However, to show how some parameters influence the fitting time, two different parameter sets have been tested. The difference between parameter set 1 and parameter set 2 is primarily defined by the spline knot delta (which is the time distance between each knot of the B-Spline) and the covariance parameter ν (used in the covariance matrix of the GMMs). Parameter set 1, with a spline knot delta of 500 and ν of 200, is optimized for more precision in the spline representation but may sacrifice a bit of runtime. On the other hand, parameter set 2, with a spline knot delta of 1000 and a ν of 500, tends to have shorter fitting times because it requires fewer precise adjustments and fewer knot updates. The comparison of the parameter sets was only done with two or three people in the scene, as the fitting time for one person is already relatively low.

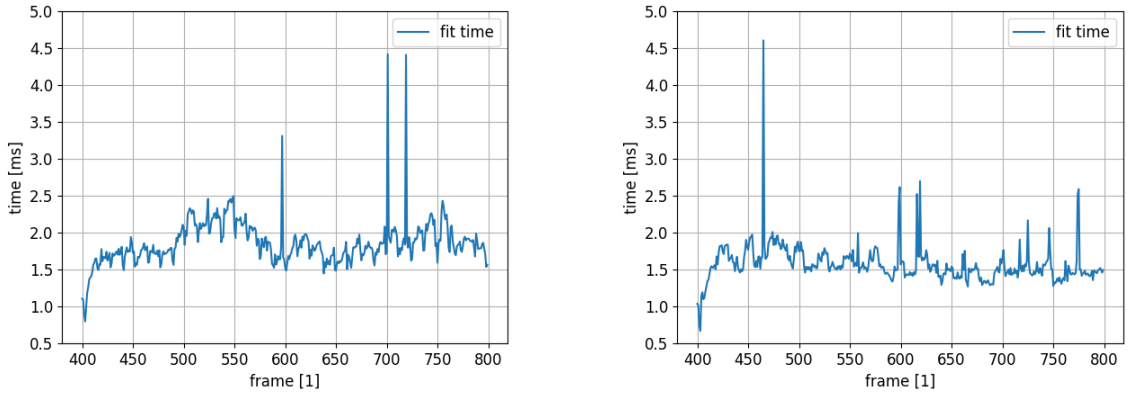
The fitting process is generally fast, with mean times ranging from approximately 0.74 ms for one person to around 4.40 ms for three people when using parameter set 1, as shown in table 5. Parameter set 1 exhibits slightly higher fitting times, mainly when more people are in the scene. The small spikes in fitting time, illustrated in figure 44, can be attributed to instances where a new frame exceeds the last knot of the B-Spline. This overflow necessitates the addition of a new knot and an increase in the parameters to be fitted, thereby causing a temporary spike in the computation time. The figure also shows how the increased spline knot delta and precision parameter in parameter set 2 lead to lower fitting times overall, with fewer spikes compared to parameter set 1. The reason for the bigger spikes in the fitting times is not entirely clear, but might be a result of the operating system's scheduling of its threads.

	Parameter Set 1			**Parameter Set 2**	
	One Person	*Two People*	*Three People*	*Two People*	*Three People*
mean $\pm$ std.	0.74 ± 0.23	1.86 ± 0.31	4.40 ± 0.48	1.57 ± 0.26	3.62 ± 0.58
max	1.20	4.41	8.46	4.60	5.20
min	0.35	0.80	3.35	0.67	1.93

Table 5: Fitting time comparison for different numbers of people in the scene (in ms, lower is better).

	Parameter Set 1			**Parameter Set 2**	
	One Person	*Two People*	*Three People*	*Two People*	*Three People*
mean $\pm$ std.	1497 ± 522	552 ± 90	229 ± 22	552 ± 90	284 ± 53
max	2864	1252	298	1252	519
min	835	227	118	227	192

Table 6: Fitting time comparison for different numbers of people in the scene (in FPS, higher is better).



(a) Parameter set 1.

(b) Parameter set 2.

Figure 44: Fitting time needed by the algorithm after adding two frames for two hypotheses (in ms, lower is better).

Figure 45 shows the fitting time the algorithm needs for three people over the whole sequence, highlighting the differences in the number of hypotheses the algorithm has determined in the frame. The figure points out how fast the algorithm is and that the hypotheses' management is not yet optimal. Between frames 180 and 410, there should be, in fact, three hypotheses - as three people are present in the scene. However, the algorithm only detects two, increases the number of hypotheses to three at frame 190, then decreases it to two at frame 280 and increases it again to three at frame 320.

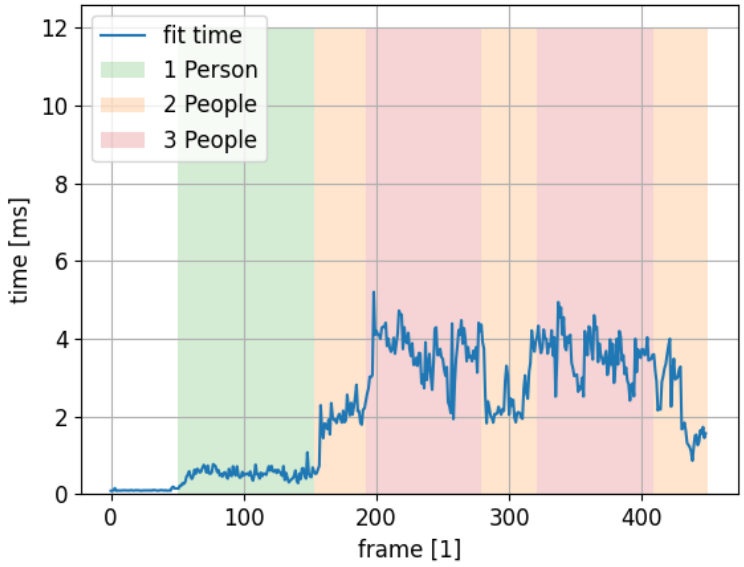
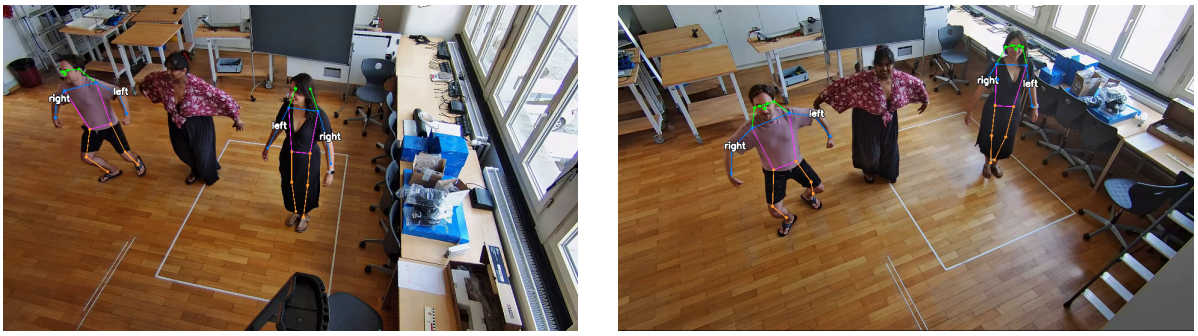


Figure 45: Fitting time needed and number of hypotheses determined by the algorithm for three people over the whole sequence, parameter set 2 (in ms, lower is better).

Figure 46 illustrates that this phenomenon occurs because the YOLOv7 algorithm struggles to detect the person in the centre of the frame. This difficulty appears to be linked to the person's darker skin tone, as they are the only subject with a dark skin colour and the only one who was inconsistently recognized.



(a) Frame of camera 1.

(b) Frame of camera 2.

Figure 46: Subject in the centre of the scene not recognized by YOLOv7.

4.4.2 Inference Time Analysis

In contrast to the fitting process, the inference process of the YOLOv7 takes considerably more time, with mean times being around 10 ms but also spike up to 93.12 ms. The time required for

inference shows significant variability, with occasional spikes that do not have a clear explanation, as seen in figure 47. These spikes suggest that external factors may influence the inference time, such as variations in scene complexity or computational resource allocation. The inference process is noticeably slower than the fitting process, which is expected, given the more computationally intensive nature of inference tasks. This difference underscores the importance of optimizing inference algorithms, particularly when real-time performance is a requirement. However, YOLOv7's overall speed seems not to be impacted by the number of people present in the scene.

	One Person	*Two People*	*Three People*
mean $\pm$ std.	11.36 ± 5.61	11.39 ± 6.33	10.57 ± 1.69
max	53.15	93.12	23.50
min	8.93	9.03	9.29

Table 7: Inference time comparison of YOLOv7 for different numbers of people in the scene (in ms, lower is better).

	One Person	*Two People*	*Three People*
mean $\pm$ std.	95 ± 16	94 ± 14	96 ± 8
max	112	111	108
min	19	11	43

Table 8: Inference time comparison of YOLOv7 for different numbers of people in the scene (in FPS, higher is better).

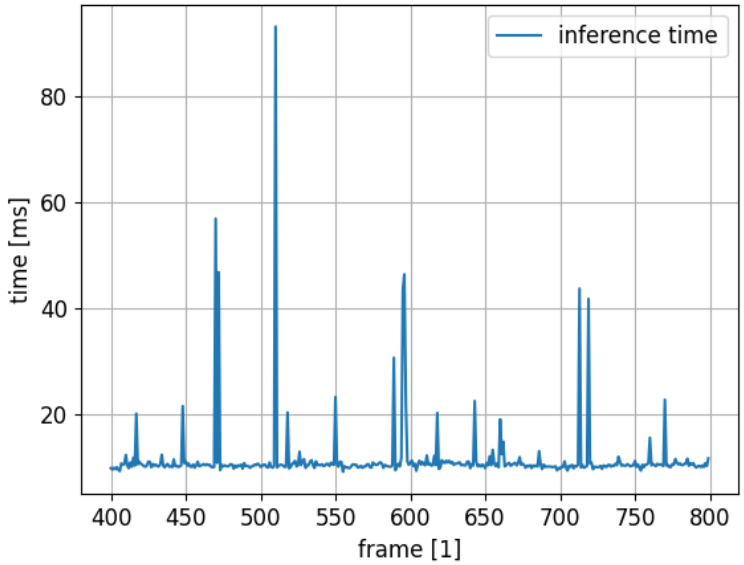


Figure 47: Inference time of YOLOv7 for two people in the scene (in ms, lower is better).

In summary, MC3D-TRECSIM can achieve real-time performance for the fitting process of at least three subjects - with a FPS minimum of 118 - but the inference process seems to be the main bottleneck in terms of speed. In order to improve the overall performance of the algorithm, it is crucial to optimize the inference process further. More plots can be found in the appendix F Speed Analysis Plots.

5 Discussion and Outlook

The MC3D-TRECSIM algorithm performs well in typical conditions, effectively handling the tasks of tracking individuals and reconstructing their 3D movements in real-time. Its accuracy in limb length estimation - with most approximations closely aligning with measured lengths - underscores its potential for practical applications, where precise human motion tracking is essential. However, minor glitches due to inaccuracies in YOLOv7 detections suggest that further refinement in the detection process could improve the system's overall reliability.

The algorithm faces challenges in edge cases, particularly when individuals cross paths or enter the scene close to the cameras. These scenarios reveal weaknesses in the system's ability to differentiate and track individuals, leading to distorted reconstructions accurately. Additionally, the algorithm struggles with uncommon body orientations, such as when a person is lying down, leading to significant inaccuracies in YOLOv7's detections.

While MC3D-TRECSIM is robust under typical conditions, its performance in complex scenarios can be enhanced by improving detection accuracy, refining key point tracking, and adapting the system to handle a broader range of body orientations and entry scenarios.

The future development of the MC3D-TRECSIM algorithm offers several potential paths for enhancement. One key area is expanding the model to incorporate more detailed information about limbs and limb lengths. This could improve the accuracy of 3D reconstructions, particularly in scenarios where the current system struggles, such as with uncommon body orientations or proximity to cameras. By integrating these additional parameters, the system could better handle the nuances of human movement.

Another essential step will be benchmarking the MC3D-TRECSIM algorithm against other state-of-the-art motion capture systems, such as Vicon [1] and Qualisys [2]. This comparison would validate the algorithm's effectiveness and highlight areas where it can be further optimized to match or exceed the performance of established systems.

Conducting a comprehensive parameter study is essential to fine-tuning the algorithm's settings and ensuring optimal performance across different environments and conditions. This study would help identify the most critical parameters that influence the system's accuracy and robustness, providing a foundation for further refinement and customization of the model.

Expanding the system to include more cameras, each with non-overlapping fields of view, presents an exciting opportunity to enhance the algorithm's capabilities. By integrating data from multiple cameras with distinct perspectives, the system could achieve more comprehensive 3D reconstructions and improve tracking accuracy, particularly in complex scenarios.

Lastly, testing other 2D pose estimation algorithms and exploring their compatibility with the MC3D-TRECSIM system could offer valuable insights into alternative approaches for tracking human motion. The most effective methods for achieving accurate 3D reconstructions and

tracking trajectories in various scenarios could be identified by comparing the performance of different algorithms.

6 Listings

6.1 References

[1] *Award Winning Motion Capture Systems — Vicon*, 2024. [Online]. Available: https://www.vicon.com/ (visited on Aug. 7, 2024).

[2] *Motion Capture Technology and Systems — Qualisys*, Sep. 2013. [Online]. Available: https://www.qualisys.com/ (visited on Aug. 7, 2024).

[3] P. Karashchuk, K. L. Rupp, E. S. Dickinson, *et al.*, "Anipose: A toolkit for robust markerless 3D pose estimation", *Cell Reports*, vol. 36, no. 13, p. 109 730, Sep. 2021, ISSN: 22111247. DOI: 10.1016/j.celrep.2021.109730. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2211124721011797 (visited on Aug. 18, 2023).

[4] *OpenCV: Camera Calibration and 3D Reconstruction*. [Online]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html (visited on Aug. 9, 2024).

[5] A. Anwar, *What are Intrinsic and Extrinsic Camera Parameters in Computer Vision?*, Mar. 2022. [Online]. Available: https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec (visited on Aug. 9, 2024).

[6] *CalibrateCamera() OpenCV in Python*, Jun. 2023. [Online]. Available: https://www.geeksforgeeks.org/calibratecamera-opencv-in-python/ (visited on Sep. 5, 2024).

[7] T. Weinmann, "Konzepte des Maschinellen Lernens", handout, ZHAW Winterthur School of Engineering, Feb. 2023.

[8] L. Mao, *LogSumExp and Its Numerical Stability*, May 2018. [Online]. Available: https://leimao.github.io/blog/LogSumExp/ (visited on Aug. 12, 2024).

[9] S. Stingelin, "Numerik und Differentialgleichungen", handout, ZHAW Winterthur School of Engineering, Jun. 2016.

[10] *Eigen: Main Page*. [Online]. Available: https://eigen.tuxfamily.org/dox/ (visited on Aug. 14, 2024).

[11] *LBFGS++: A Header-only C++ Library for L-BFGS and L-BFGS-B Algorithms*. [Online]. Available: https://lbfgspp.statr.me/ (visited on Aug. 14, 2024).

[12] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, Jul. 2022. arXiv: 2207.02696 [cs]. [Online]. Available: http://arxiv.org/abs/2207.02696 (visited on Aug. 12, 2024).

[13] K.-Y. Wong, *WongKinYiu/Yolov7*, Aug. 2024. [Online]. Available: https://github.com/WongKinYiu/yolov7 (visited on Aug. 12, 2024).

[14] *PyQtGraph - Scientific Graphics and GUI Library for Python*. [Online]. Available: https://www.pyqtgraph.org/ (visited on Aug. 14, 2024).

[15] *OpenCV - Open Computer Vision Library*. [Online]. Available: https://opencv.org/ (visited on Aug. 14, 2024).

[16] F. Rameau, J. Park, O. Bailo, and I. S. Kweon, "MC-Calib: A generic and robust calibration toolbox for multi-camera systems", *Computer Vision and Image Understanding*, vol. 217, p. 103 353, Mar. 2022, ISSN: 10773142. DOI: 10.1016/j.cviu.2021.103353. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1077314221001818 (visited on Aug. 18, 2023).

[17] rameau-fr, *Rameau-Fr/MC-Calib*, Jul. 2024. [Online]. Available: https://github.com/rameau-fr/MC-Calib (visited on Aug. 6, 2024).

