

Hi-ROS: Open-source multi-camera sensor fusion for real-time people tracking

Mattia Guidolin^{a,*}, Luca Tagliapietra^b, Emanuele Menegatti^c, Monica Reggiani^a

^a Department of Management and Engineering, University of Padova, Stradella San Nicola 3, 36100 Vicenza, Italy

^b Henesis s.r.l., Strada Budellungo 2, 43123 Parma, Italy

^c Department of Information Engineering, University of Padova, Italy, Via Gradenigo 6/b, 35131 Padova, Italy

ARTICLE INFO

Communicated by Nikos Paragios

Dataset link: <https://doi.org/10.17605/OSF.IO/YJ9Q4>

MSC:

68M14

68T45

68U10

68W15

Keywords:

Markerless motion capture

Multi-view body tracking

Real-time

ROS

ABSTRACT

This paper presents *Hi-ROS* (Human Interaction in ROS), an open source framework focused on real-time accurate assessment of human motion. The system offers a series of tools to track multiple people in real-time by exploiting a calibrated camera network. No assumptions are made about the typology or number of cameras, nor about the body pose estimation algorithm used to extract the 3D poses of the people in the scene. The tools provided by *Hi-ROS* include a *Skeleton Tracker* to ensure temporal consistency of the detected poses, a *Skeleton Merger* to fuse the tracks from multiple cameras, thus limiting flickering phenomena, a *Skeleton Optimizer* to ensure limb length consistency, and a *Skeleton Filter* to perform real-time smoothing of the detected joint trajectories. Accuracy, tracking robustness, and real-time performance of the proposed system were evaluated on a public dataset, containing both single-person and multi-person sequences with up to 4 people interacting. The results obtained using different subsets of the proposed tools show how the complete *Hi-ROS* pipeline provides accurate and reliable estimates also in challenging scenarios, with a reduction of the RMSE of up to 27% with respect to a pure tracking approach. This work aims to push forward the development of unobtrusive human-robot interaction applications, multi-person automated posture analyses, rehabilitation performance assessments, and any possible application enabled by real-time accurate assessment of human motion via markerless motion capture.

1. Introduction

The ability to recognize and track human movements is of paramount importance in diverse fields, such as surveillance, human-robot interaction, telerehabilitation, and autonomous driving. Moreover, new emerging trends in Industry 4.0, such as the use of cobots to support workers in their tasks (Vysocky and Novak, 2016; Bragança et al., 2019) and the active monitoring of operators to provide online ergonomic feedback (Lim and D'Souza, 2020; Battini et al., 2022), require real-time knowledge of the operators' poses. The usage of inertial suits or optoelectronic systems to assess motion might not be viable in such contexts due to their high costs and complex setups, leaving markerless motion capture as the only feasible option.

The estimation of human motion without the aid of any body-mounted sensor or marker has been an intensive research topic for decades. Despite the improvements achieved in the latest years thanks to machine learning and deep learning approaches (Toshev and Szegegy, 2014; Newell et al., 2016; Chen et al., 2018), real-time accurate assessment of human motion via markerless motion capture remains an open problem. Common challenges come from background

clutters, limited fields of view, occlusions, and the general difficulty of tracking the human body, a system characterized by a large number of degrees of freedom and prone to self-occlusions.

One promising technology to mitigate such issues is to exploit a distributed network of RGB-D (red-green-blue-depth) cameras that can acquire colored point clouds. By fusing the partial information coming from each camera, it is possible to increase stability, accuracy, and reduce occlusions, allowing to obtain stable 3D reconstructions of the subjects' movements. Thanks to the latest performance improvements for markerless body pose estimation (e.g., Cao et al., 2019), in combination with relatively high frame rates in recent RGB-D cameras, simpler tracking approaches can achieve high accuracies without the overhead required by more sophisticated tracking algorithms (Bochinski et al., 2017). In this context, a promising approach is to take advantage of the tracking-by-detection paradigm. Each camera is considered an independent detector that extracts information on the poses of the people being seen. Then, a single tracker takes as input all the detections and estimates the full trajectories of each person's movements via data association.

* Corresponding author.