[18] *IPC-HDBW3841R-ZS-S2 - Dahua International*. [Online]. Available: https://www.dahuasecurity.com/products/All-Products/Network-Cameras/WizSense-Series/3-Series/8MP/IPC-HDBW3841R-ZS-S2=S2 (visited on Aug. 15, 2024).

[19] W. Dahmen and A. Reusken, *Numerik für Ingenieure und Naturwissenschaftler* (Springer-Lehrbuch). Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, ISBN: 978-3-540-76493-9. DOI: 10.1007/978-3-540-76493-9. [Online]. Available: http://link.springer.com/10.1007/978-3-540-76493-9 (visited on Aug. 27, 2024).

[20] *Rothen/mc3d-trecsim*. [Online]. Available: https://github.com/Rothen/mc3d-trecsim (visited on Aug. 6, 2024).

6.2 Glossary

Eigen A C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. 15

MC-Calib A generic and robust calibration toolbox for multi-camera systems. 18, 19

MC3D-TRECSIM A 3D trajectory reconstruction software using multi-camera 2D data. 15, 35, 39, 41, 50, 54, 61, 62

OpenCV An open-source library for real-time image and video processing in computer vision. 6, 17, 18

PyQtGraph A pure-python graphics and GUI library built on PyQt / PySide and numpy.. 17

YOLO A real-time object detection system simultaneously predicting bounding boxes and class labels for multiple objects in images. 16

YOLOv7 An advanced real-time object detection model that enhances accuracy and efficiency through optimized architecture and improved training techniques. 16, 17, 19, 21, 25, 27, 32, 33, 35, 36, 39, 41–47, 49, 50, 53–55, 59–62

6.3 List of Figures

6.4 List of Generative AI Systems

- OpenAI. (2024) ChatGPT (Version 4o).
  `https://chat.openai.com/chat`
  - Generation of text proposals
  - Generation of code proposals for the implementation of the system
- Microsoft (2024) Copilot (GPT-4o).
  `https://github.com/features/copilot`
  - Generation of code proposals for the implementation of the system
- DeepL (2023). DeepL Translator (version DeepL Free).
  `https://www.deepl.com/translator`
  - Translation of text passages

6.5 List of Tables

6.6 Acronyms

2D two-dimensional. 2–4, 6, 8, 16, 17, 22–25, 42, 43, 62

3D three-dimensional. 1–9, 15, 17–28, 32–34, 41–45, 47, 52, 62

B-Spline Basis Spline. 12, 13, 22, 25–29, 56, 57

BFGS Broyden-Fletcher-Goldfarb-Shanno. 15, 16

CAD computer-aided design. 12

EM Expectation-Maximization. 10, 11, 22, 24, 26, 30–32

FPS frames per second. 18, 57, 60, 61

GMM Gaussian Mixture Model. 10, 11, 22, 56

HMM Hidden Markov Model. 11

L-BFGS Limited-memory Broyden-Fletcher-Goldfarb-Shanno. 15, 16, 26

PDF probability density function. 75

Appendices

ZHAW Winterthur
Institute of Applied Mathematics and Physics

Proposal Master Thesis

MC3D-TRECSIM: Multi-Camera 3D Trajectory Reconstruction for Enhanced Crowd Safety and Infrastructure Monitoring

Benjamin Heuberger

Time frame: 18.09.2023 - 18.09.2024

Supervisor
Dr Thomas Oskar Weinmann

1 Background

To ensure the safety of people in crowded areas or automated infrastructure such as public transport, people are employed to monitor the situation utilising camera surveillance. However, the need for more personnel in this field and the complex nature of the problem on hand demands a digital solution, which can determine whether the infrastructure operates as intended.

2 Solution

The suggested solution is to develop a multi-camera system to track individuals' three-dimensional (3D) pose. To achieve this goal, the proposed method takes advantage of existing two-dimensional (2D) pose estimation algorithms that use camera feeds to identify the spatial orientation of individuals. The main objective is fusing the 2D pose information acquired from multiple camera sources to compute a 3D representation of these individuals, including their pose and position.

3 Goals

This thesis aims to develop and evaluate a robust machine learning system to extract 3D pose trajectories of an arbitrary number of individuals simultaneously by fusing the information of an arbitrary number of cameras. Specific objectives include:

- Develop an in-depth understanding of existing 2D pose estimation algorithms and their applicability to multi-camera setups.
- Investigate and implement methodologies for fusing 2D pose information from multiple cameras into a reasonable 3D reconstruction.
- Devise algorithms for accurately tracking and predicting the trajectories of individuals as they navigate within the monitored space.
- Evaluate the system's boundaries by establishing and testing edge cases.
- Establish a framework for real-time assessment of potential hazards and unauthorised access based on the reconstructed 3D trajectories.

4 Deliverables

Upon successful completion of this research, the following deliverables will be provided:

- Code and Documentation: A functional software solution and comprehensive documentation for easy setup, usage, and future development.

- Technical Reports: In-depth reports explaining methodologies, design choices, and implemented algorithms.
- Demonstration Materials: Visual aids like presentation slides and videos to showcase system functionality.

These deliverables ensure accessibility, usability, and broader dissemination of the multi-camera 3D person trajectory reconstruction system and its findings.

5 Reporting

The work will be conducted at the ZHAW or remotely from home, and the progress reporting will be done biweekly with Dr Weinmann and Mr Rejzek.

B Log-Likelihood Calculations

This section shows the calculations for the bivariate normal distribution used in the E-step of the EM algorithm and how equation 37 was derived. We start with the log-likelihood function which should be maximized:

$$\sum_{n=1}^{N}\sum_{j=1}^{J} r_{n,j}(\ln\left(\mathbb{P}(m_n|z_n,\theta)\right)+\ln\left(\pi_j\right)) \tag{59}$$

Splitting the sum into two parts and focusing on the first part:

$$\sum_{n=1}^{N}\sum_{j=1}^{J} r_{n,j}(\ln\left(\mathbb{P}(m_n|z_n,\theta)\right)) \quad + \quad \sum_{n=1}^{N}\sum_{j=1}^{J} r_{n,j}(\ln\left(\pi_j\right)) \tag{60}$$

Replacing $\mathbb{P}(m_n|z_n,\theta)$ by the bivariate normal distribution:

$$= \sum_{n=1}^{N}\sum_{j=1}^{J} r_{n,j}(\ln\left(\mathcal{N}(m_n|P_K(h_j(t_n|\theta_j)),\Sigma)\right)) \tag{61}$$

Replacing $P_K(h_j(t_n|\theta_j))$ by the calculated world point P_w:

$$\sum_{n=1}^{N}\sum_{j=1}^{J} r_{n,j}(\ln\left(\mathcal{N}(m_n|P_w,\Sigma)\right)) \tag{62}$$

The $\ln$ of the probability density function (PDF) bivariate normal distribution is given by

$$-\frac{1}{2}\left(k\ln\left(2\pi\right)+\ln\left(|\Sigma|\right)+(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \tag{63}$$

where k is the dimension of the distribution, μ is the mean and Σ is the covariance matrix. In our case, $k=2$, $\mu=P_w$, $x=m_n$ and $\Sigma=\nu I_2$. Replacing these values in the equation above results in

$$-\frac{1}{2}\left(2\ln\left(2\pi\right)+\ln\left(|\nu I_2|\right)+(m_n-P_w)^T(\nu I_2)^{-1}(m_n-P_w)\right), \tag{64}$$

by applying these rules

$$\ln\left(|\nu I_2|\right) = \ln\left(\left|\begin{bmatrix}\nu & 0\\ 0 & \nu\end{bmatrix}\right|\right) = \ln\left(\nu^2 - 0^2\right) = \ln\left(\nu^2\right) = 2\ln\left(\nu\right) \tag{65}$$

$$(\nu I_2)^{-1} = \nu^{-1} I_2^{-1} = \nu^{-1} I_2 = \frac{1}{\nu} I_2, \tag{66}$$

and then applying the transformation $(\nu I_2)^{-1} = \frac{1}{\nu} I_2$ to $(m_n - P_w)^T (\nu I_2)^{-1} (m_n - P_w)$

$$(m_n - P_w)^T (\nu I_2)^{-1} (m_n - P_w) \tag{67}$$

$$= (m_n - P_w)^T (\frac{1}{\nu} I_2)(m_n - P_w) \tag{68}$$

$$= \frac{1}{\nu}(m_n - P_w)^T I_2 (m_n - P_w) \tag{69}$$

$$= \frac{1}{\nu}(m_n - P_w)^T (m_n - P_w) \tag{70}$$

$$= \frac{1}{\nu}||(m_n - P_w)||^2 \tag{71}$$

the equation simplifies to

$$-\frac{1}{2}\left(2\ln\left(2\pi\right) + 2\ln\left(\nu\right) + \frac{1}{\nu}||(m_n - P_w)||^2\right), \tag{72}$$

which then can finally be simplified to

$$-\ln\left(2\pi\right) - \ln\left(\nu\right) - \frac{1}{2\nu}||(m_n - P_w)||^2 \tag{73}$$

which then can be inserted into the original equation

$$\sum_{n=1}^{N}\sum_{j=1}^{J} r_{n,j}\left(-\ln\left(2\pi\right) - \ln\left(\nu\right) - \frac{1}{2\nu}||(m_n - P_w)||^2\right) \tag{74}$$

C Plots For Visual Robustness Inspection

One person walking

(a) Frame of camera 1.

(b) Frame of camera 2.

Figure 48: Frames of one person walking.

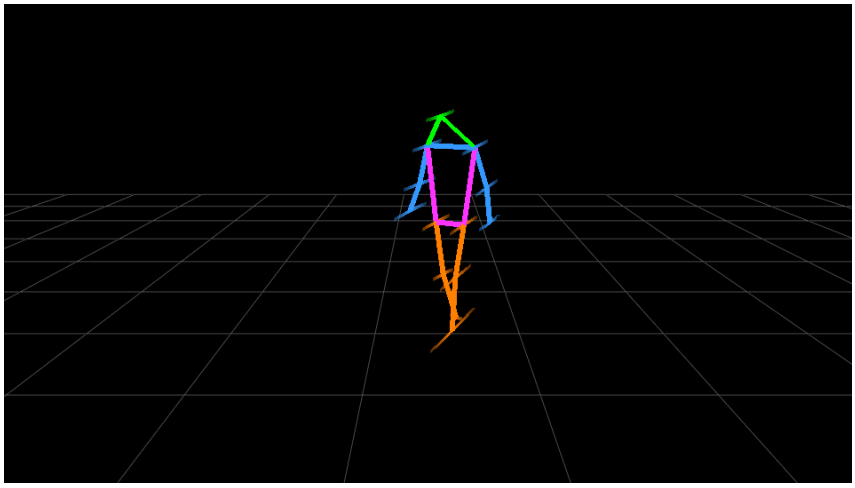


Figure 49: 3D reconstruction one person walking.

One person putting hands in the air

(a) Frame of camera 1.

(b) Frame of camera 2.

Figure 50: Frames of one person walking.

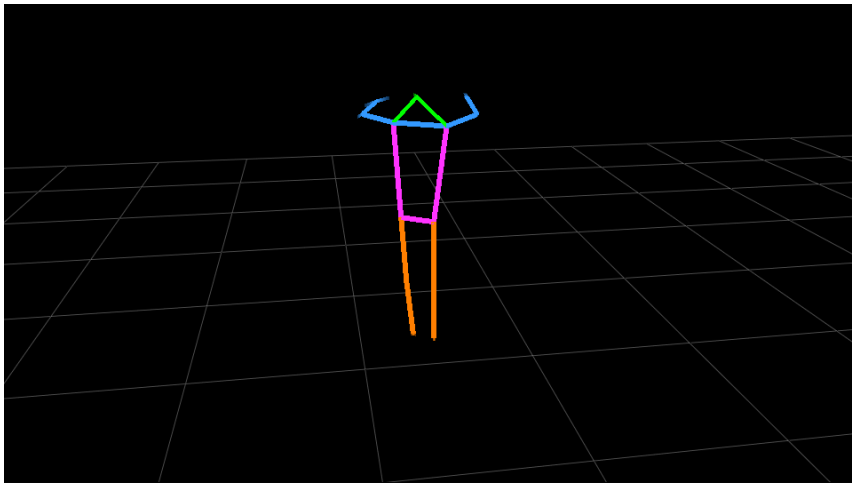
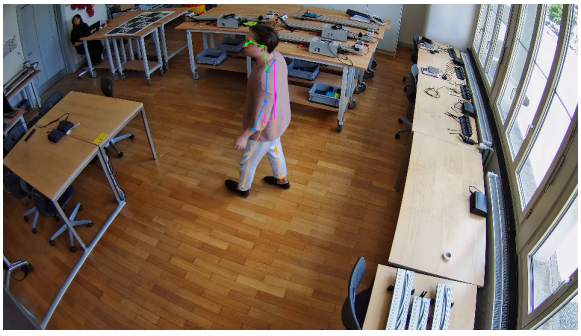
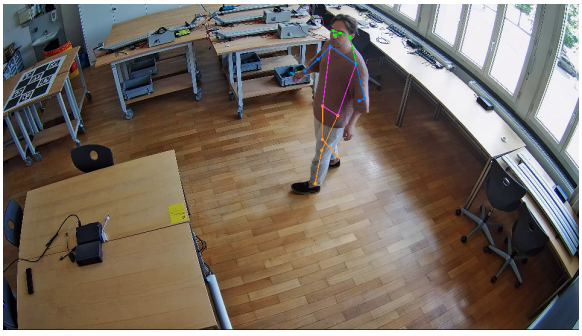


Figure 51: 3D reconstruction one person walking.

One person rotates



(a) Frame of camera 1.



(b) Frame of camera 2.

Figure 52: Frames of one person walking.

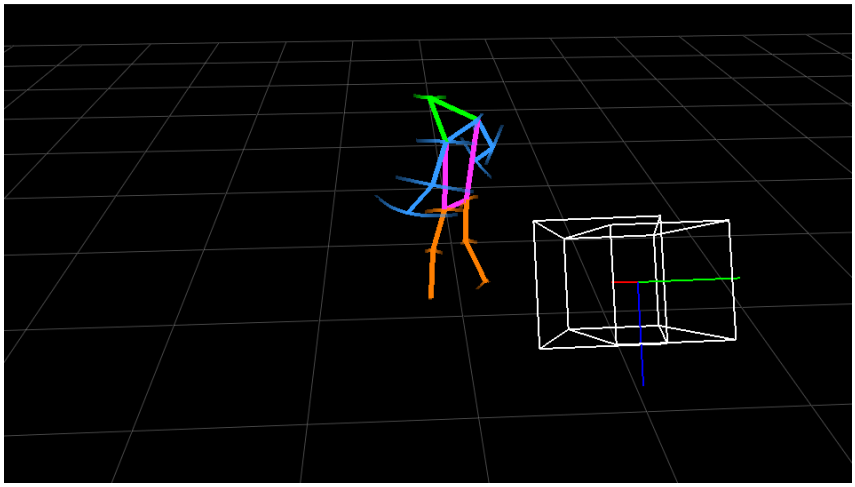
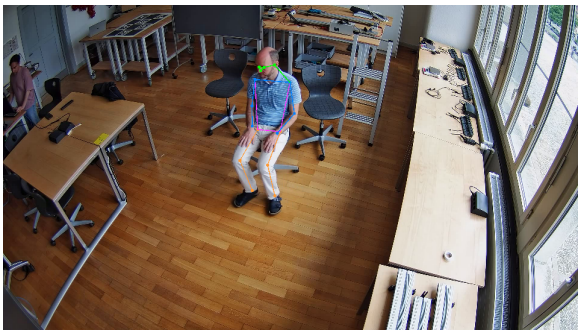
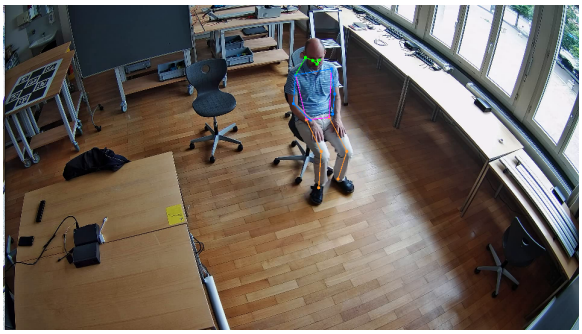


Figure 53: 3D reconstruction one person walking.

One person is sitting



(a) Frame of camera 1.



(b) Frame of camera 2.

Figure 54: Frames of one person walking.

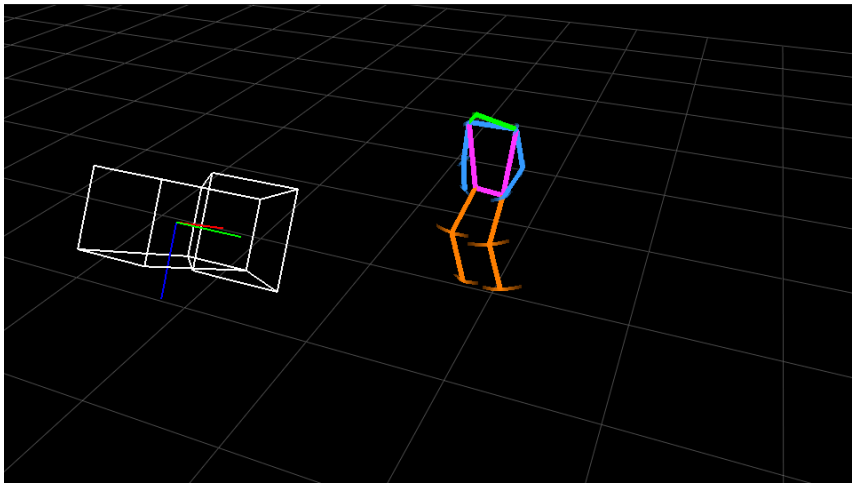
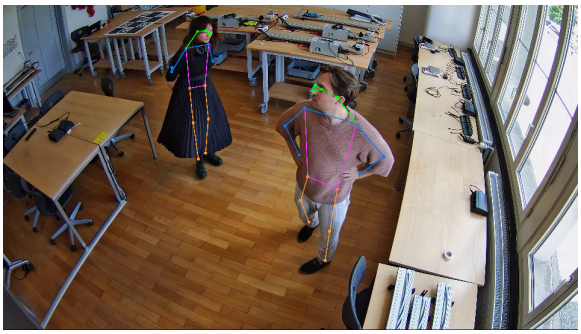


Figure 55: 3D reconstruction one person walking.

Two people are rotating



(a) Frame of camera 1.

(b) Frame of camera 2.

Figure 56: Frames of one person walking.

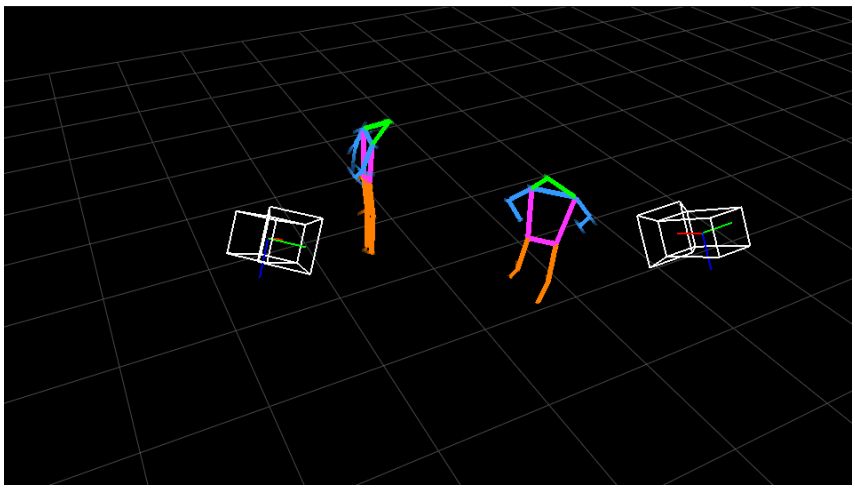


Figure 57: 3D reconstruction one person walking.