E-mail address: mattia.guidolin@unipd.it (M. Guidolin).

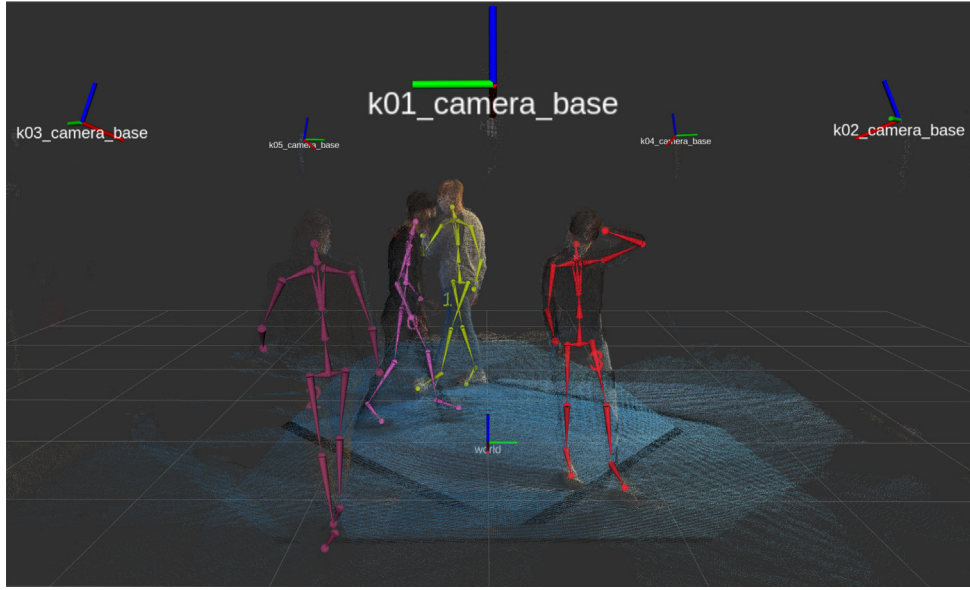


Fig. 1. Output of the *Hi-ROS* framework while tracking 4 people simultaneously. The input detections are obtained by exploiting the *Microsoft Azure Kinect Body Tracking SDK* (Microsoft, 2021).

In this work, we present *Hi-ROS* (Human Interaction in ROS), an open-source framework for real-time accurate assessment of human motion based on the tracking-by-detection paradigm. The proposed framework offers a series of advanced tools to enhance accuracy and robustness of multi-person markerless body pose estimation and tracking when exploiting a calibrated camera network. *Hi-ROS* is the result of years of experience in markerless body pose estimation and tracking (Munaro et al., 2016; Malaguti et al., 2019). The proposed framework was designed with four major features in mind: (1) maximum accuracy of the estimated body poses, (2) simplicity of use, (3) real-time performance, and (4) generality, requiring the least possible number of assumptions. In fact, no assumptions are made about the typology or number of sensors that are used, nor about the detection algorithm that extracts the 3D poses of the persons. Fig. 1 shows an example output of the proposed framework while tracking 4 people.

The *Hi-ROS* framework is available at github.com/hiros-unipd under the Apache v.2 license, and aims at putting the basis to allow accurate human pose estimation and tracking without the necessity of any sensor or marker on the body that might hinder a person's movements.

Accuracy, tracking robustness, and real-time performance of the system were evaluated on a public dataset (Guidolin et al., 2022). The dataset includes multiple sequences with different numbers of people interacting. The data provided include raw RGB and depth recordings obtained from a network composed of 5 synchronized Microsoft Azure Kinect cameras, as well as accurate measurements of the participants' full-body kinematics via inertial motion capture. Due to the specific hardware and software used during the acquisitions, inertial body poses are estimated with an accuracy comparable to that of state-of-the-art marker-based optoelectronic systems, as demonstrated in Schepers et al. (2018).

The remainder of the paper is organized as follows. Section 2 analyzes other works related to markerless body pose estimation and tracking. Section 3 describes in detail the proposed framework. Section 4 discusses the experiments performed and the results obtained in terms of accuracy, tracking robustness, and real-time performance. Finally, Section 5 draws the final remarks and concludes with the improvements we plan to include in the future.