D Limb Length Comparisons

D.1 Limb Length Estimation Accuracy and Consistency

D.1.1 Full Limb Length Comparison Plots

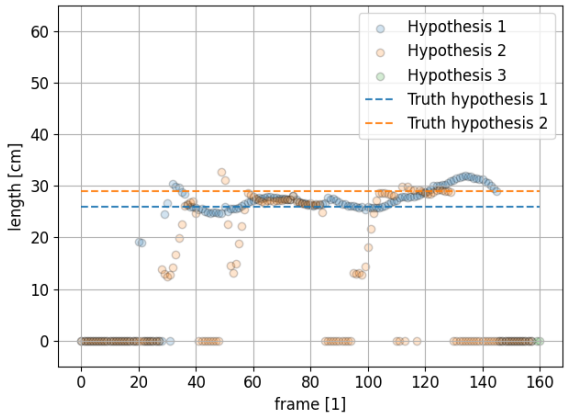


Figure 58: Nose - Right Shoulder

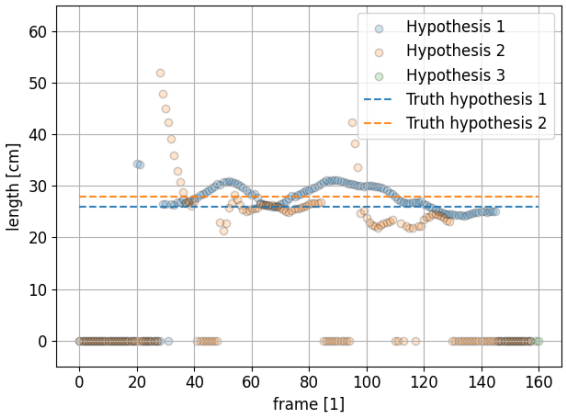


Figure 59: Nose - Left Shoulder

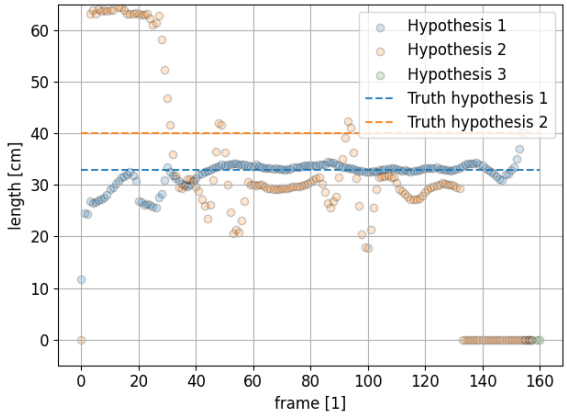


Figure 60: Shoulder Line

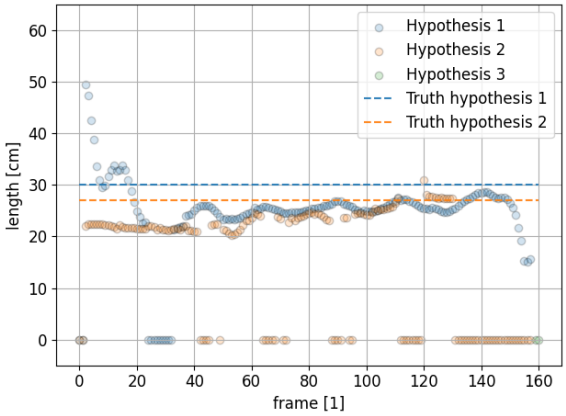


Figure 61: Right Brachium

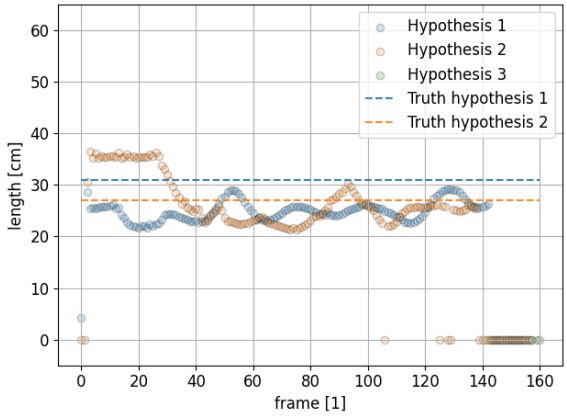


Figure 62: Left Brachium

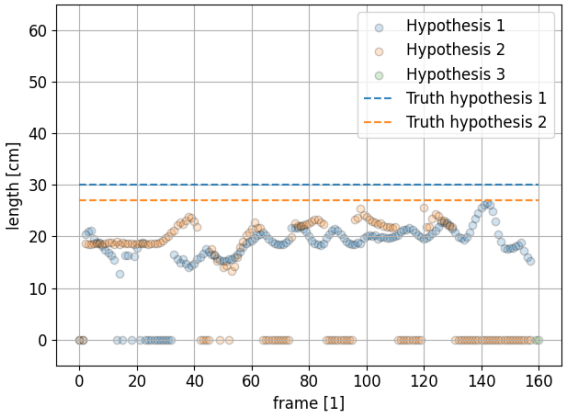


Figure 63: Right Antebrachium

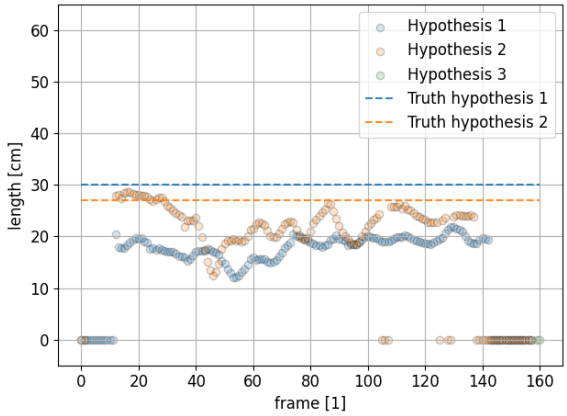


Figure 64: Left Antebrachium

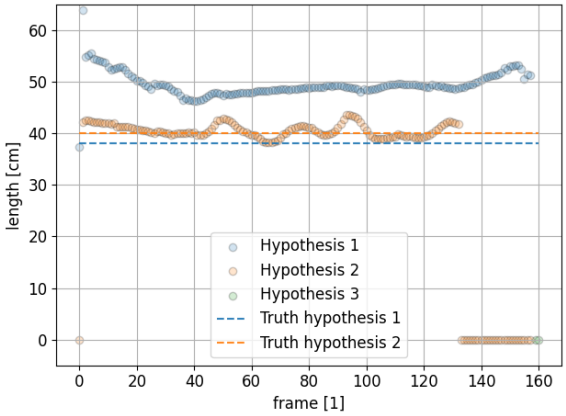


Figure 65: Right Flank

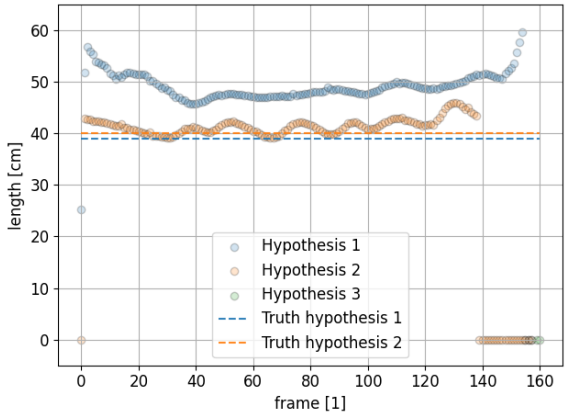


Figure 66: Left Flank

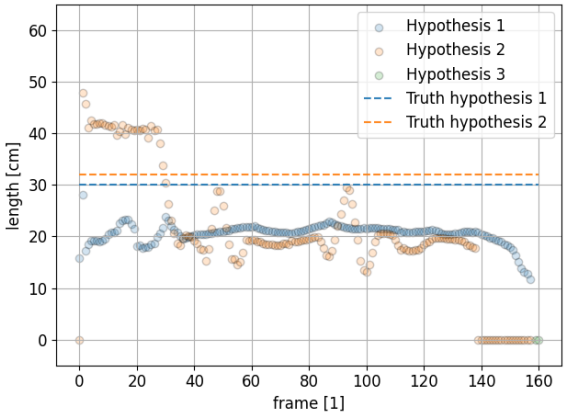


Figure 67: Waist

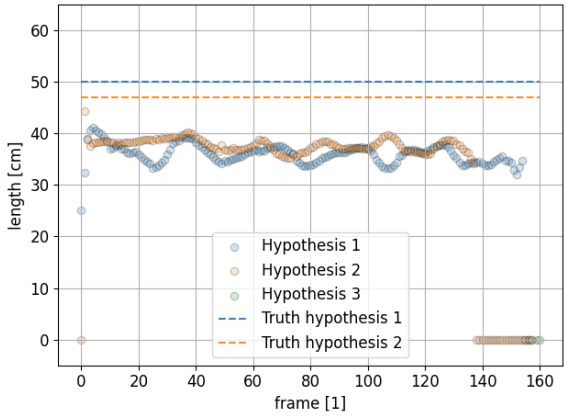


Figure 68: Right Thigh

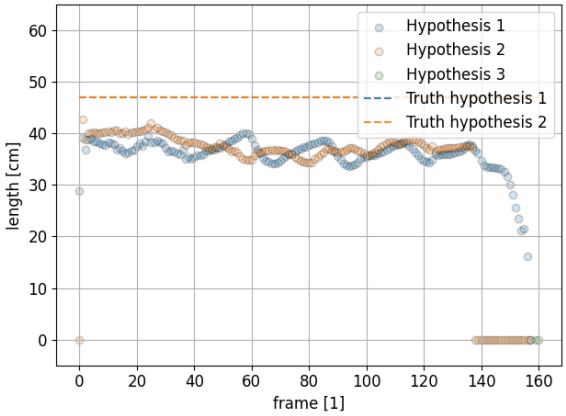


Figure 69: Left Thigh

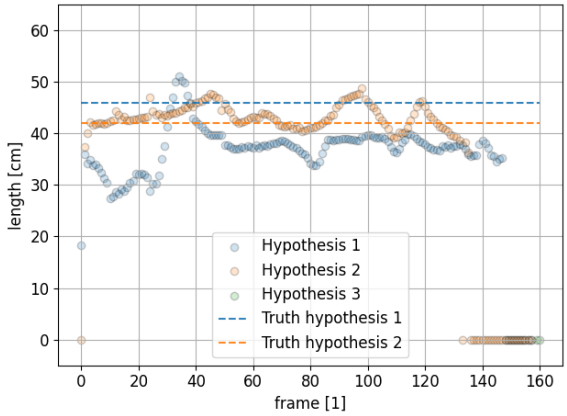


Figure 70: Right Shin

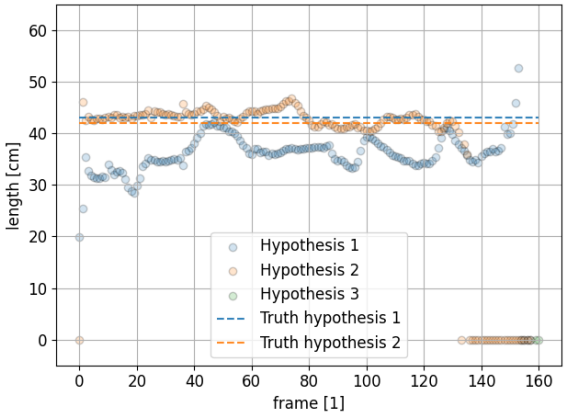


Figure 71: Left Shin

D.1.2 Limb Length Comparison Plots With Unsupported Key Points

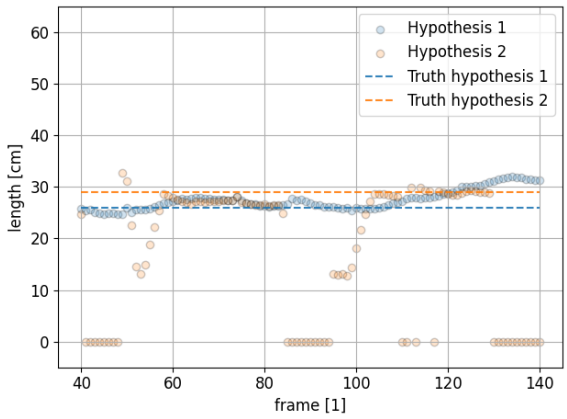


Figure 72: Nose - Right Shoulder

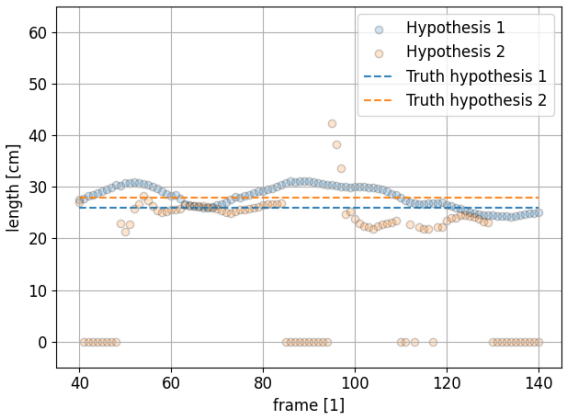


Figure 73: Nose - Left Shoulder

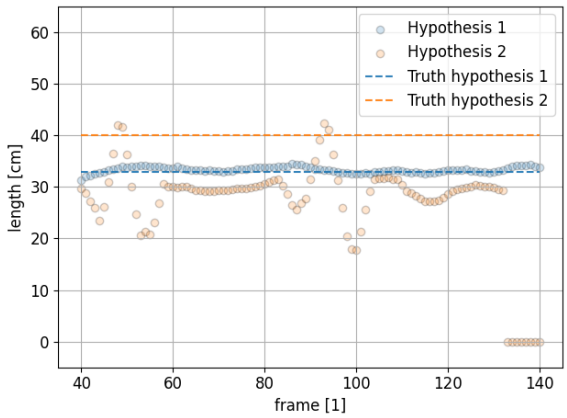


Figure 74: Shoulder Line

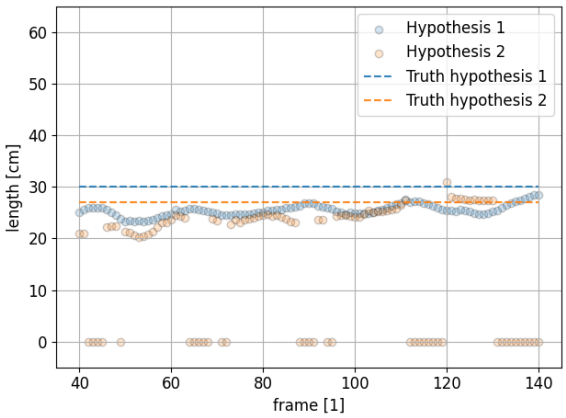


Figure 75: Right Brachium

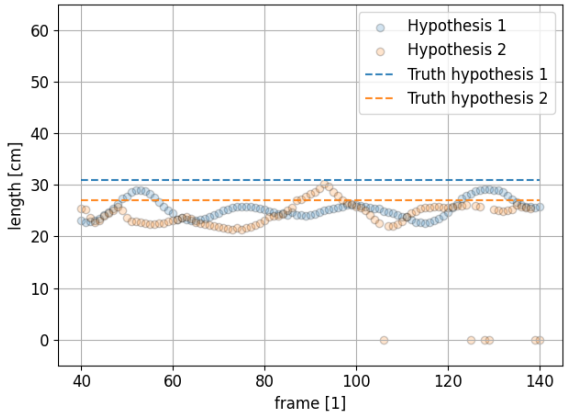


Figure 76: Left Brachium

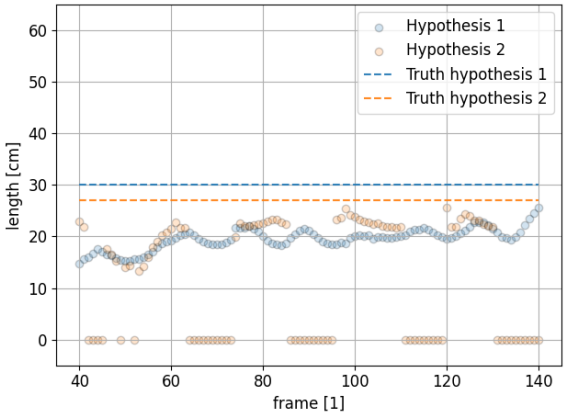


Figure 77: Right Antebrachium

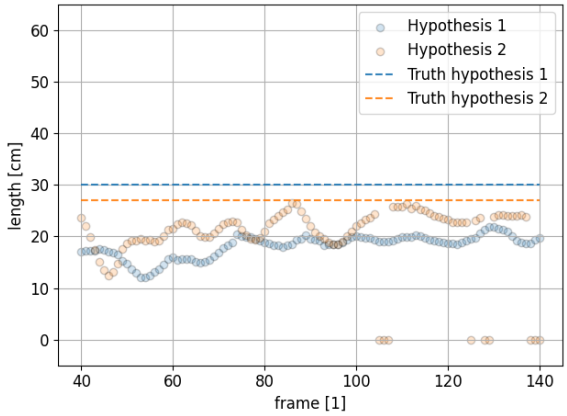


Figure 78: Left Antebrachium

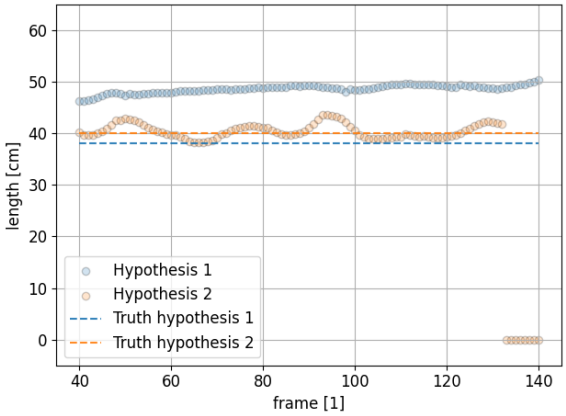


Figure 79: Right Flank

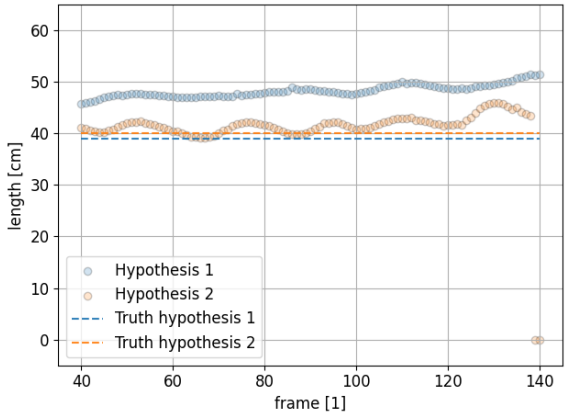


Figure 80: Left Flank

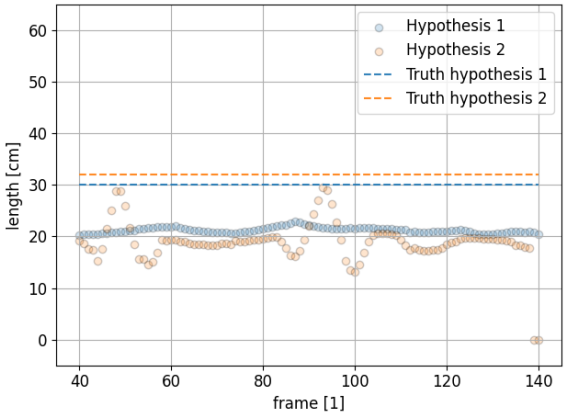


Figure 81: Waist

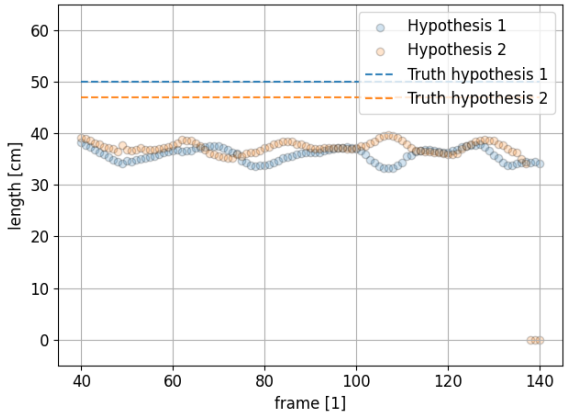


Figure 82: Right Thigh

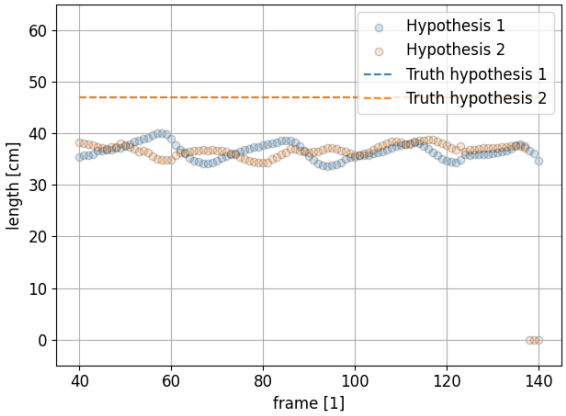


Figure 83: Left Thigh

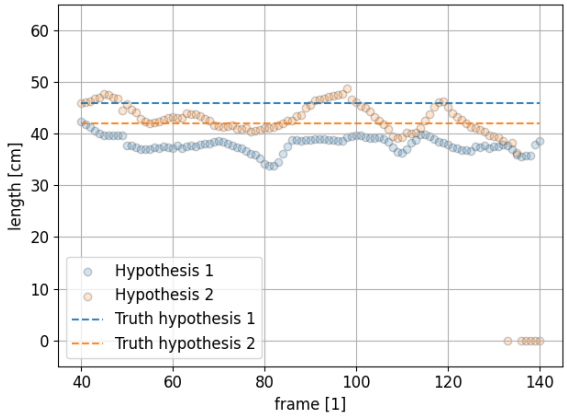


Figure 84: Right Shin

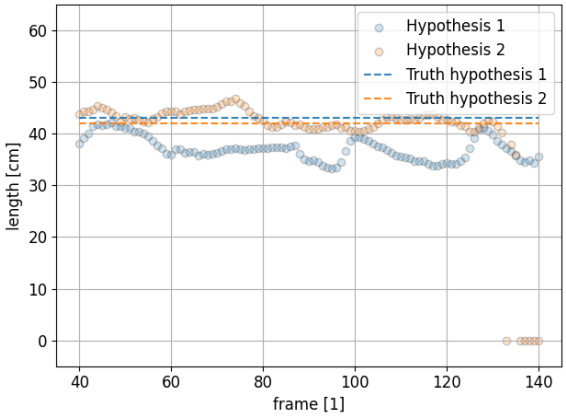


Figure 85: Left Shin

D.1.3 Limb Length Comparison Plots Without Unsupported Key Points

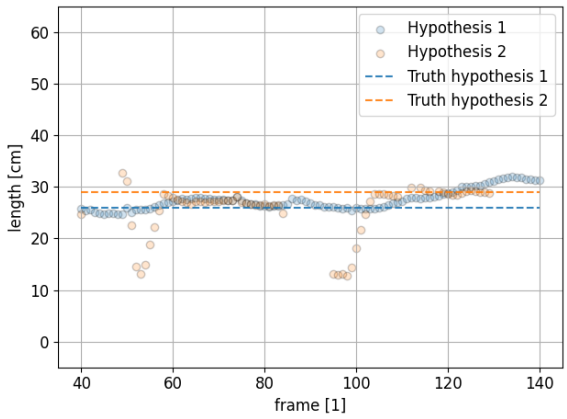


Figure 86: Nose - Right Shoulder

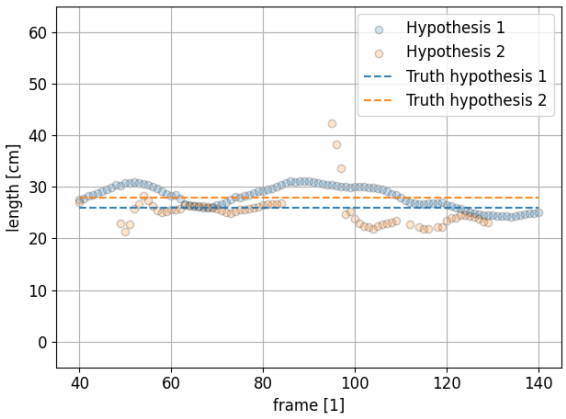


Figure 87: Nose - Left Shoulder

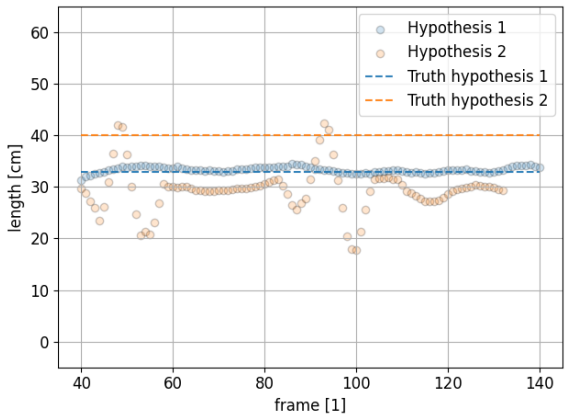


Figure 88: Shoulder Line

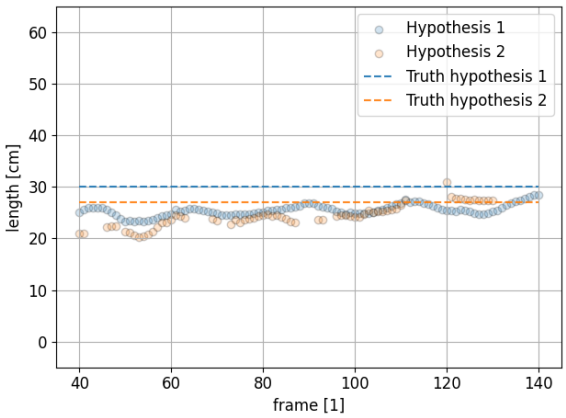


Figure 89: Right Brachium

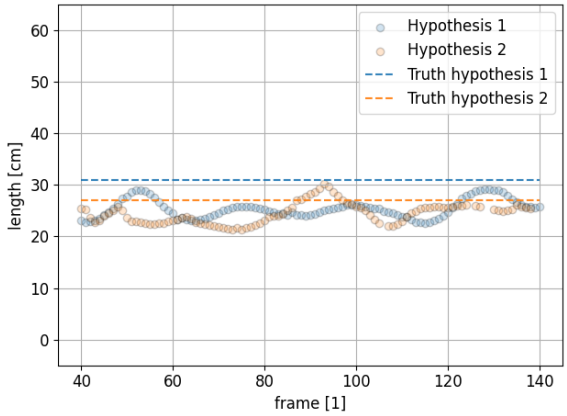


Figure 90: Left Brachium

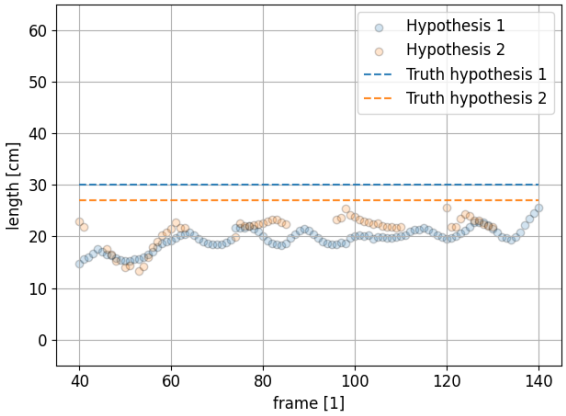


Figure 91: Right Antebrachium

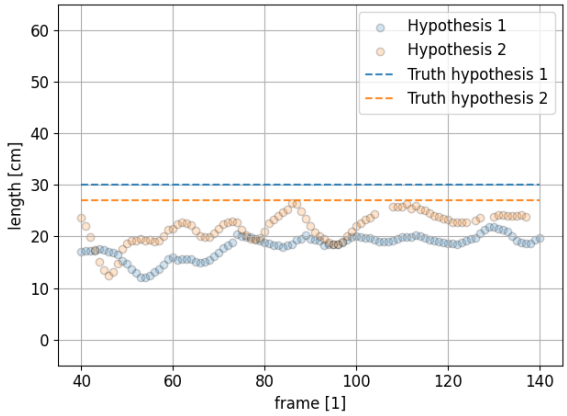


Figure 92: Left Antebrachium

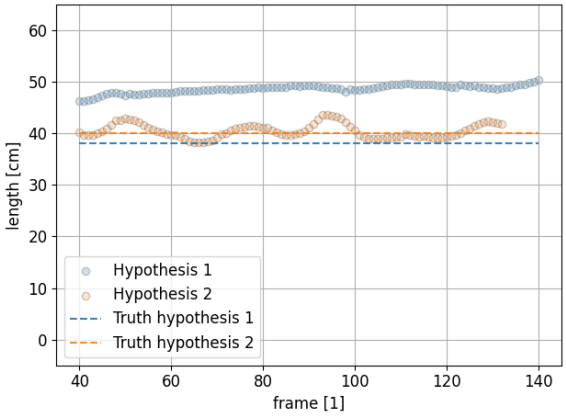


Figure 93: Right Flank

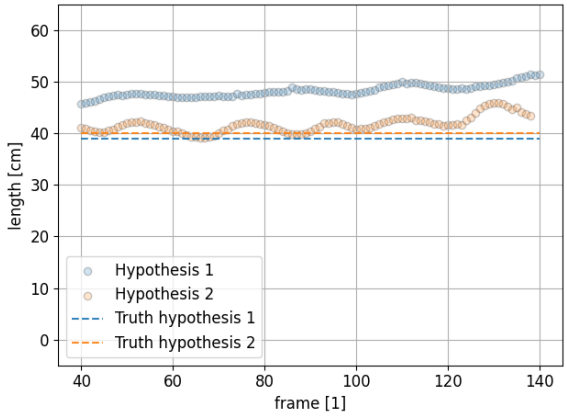


Figure 94: Left Flank

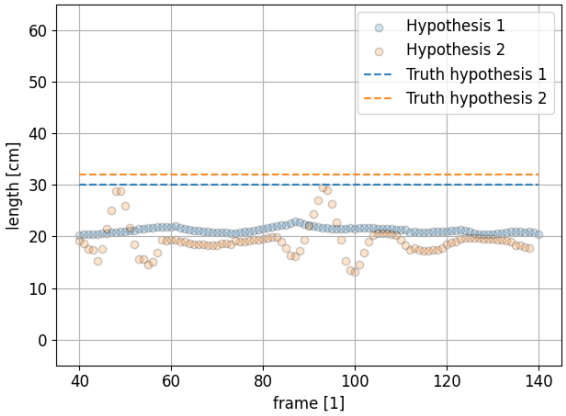


Figure 95: Waist

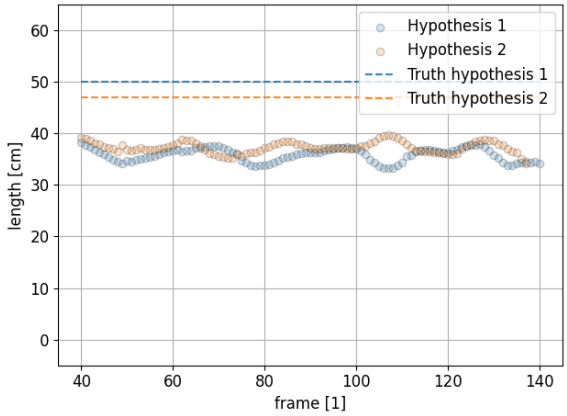


Figure 96: Right Thigh

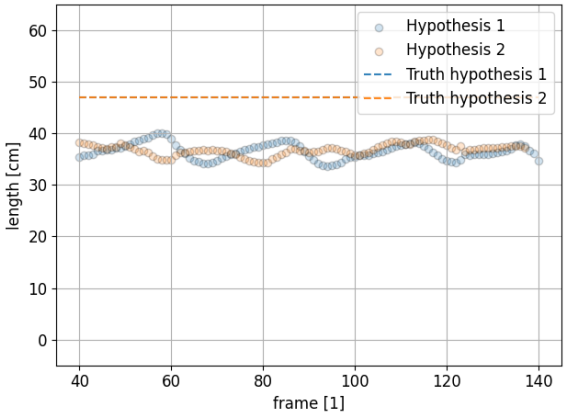


Figure 97: Left Thigh

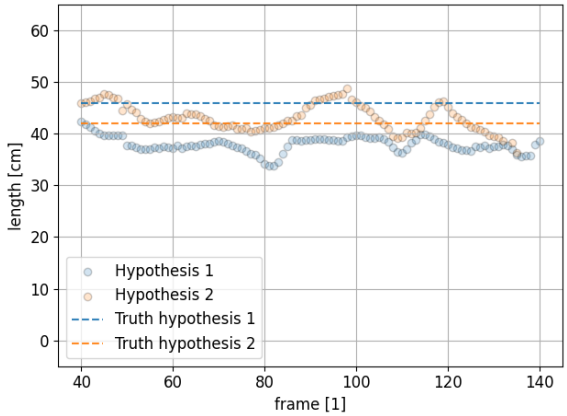


Figure 98: Right Shin

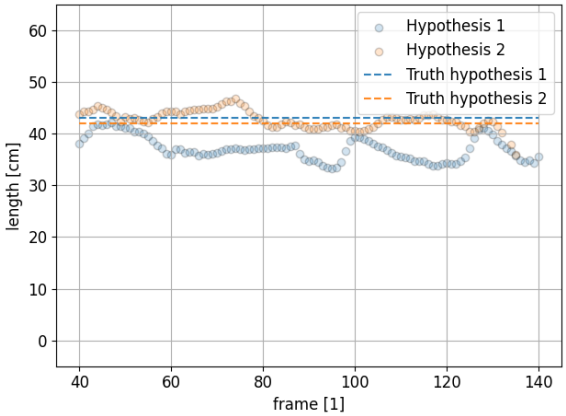


Figure 99: Left Shin

D.1.4 Limb Length Estimation with Less Smoothing

	Subject 1			Subject 2		
Limb	*Truth*	*Mean* $\pm$ *Std.*	Δ	*Truth*	*Mean* $\pm$ *Std.*	Δ
Nose-R. Shoulder	29	27.47 ± 1.94	-1.53	26	25.91 ± 5.26	-0.09
Nose-L. Shoulder	28	28.08 ± 2.15	$+0.08$	26	24.96 ± 3.13	-1.04
Shoulder Line	40	33.26 ± 0.59	-6.74	33	28.87 ± 4.38	-4.13
R. Brachium	27	25.45 ± 1.01	-1.55	30	24.30 ± 2.07	-5.70
L. Brachium	27	25.35 ± 1.76	-1.65	31	24.43 ± 1.93	-6.57
R. Antebrachium	27	19.42 ± 1.97	-7.58	30	21.25 ± 2.75	-8.75
L. Antebrachium	27	18.09 ± 2.05	-8.91	30	21.56 ± 2.98	-8.44
R. Flank	40	48.60 ± 0.75	$+8.60$	38	40.40 ± 1.30	2.40
L. Flank	40	48.25 ± 1.18	$+8.25$	39	41.72 ± 1.51	2.72
Waist	32	21.18 ± 0.55	-10.82	30	19.05 ± 2.88	-10.95
R. Thigh	47	35.72 ± 1.26	-11.28	50	37.26 ± 1.08	-12.74
L. Thigh	47	36.47 ± 1.53	-10.53	47	36.71 ± 1.02	-10.29
R. Shin	42	37.88 ± 1.48	-4.12	46	43.08 ± 2.47	-2.92
L. Shin	42	37.03 ± 2.18	-4.97	43	42.72 ± 1.65	-0.28

Table 9: Comparison of the measured and the approximated lengths (in cm) for the limbs of two subjects with less smoothing.