2. Related work

The release in recent years of affordable low-cost RGB-D sensors, like the Microsoft Kinect cameras, together with the growing importance of knowing a person's full-body pose in multiple fields, brought increasing attention to markerless human body pose estimation and tracking. This resulted in various works that focused on providing reliable estimates of human pose in real-time (Mehta et al., 2017, 2019; Martínez-González et al., 2018; Cao et al., 2019; Rim et al., 2020). While single-camera human body tracking has been made accessible by exploiting such tools, an easy-to-use accurate framework for real-time multi-camera multi-person skeletal fusion and tracking is still missing.

Multi-camera people tracking systems can be divided into two groups: works that focus on leveraging 2D information to obtain 3D body poses and works that rely on RGB-D camera networks to increase the accuracy of the single-camera 3D detections.

2.1. Estimation of 3D poses from 2D data

Most of the works that fall into this category utilize a calibrated network of RGB cameras to extract 2D body poses of the persons in a common scene from different viewpoints. Knowing the pose of each camera with respect to a global reference frame, it is possible to extract the 3D body poses of the people based on triangulation. However, when dealing with multiple people, a prior tracking step is required to match the correct 2D points extracted by each camera with the correct 3D track. That is, multiple detections obtained from different cameras, but referring to the same person, need to be correctly associated, allowing to separate the sets of keypoints describing each subject.

Chen et al. (2020) proposed an iterative process for estimating 3D poses from 2D body joints obtained by multiple cameras in real-time. Each camera stream is considered independent and does not require to be synchronized with the other ones. For this reason, the authors proposed an enhanced triangulation algorithm that does not require the 2D points of each view to be acquired simultaneously. Multi-person tracking is achieved by solving a weighted bipartite graph matching problem, where its affinity matrix is computed both from 2D and 3D geometric correspondences between each detection and track.

Reddy et al. (2021) also presented a method for 3D pose estimation and tracking from an arbitrary number of camera feeds. The proposed system used a 4D convolutional neural network to produce a short

time description of the people to be tracked. Tracking, in this case, is achieved by maximizing the total score of a cost matrix, where each element is computed as the inner product between pairs of descriptors. No information on the computation time is provided.

Chu et al. (2021) tackled the same problem by taking advantage of temporal consistency to match the 2D poses obtained during each time frame with the previously estimated 3D skeletons. The affinity of a 2D detection to a 3D track, in this case, is measured with the geometric constraints of the projection difference between the 2D pose and the reprojection of the 3D pose in the specific camera view. Once the affinity matrix is computed, the corresponding weighted bipartite problem is solved in real-time by exploiting the Hungarian algorithm. The authors also included a preprocessing joints filter step that allows to remove outliers by computing the distance between each joint's epipolar line and the corresponding point and then comparing it to a predefined threshold.

The main focus of all previous works is on computing accurate 3D estimates of the human pose. However, none of them considers the articulated object being tracked as a person. Including prior information of the body being tracked (e.g., by requiring a person's limb lengths not to vary between consecutive frames) can be a key feature to further increase markerless motion capture accuracy.

In this regard, Bultmann and Behnke (2021) presented a novel method for real-time estimation of 3D human poses from a multi-camera setup. The 3D poses are recovered from 2D joints based on triangulation and refined by exploiting a body model that incorporates prior knowledge of the human skeleton. Multiple person detections are associated across camera views based on the epipolar distance of their joints. Then, each 3D skeleton is further optimized by exploiting prior information on the typical bone lengths of the human skeleton. However, such information is not considered as person-specific; thus, it does not take into account significant differences that can be present among different persons' body proportions.

Finally, this category can include works on multi-camera and multi-person pedestrian tracking. Pedestrian tracking typically refers to understanding (and, possibly, predicting) the behavior of people as seen by one or multiple RGB cameras. In such a context, the focus is typically on the estimation of each person's centroid and of a bounding box containing the body. However, recent works started to integrate higher-level information, such as the keypoint positions describing the detected people's full-body poses.