D.2 Limb Length Comparison Plots Of Standing Up

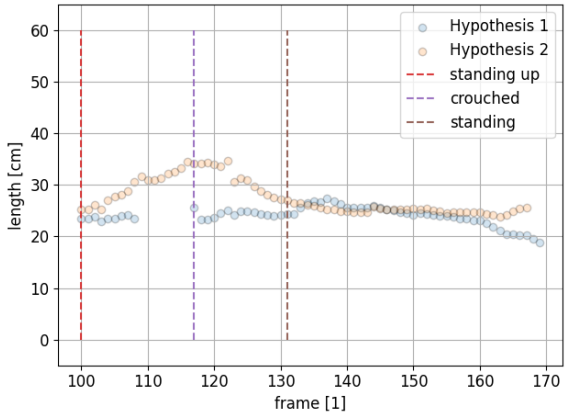


Figure 100: Nose - Right Shoulder

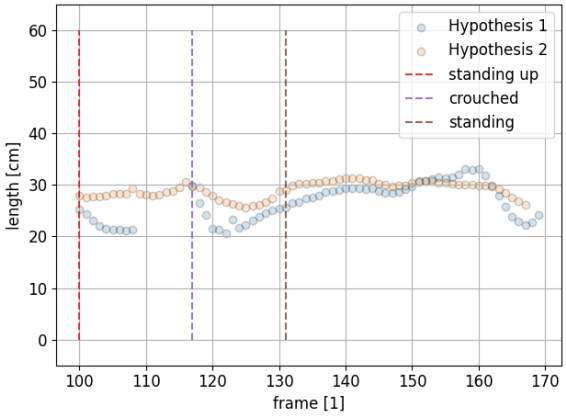


Figure 101: Nose - Left Shoulder

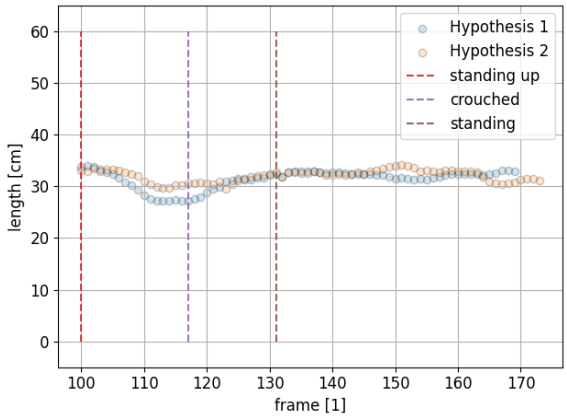


Figure 102: Shoulder Line

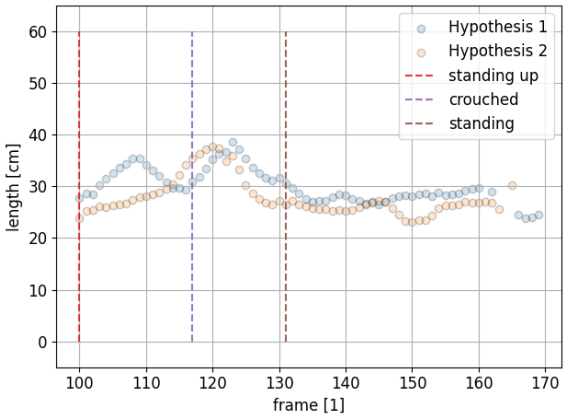


Figure 103: Right Brachium

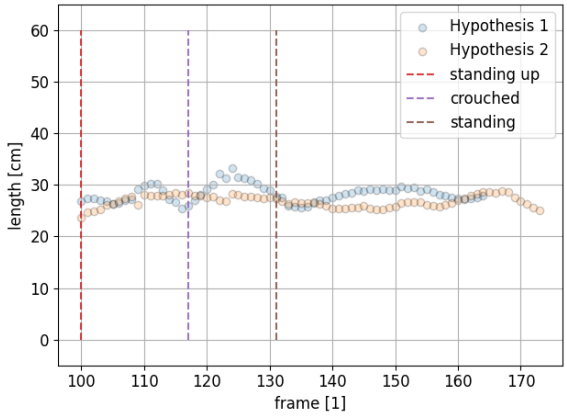


Figure 104: Left Brachium

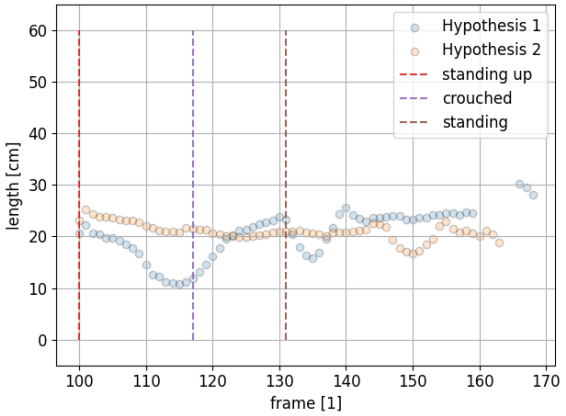


Figure 105: Right Antebrachium

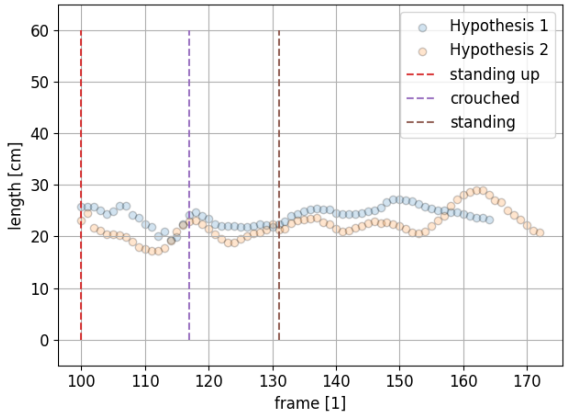


Figure 106: Left Antebrachium

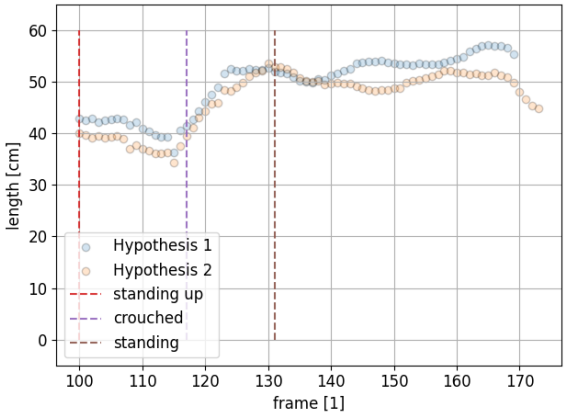


Figure 107: Right Flank

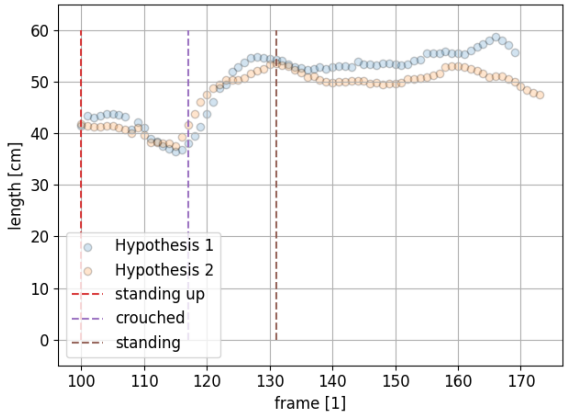


Figure 108: Left Flank

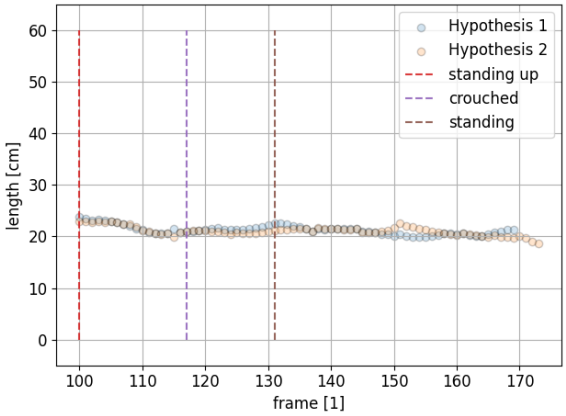


Figure 109: Waist

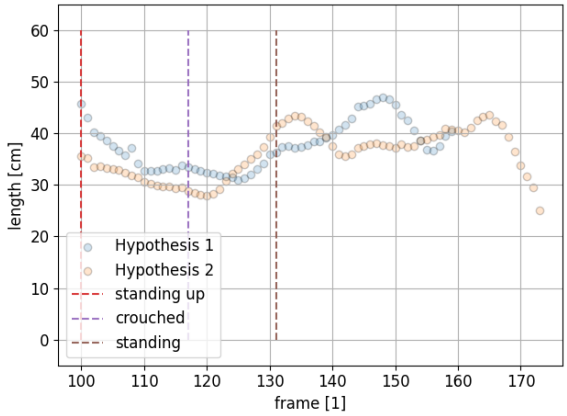


Figure 110: Right Thigh

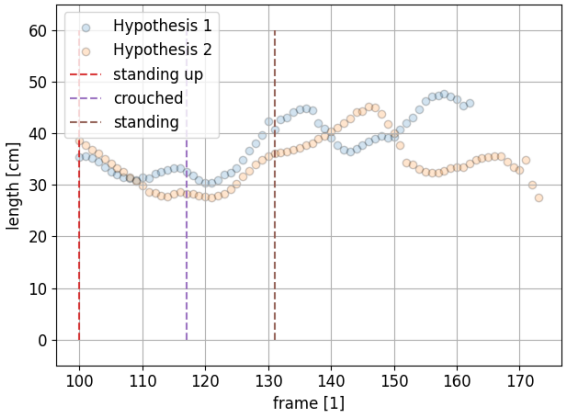


Figure 111: Left Thigh

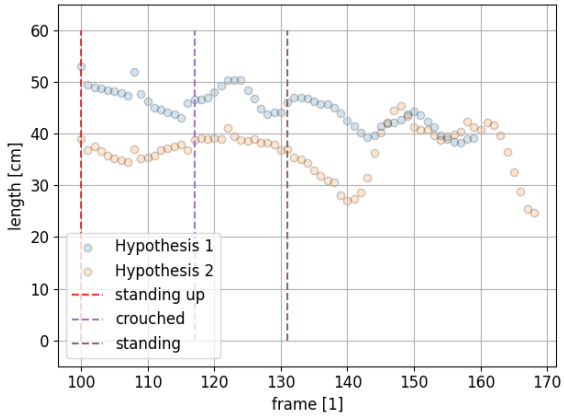


Figure 112: Right Shin

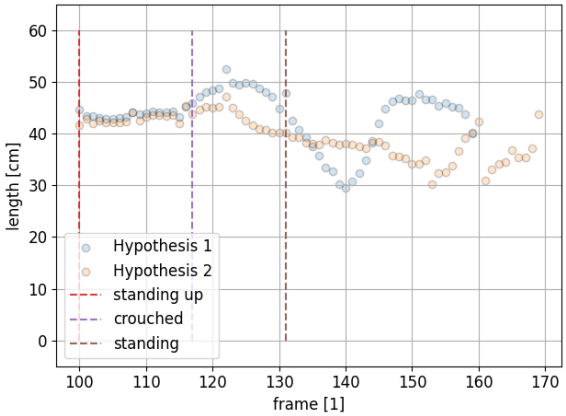


Figure 113: Left Shin

D.3 Limb Length Comparison Plots Of Lying Down

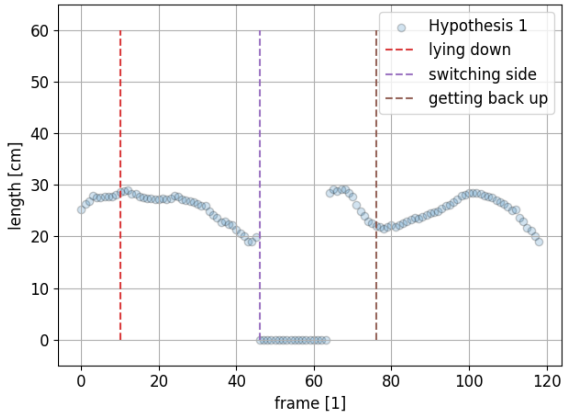


Figure 114: Nose - Right Shoulder

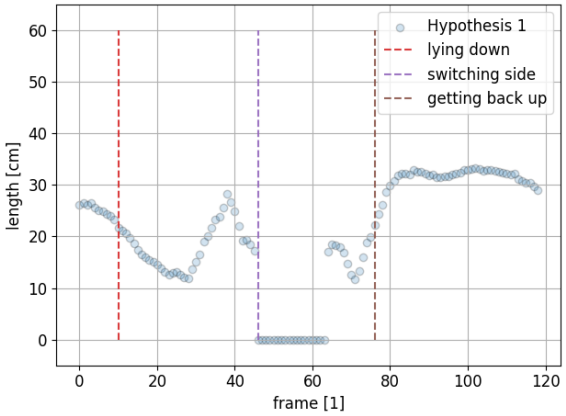


Figure 115: Nose - Left Shoulder

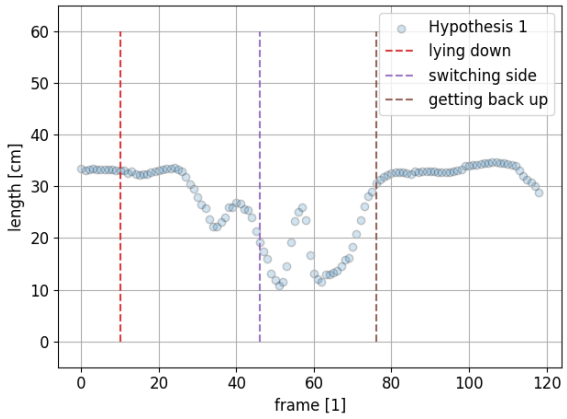


Figure 116: Shoulder Line

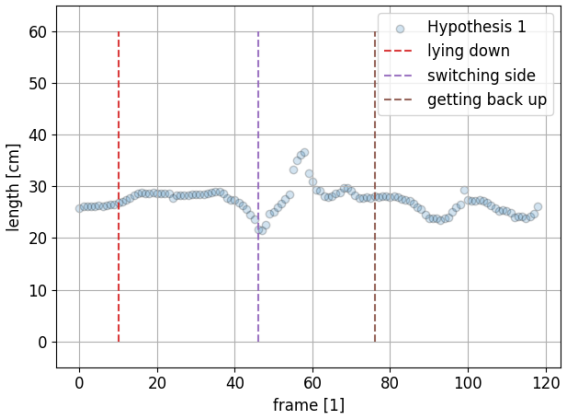


Figure 117: Right Brachium

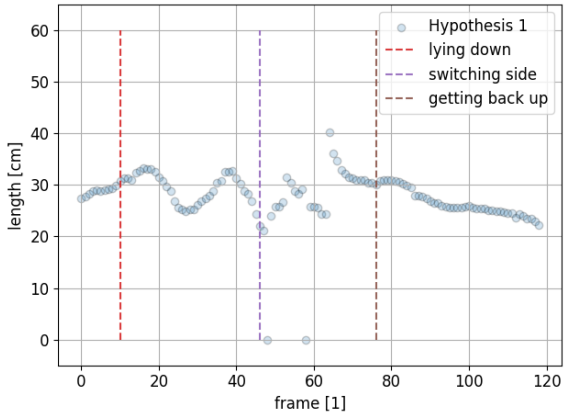


Figure 118: Left Brachium

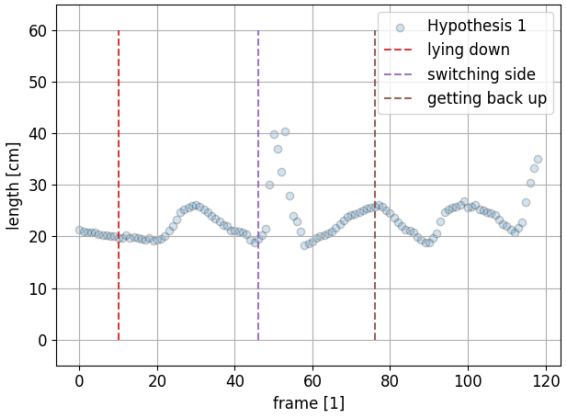


Figure 119: Right Antebrachium

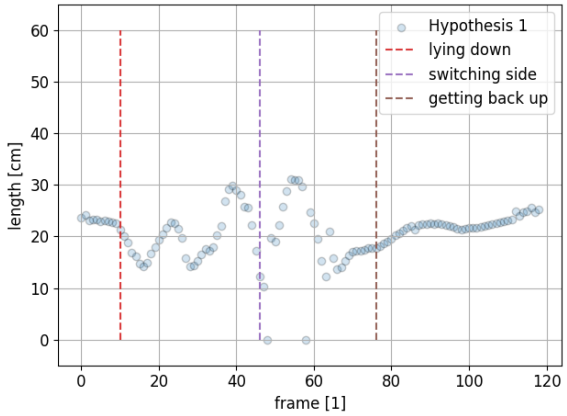


Figure 120: Left Antebrachium

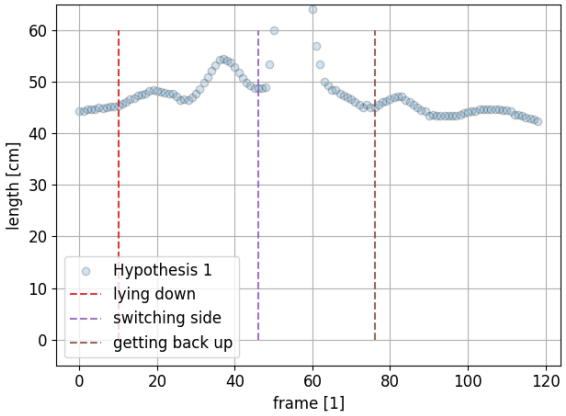


Figure 121: Right Flank

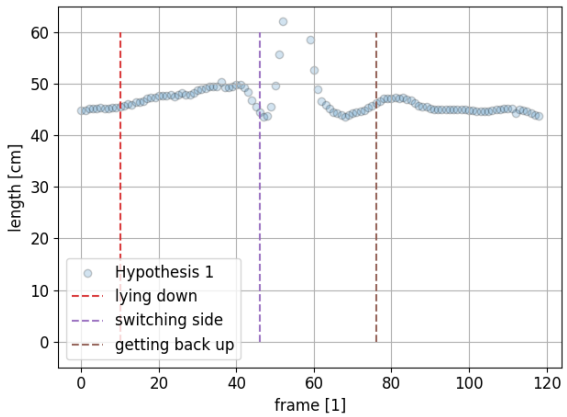


Figure 122: Left Flank

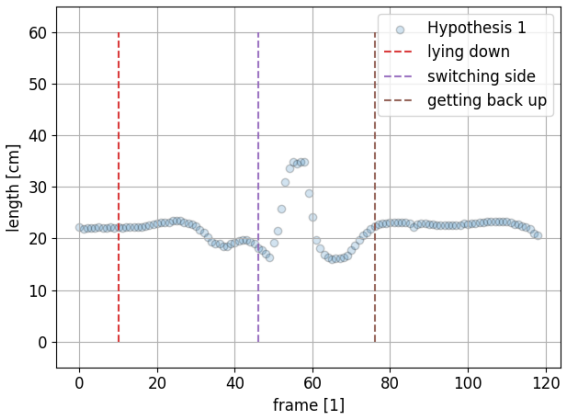


Figure 123: Waist

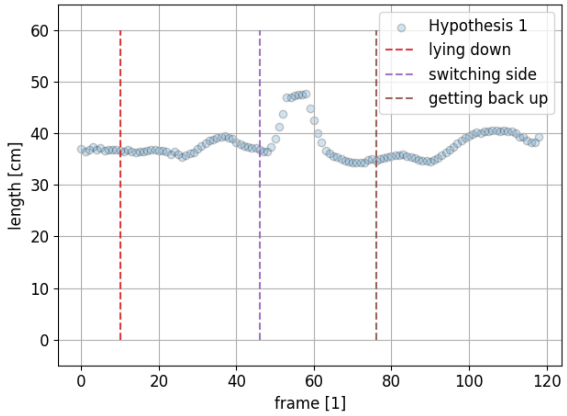


Figure 124: Right Thigh

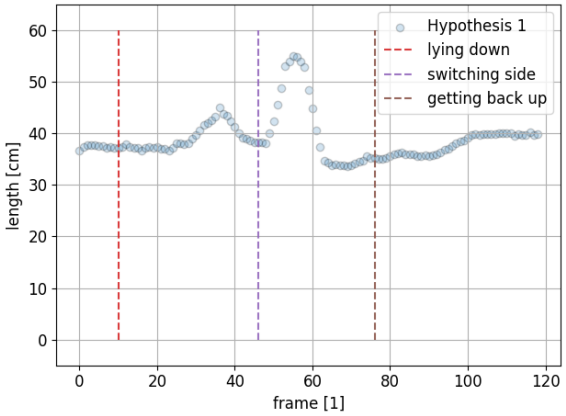


Figure 125: Left Thigh

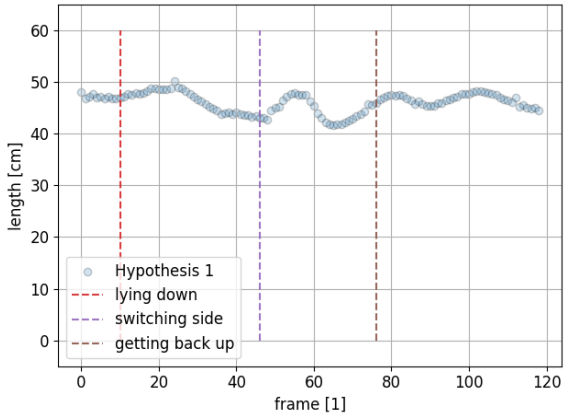


Figure 126: Right Shin

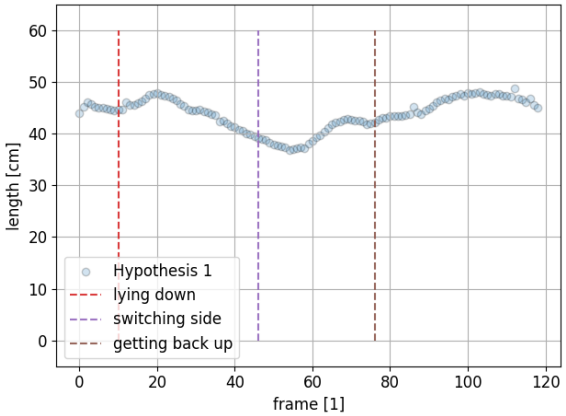
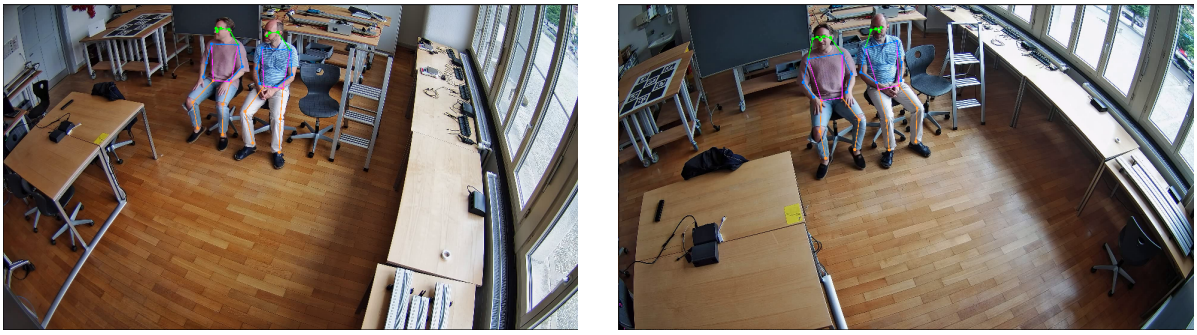


Figure 127: Left Shin

E Reconstruction of Sequences

E.1 Reconstruction of Sequence of People Standing Up



(a) Frame of camera 1.

(b) Frame of camera 2.

Figure 128: Frames of one person walking.

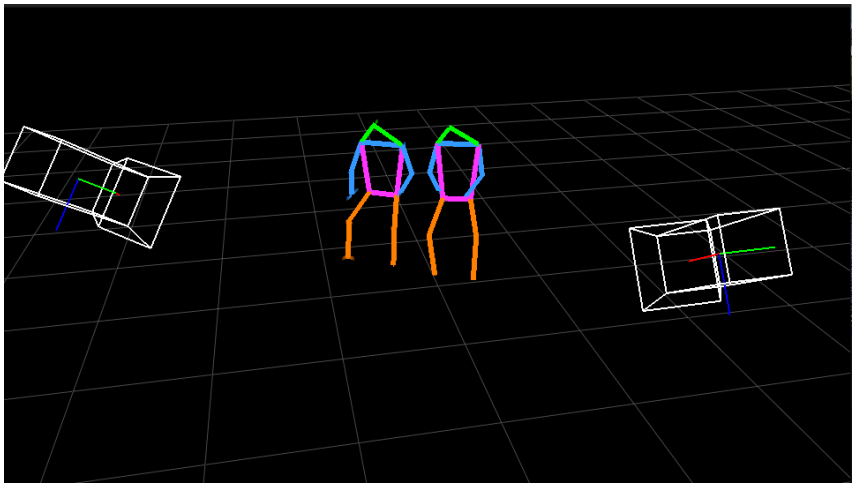
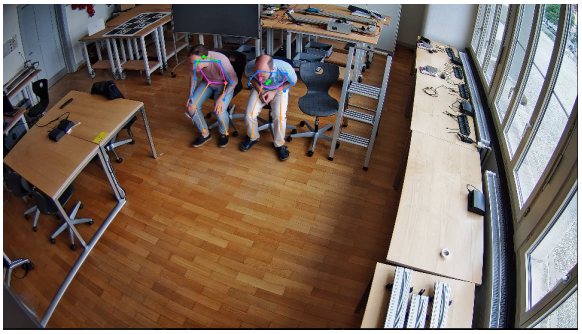
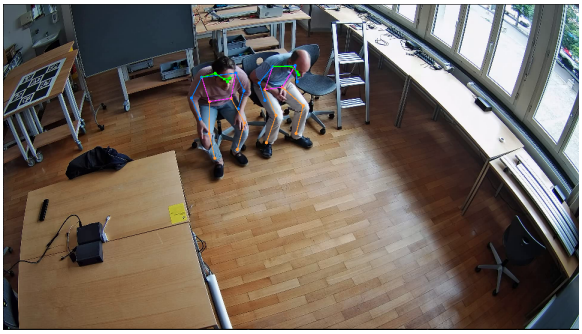


Figure 129: 3D reconstruction one person walking.



(a) Frame of camera 1.



(b) Frame of camera 2.

Figure 130: Frames of one person walking.

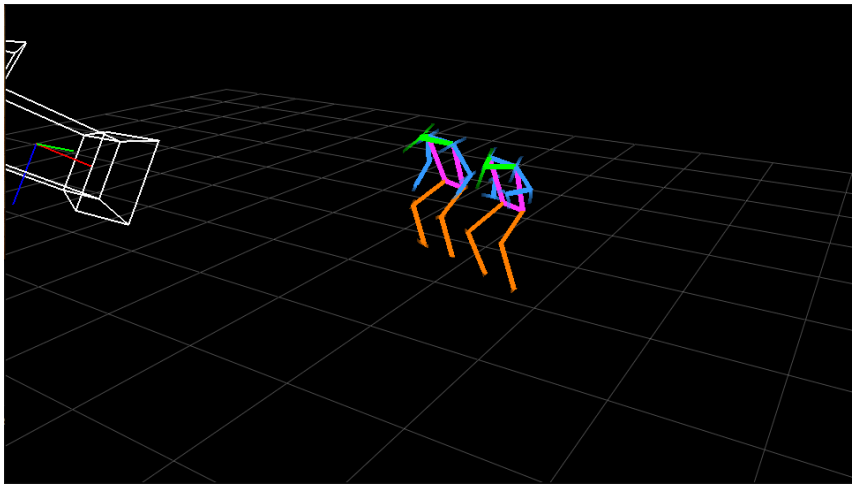
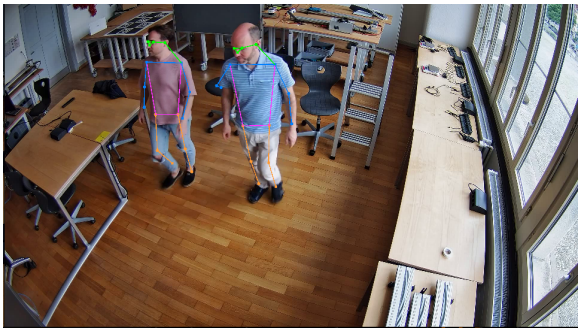


Figure 131: 3D reconstruction one person walking.



(a) Frame of camera 1.

(b) Frame of camera 2.

Figure 132: Frames of one person walking.

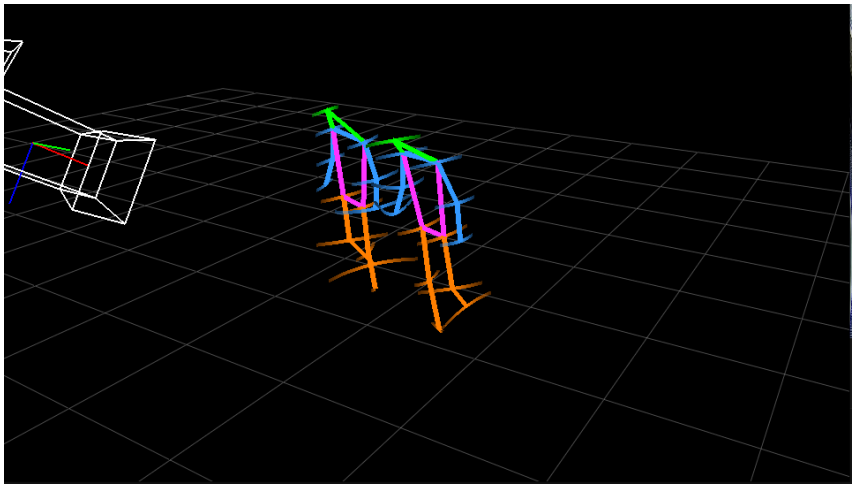


Figure 133: 3D reconstruction one person walking.

E.2 Reconstruction of Sequence of People Crossing Paths

Before crossing paths

(a) Before crossing camera 1.

(b) Before crossing camera 2.

Figure 134: Two people before crossing paths.

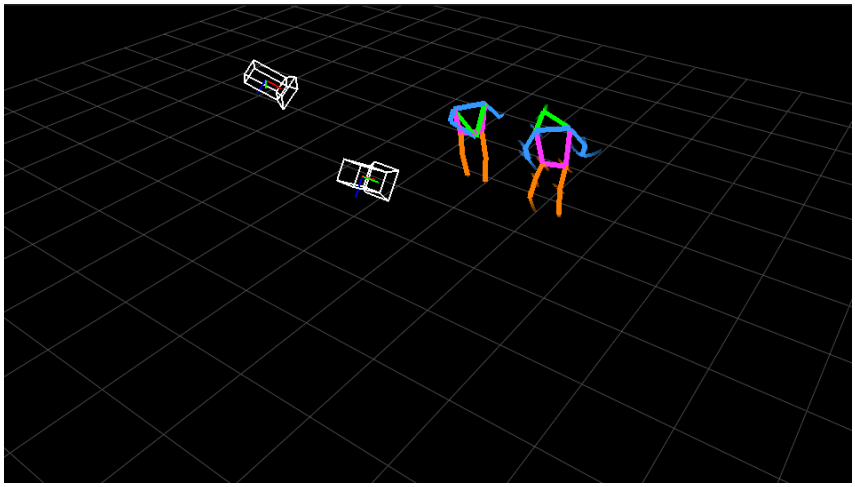
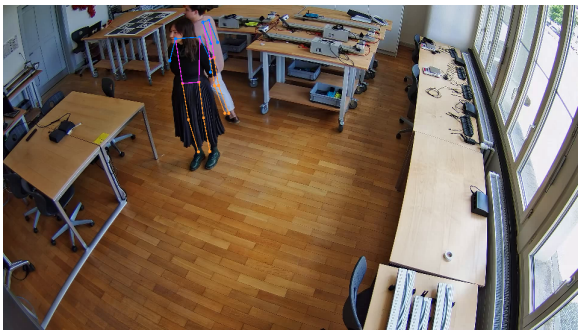
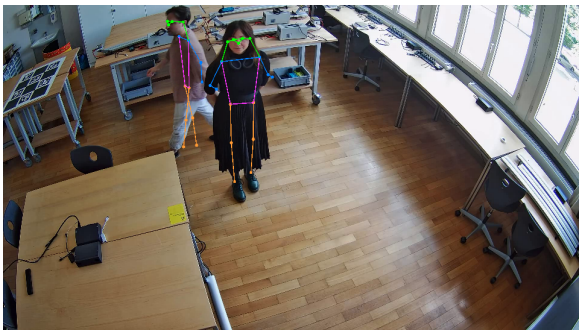


Figure 135: 3D reconstruction of two people before crossing paths.

While crossing paths



(a) Crossing camera 1.



(b) Crossing camera 2.

Figure 136: Two people crossing paths.

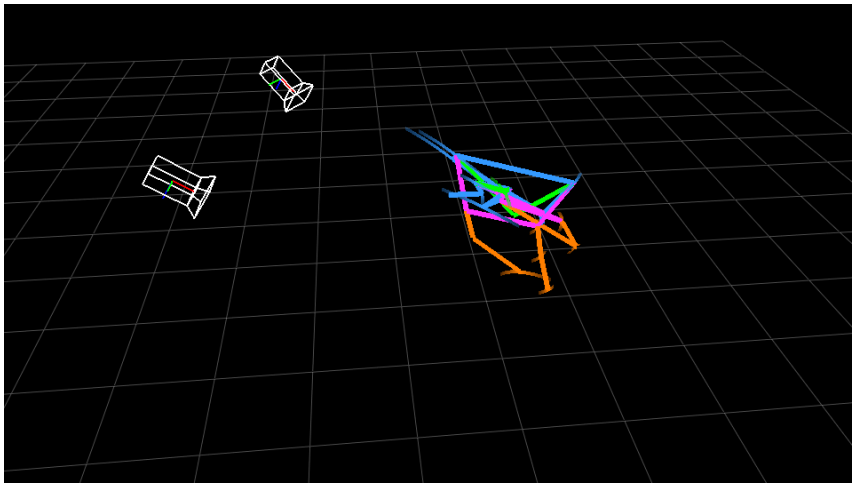
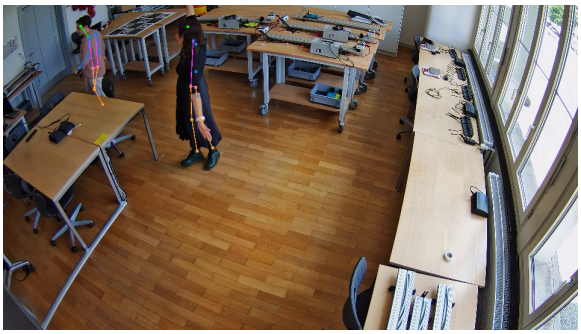
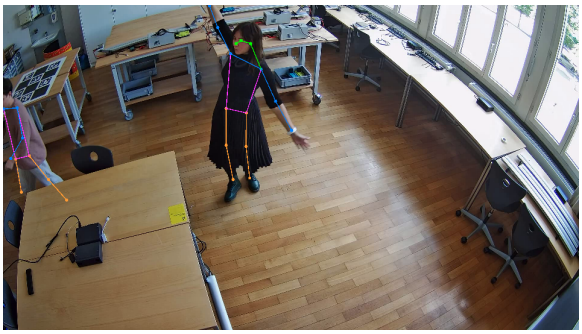


Figure 137: 3D reconstruction of two people crossing paths.

Shortly after crossing paths



(a) Shortly after Cossing camera 1.



(b) Shortly after crossing camera 2.

Figure 138: Two people shortly after crossing paths.

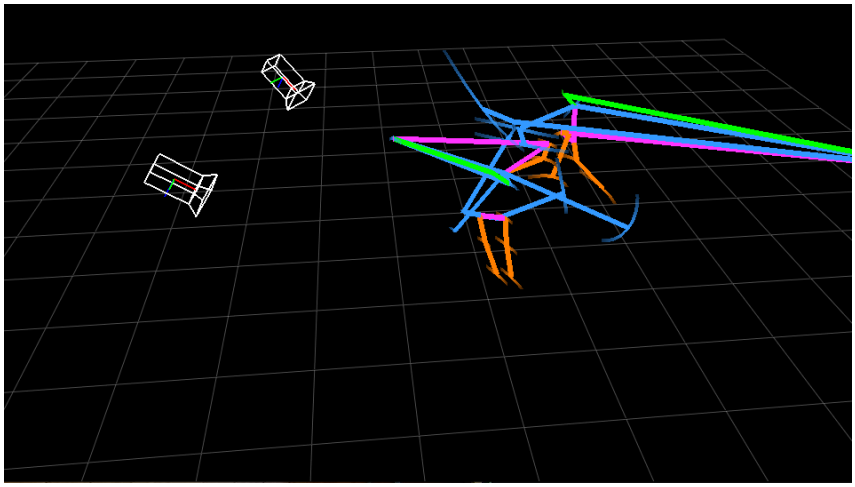
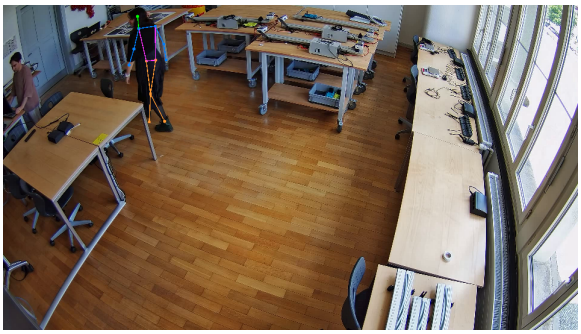
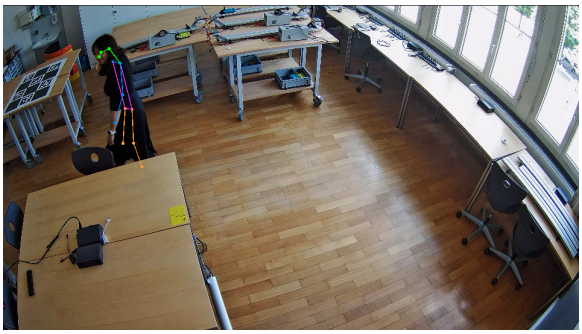


Figure 139: 3D reconstruction of two people shortly after crossing paths.

After crossing paths and one hypothesis has been removed



(a) After crossing camera 1.



(b) After crossing camera 2.

Figure 140: One person after crossing paths.

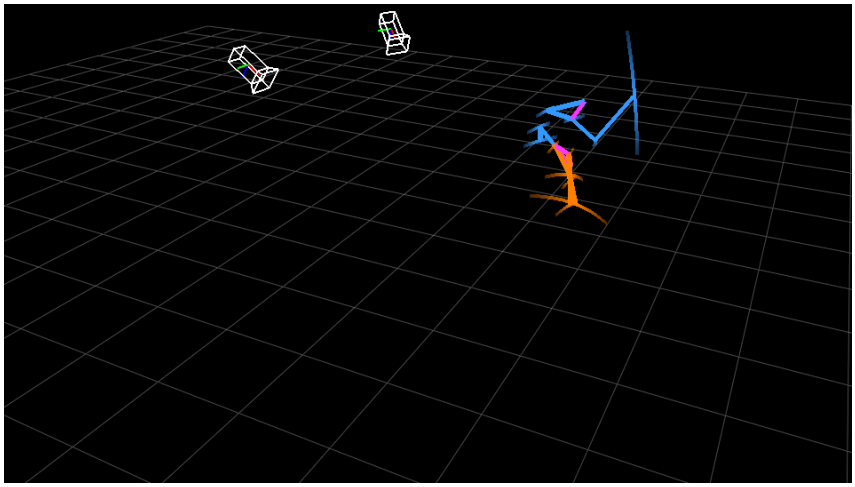


Figure 141: 3D reconstruction of one person after crossing paths.