Zhao et al. (2019) proposed a system that exploits the human pose to improve the accuracy of pedestrian detection and localization. However, the human pose takes into account torso and lower limbs only. Then, such data is used to extract both joint-indexed features and symmetry-indexes features, allowing to refine the original detections.

Similarly, Jiao et al. (2020) focused on the problem of occlusions in the video feeds and proposed to exploit body keypoints data to improve the detection accuracy. By knowing the human pose, together with the visual information, they show how it is possible to reduce the number of false positive failures in pedestrian detection.

Despite using information describing the pose of a person, these works do not take into account the pose estimation accuracy, nor it is the primary focus of their research. In fact, high pose estimation accuracy might not be necessary to achieve accurate tracking results. While this is true in the context of pedestrian tracking, the same cannot be applied to different fields, such as in human-robot cooperation, where body pose estimation accuracy, together with real-time performance, are of paramount importance. Thus, in this work, the main focus is on maximizing the full-body pose estimation accuracy, independently of the number of cameras and tracked people, and in real-time. To this end, accurate multi-camera and multi-people tracking is the first step required to achieve such an objective.

2.2. Enhancement of 3D poses from 3D data

Works that rely on RGB-D camera networks do not require triangulation to compute the 3D poses of the persons in the scene. In this case, the main benefit of using multiple cameras is the lower sensibility of the system to occlusions. This allows the achievable accuracy to be increased, since missing information from one sensor might be available from another.

In this context, Moon et al. (2016) used multiple Kinect sensors to correct inaccurate tracking data from a single camera. The developed system exploits a Kalman filter for real-time 3D human skeleton tracking, where the measurement noise vector is adjusted according to the reliability of each detection. In this case, however, the focus is on data fusion only, since the system only supports a single person's body tracking.

Kadkhodamohammadi et al. (2017) also proposed a 3D human pose estimation system from multi-view synchronized RGB-D images that do not require prior knowledge of the number of persons in the scene. The multi-view fusion is solved as an iterative process where, for each time frame, the two closest skeletons that do not originate from the same view are merged until no pair of merging candidates are left. Then, a multi-view energy function is used to drive the body parts towards their optimal locations. The energy function takes into account how much the estimated body lengths differ from the average lengths computed during a training phase. However, such average lengths are computed across their entire training dataset, thus not being person-specific and suffering from the same limitations as in Bultmann and Behnke (2021). Additionally, no information is provided on the computation time required by the optimization step.

Liu et al. (2018), on the other hand, presented a work on human action recognition using a distributed calibrated RGB-D camera network. Even though the focus is not on multi-camera tracking, the authors proposed the usage of an information-weighted consensus filter to fuse the 3D skeleton detected by each camera. However, no tracking is performed, since the system only supports a single person.

Ryselis et al. (2020) also exploited multiple Kinect cameras to monitor key performance indicators in physical training. To achieve higher accuracy, the authors used 3 cameras to provide complete spatial coverage of the subject. Data fusion in this case is accomplished by calculating the averages of the joint coordinates estimated by each camera projected in a global reference frame. Also in this case, however, no tracking is performed, since the system only supports the analysis of a single subject.

Zhou et al. (2021) proposed to integrate a Tobit Kalman filter (TKF) and a Differential Evolution (DE) algorithm to improve the accuracy of human motion estimation. The TKF allows ensuring kinematically admissible poses, while the DE optimization aims at minimizing the bone length variability, where the reference lengths are obtained from an initial application of the TKF. However, the system supports a single input source and only allows the analysis of a single person's pose. No information on the required computation times is provided.

A general framework for multi-RGB-D camera systems was proposed by Fender and Müller (2018). The framework can stream and process multiple RGB, depth, and skeleton streams in real-time. The authors claim to perform skeleton fusion based on distance. To disambiguate the cases in which the skeletons are properly detected with respect to those in which the skeleton is rotated 180°, body estimation is combined with face tracking. Since skeletal tracking is not the main focus of the framework, no other details are provided. The framework was planned to be released as open-source, but, to the best of the authors' knowledge, it is still not available to the public.

Finally, in our previous work (Malaguti et al., 2019), we presented an improved version of *OpenPTrack*'s skeletal tracking (Munaro et al., 2016), an open-source software for real-time multi-camera people tracking in RGB-D camera networks. The tracking algorithm was enhanced with an improved version of the Kalman filter to ensure better

temporal consistency of the body poses and an adaptation mechanism to avoid fast changes in the skeletons' limb lengths. However, the system is dependent on an internally developed human pose estimation algorithm and only supports specific sensors.

The literature review shows that, although markerless people tracking has been tackled by exploiting a variety of approaches, most works suffer from a series of limitations. Among all, real-time performance is typically omitted, very few open-source implementations are available, and additional constraints due to the specific characteristics of each person's body are often missing. In this work, we propose a system that aims to overcome such limitations. The proposed framework is capable of tracking multiple people in real-time, without requiring any assumptions on the number, typology, and frame rate of the cameras being used. Information from multiple cameras allows for the detection of errors in the raw estimated poses and an enhancement of the overall system accuracy, strongly reducing the impact of occlusions. Moreover, information on each person's body dimensions can be fed to the system to ensure anatomically correct estimates of the poses. Finally, all the developed tools are available at github.com/hiros-unipd under the Apache v.2 license.

3. System design

This work aims at providing robust and reliable 3D tracking of human motion in real-time via markerless motion capture. To this end, the proposed framework requires a network of N cameras that act as detectors. No assumptions are made about the number, typology, and frame rate of the cameras. The only requirement is the extrinsic calibration of the camera network. That is, after defining a global reference frame \mathcal{G} , the transform $T_{C_k}^{\mathcal{G}}$ between each k -th camera C_k and such frame \mathcal{G} must be known.

Each camera represents an independent detector, which is in charge of performing markerless 3D body pose estimation of the people in the scene. No assumptions are made about the algorithm used. The only requirement is to be able to use a specific detector in the proposed framework is a bridge in charge of publishing the detected skeletons as ROS messages. In fact, as the *Hi-ROS* name suggests, we chose to use ROS (Quigley et al., 2009) as the software foundation for this framework. Such decision was based on two main reasons. First, it is not operating system specific, allowing to use *Hi-ROS* on Windows, Linux, or Mac. Secondly, due to its extensive usage, ROS has gained the role of de facto standard in advanced robotics applications. For this reason, a variety of camera manufacturers support ROS, offering open-source drivers to interface with their hardware.

Hi-ROS currently consists of 4 main modules:

1. A *Skeleton Tracker* node (Section 3.1) implementing robust skeletal tracking of the detections obtained by each camera. This node ensures temporal consistency of the detected skeletons, by assigning each person a unique ID that is retained in time.
2. A *Skeleton Merger* node (Section 3.2) taking as input the tracked skeletons computed by the *Skeleton Tracker* and performing a fusion of the poses seen by all the cameras during the same time frame. The node is designed to reduce the flickering obtained when data from a distributed camera network are merged. The node can properly handle and merge partial skeletons, as long as at least one common keypoint is available.
3. A *Skeleton Optimizer* node (Section 3.3) performing a global optimization on each person's poses. After an initial calibration of the body dimensions, this tool ensures limb length consistency throughout the whole experiment, also allowing to detect and remove outliers. This node introduces kinematic constraints on the tracked body poses, after integrating prior information of the persons' body dimensions in the system.

4. A *Skeleton Filter* node (Section 3.4) containing efficient real-time state-space and Butterworth filters that can be used in cascade to any of the previous nodes. This node requires each skeleton to already have a unique ID (i.e., some prior form of frame-by-frame tracking must be performed), in order to obtain the temporal evolution of each person's pose. The filter's cutoff frequency and order can be manually tuned to obtain smoother trajectories of the detected keypoints, addressing the needs of different applications.

Fig. 2 shows an overview of the proposed system. The white blocks represent a generic number of detectors, each running a body pose estimation algorithm based on its camera's data. Then, each detection is expressed as a *SkeletonGroup* (Section 3.5) and fed to the *Skeleton Tracker*. The tracked skeletons can be used as is or fed to *Merger*, *Optimizer*, or *Filter*. The merged skeletons can in turn be the input to the *Optimizer*, or to the *Filter*, if the optimization is not needed by the application.