F Speed Analysis Plots

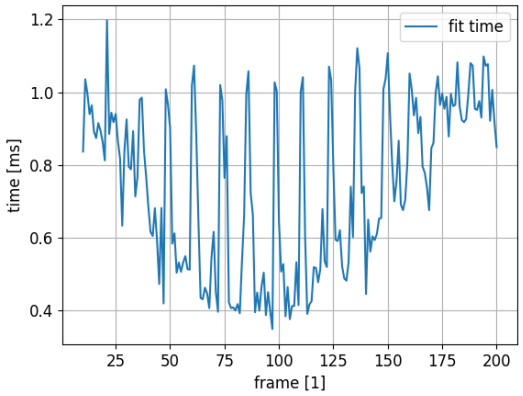


Figure 142: Fitting time of one person (ms)

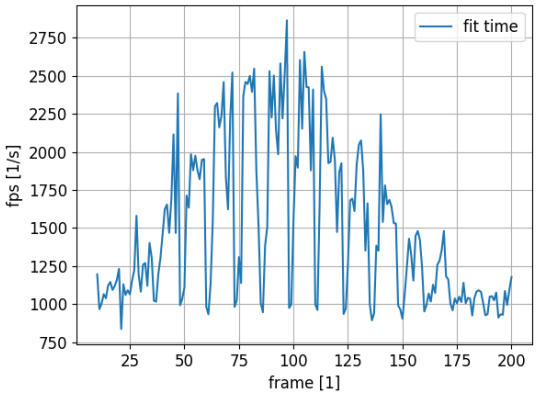


Figure 143: Fitting time of one person (FPS)

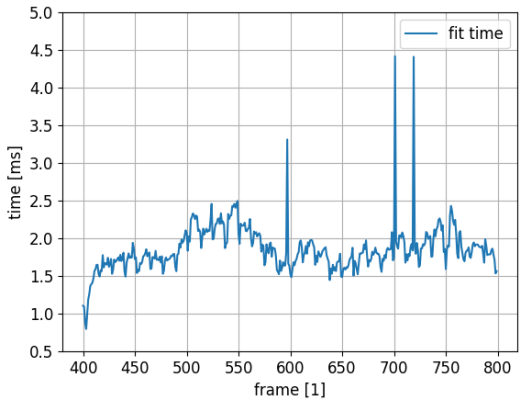


Figure 144: Fitting time of two people (ms), parameter set 1

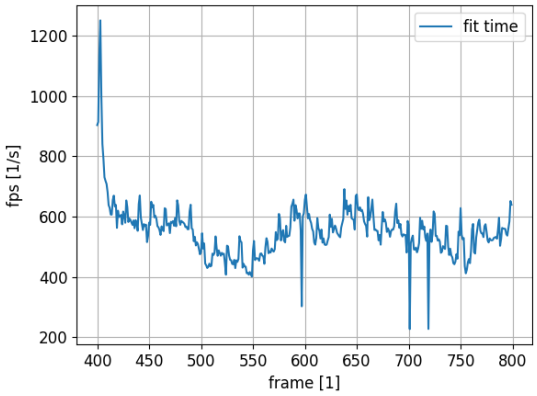


Figure 145: Fitting time of two people (FPS), parameter set 1

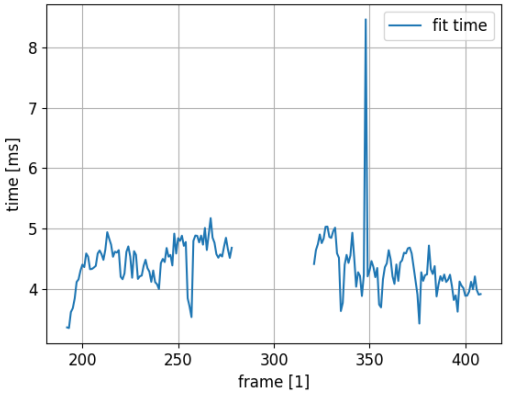


Figure 146: Fitting time of three people (ms), parameter set 1

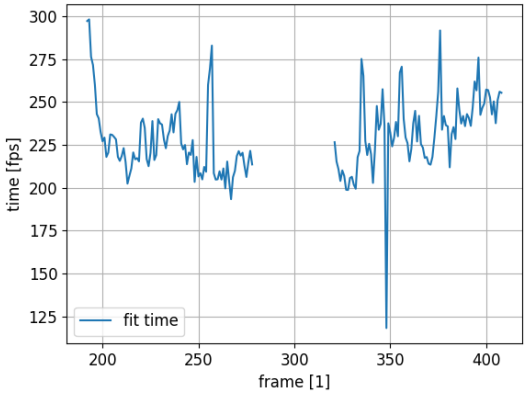


Figure 147: Fitting time of three people (FPS), parameter set 1

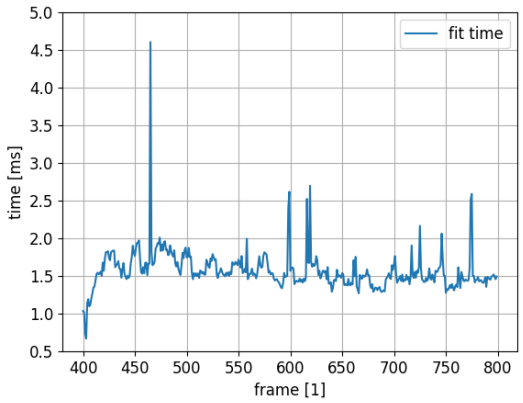


Figure 148: Fitting time of two people (ms), parameter set 2

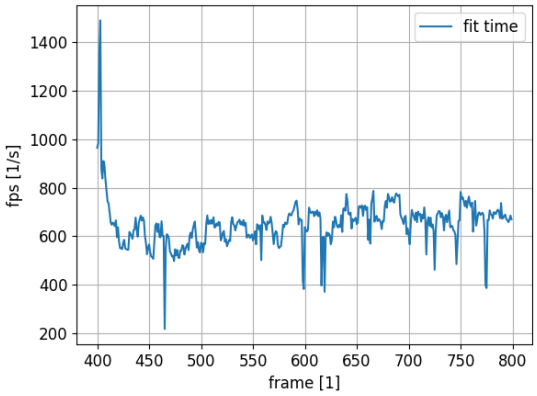


Figure 149: Fitting time of two people (FPS), parameter set 2

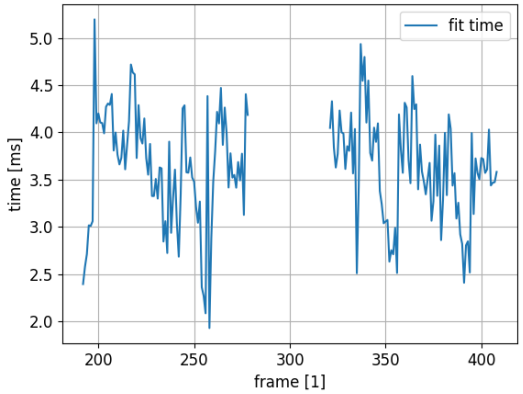


Figure 150: Fitting time of three people (ms), parameter set 2

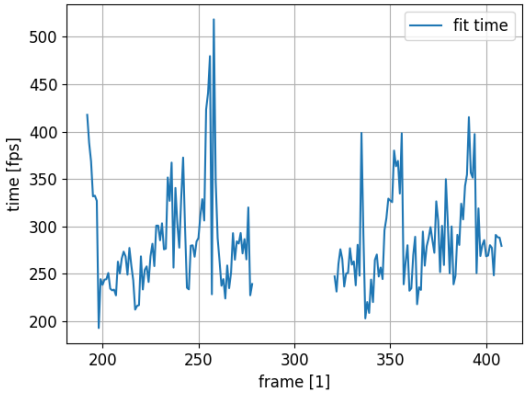


Figure 151: Fitting time of three people (FPS), parameter set 2

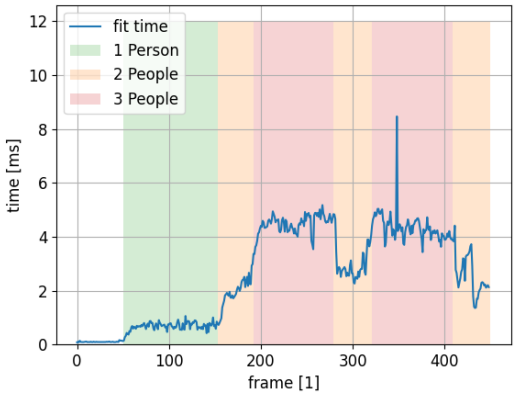


Figure 152: Fitting time of three people (ms) over whole sequence, parameter set 1

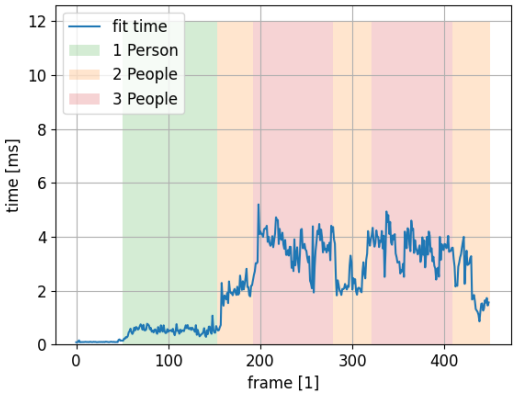


Figure 153: Fitting time of three people (FPS) over whole sequence, parameter set 2

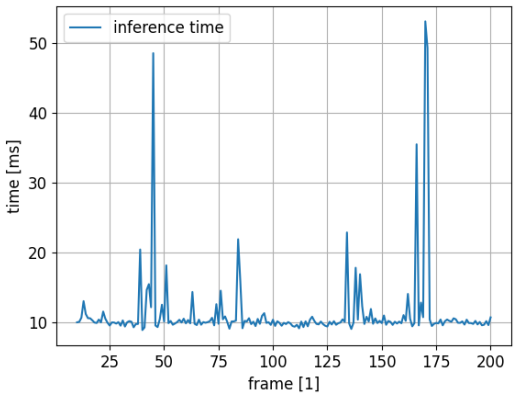


Figure 154: Inference time of one person (ms)

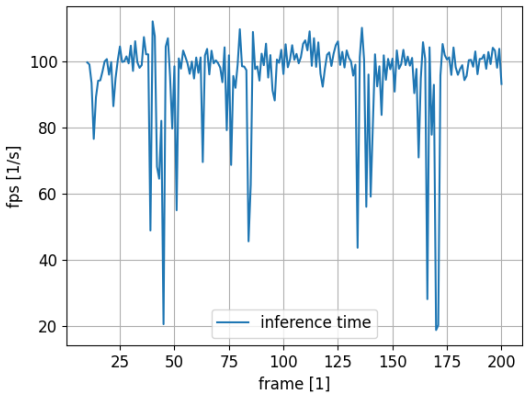


Figure 155: Inference time of one person (FPS)

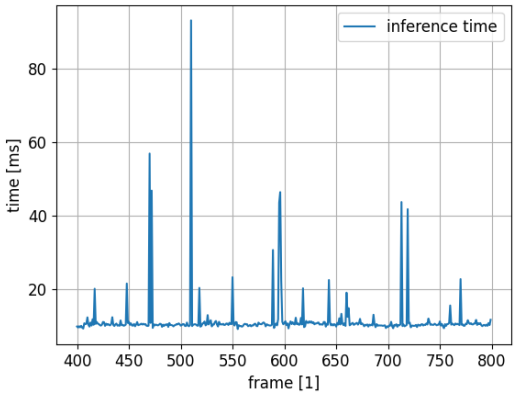


Figure 156: Inference time of two people (ms), parameter set 1

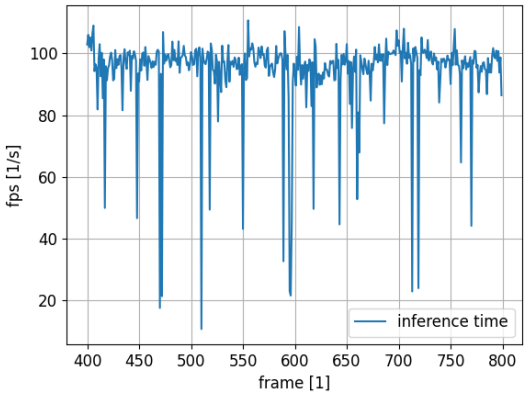


Figure 157: Inference time of two people (FPS), parameter set 1

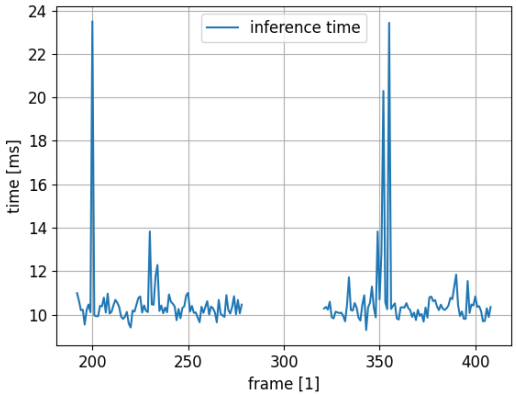


Figure 158: Inference time of three people (ms), parameter set 1

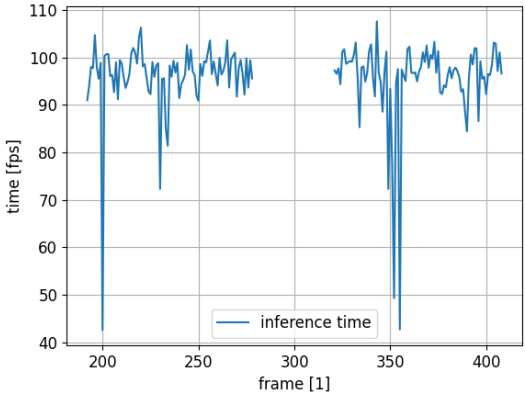


Figure 159: Inference time of three people (FPS), parameter set 1

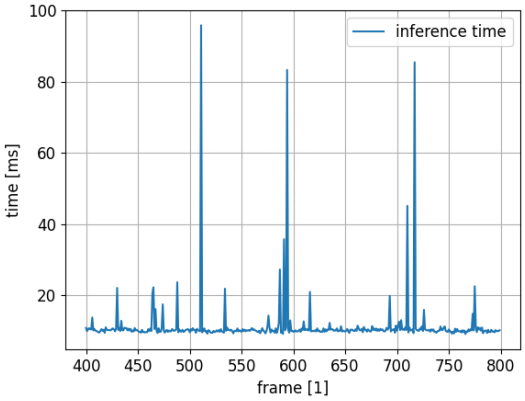


Figure 160: Inference time of two people (ms), parameter set 2

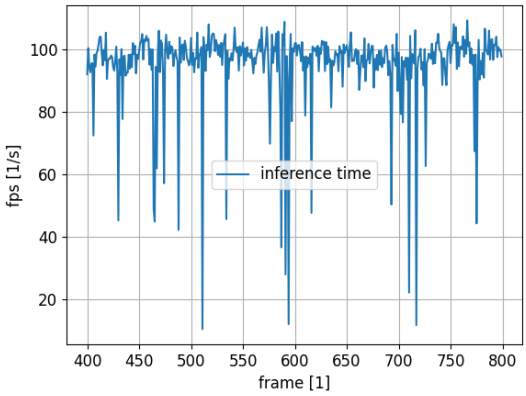


Figure 161: Inference time of two people (FPS), parameter set 2

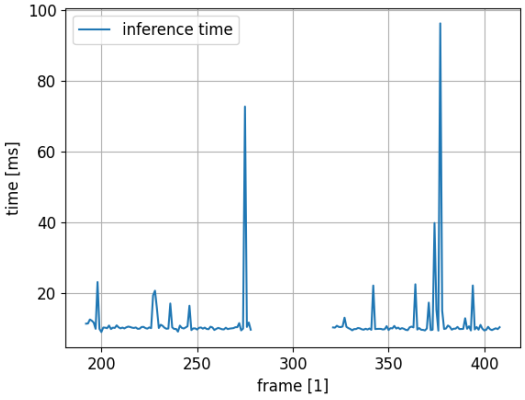


Figure 162: Inference time of three people (ms), parameter set 2

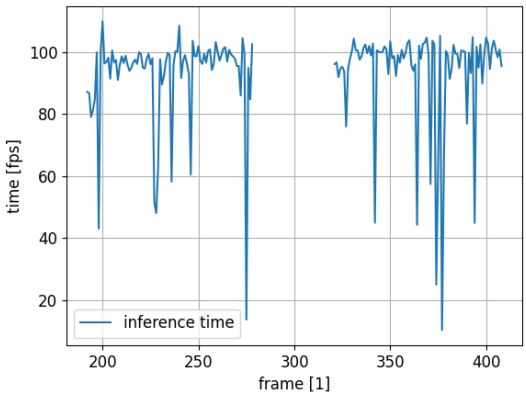


Figure 163: Inference time of three people (FPS), parameter set 2