This section analyzes in detail the components proposed within the *Hi-ROS* framework: *Skeleton Tracker* (Section 3.1), *Skeleton Merger* (Section 3.2), *Skeleton Optimizer* (Section 3.3), and *Skeleton Filter* (Section 3.4). Finally, Section 3.5 details how a person's pose is represented in the *SkeletonGroup* messages used for the communication among all the nodes of the framework.

3.1. Skeleton tracker

The *Skeleton Tracker* is in charge of taking as input all the camera's detected poses and associating each skeleton to the correct track. This is achieved by solving an assignment problem, where each element of the cost matrix is computed as the distance between the body poses of all possible pairs of detected skeletons at time t and tracked skeletons at time $t - 1$.

3.1.1. Definitions

Let a skeleton group (SG) be defined as the set of skeletons in the scene at time t , expressed with respect to a generic reference frame \mathcal{F} :

$$SG_t^{\mathcal{F}} = \{S_{t,n}^{\mathcal{F}} \mid n = \{0, \dots, N - 1\}\} \quad (1)$$

where N is the total number of people detected.

Each skeleton (S) is defined as:

$$S_t^{\mathcal{F}} = \{\mathbf{m}_{t,p}^{\mathcal{F}}, \mathbf{l}_{t,q}^{\mathcal{F}} \mid p = \{0, \dots, P - 1\}, q = \{0, \dots, Q - 1\}\} \quad (2)$$

where $\mathbf{m}_{t,p}^{\mathcal{F}}$ is a set of markers and $\mathbf{l}_{t,q}^{\mathcal{F}}$ a set of links. P is the total number of markers defining the skeleton, while Q is the total number of links connecting pairs of markers. Fig. 3 shows a representation of a possible skeleton consisting of 21 markers and 20 links.

Due to the generality of the definition, a skeleton group can represent either the output of each camera's body pose detector (detection group) or the output of any *Hi-ROS* module (track group). To simplify the notation, in the remainder of the discussion, the reference frame and time relative to a skeleton (group) will be omitted whenever possible. To this end, a detection group (DG) is defined as:

$$DG_{C_k} = \{D_{C_k,j} \mid j = \{0, \dots, J - 1\}\} \quad (3)$$

where $D_{C_k,j}$ is the j -th skeleton, as defined in Eq. (2), detected by the k -th camera C_k , and J is the total number of detected people.

Similarly, a track group (TG) is defined as:

$$TG_{C_k} = \{T_{C_k,i} \mid i = \{0, \dots, I - 1\}\} \quad (4)$$

where $T_{C_k,i}$ is the i -th tracked skeleton, as defined in Eq. (2), having as source the k -th camera C_k , and I is the total number of tracked people.

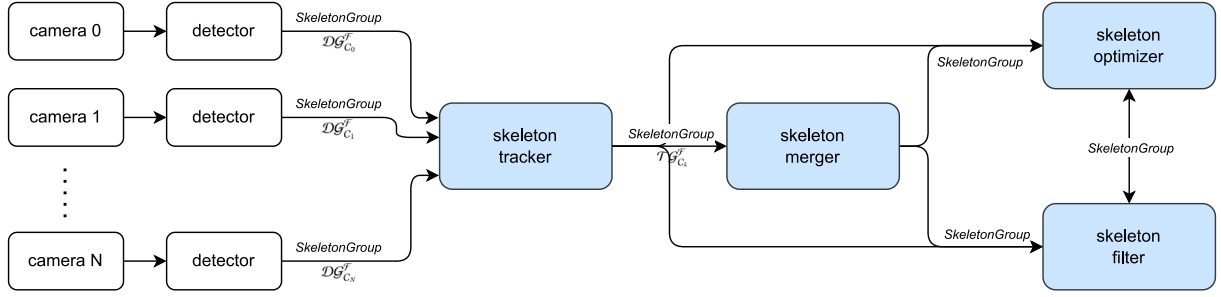


Fig. 2. Overview of the proposed system. The white blocks are independent of Hi-ROS, while the blue blocks represent the tools offered by the framework. *Merger*, *Optimizer*, and *Filter* are not mandatory, allowing for different workflows. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

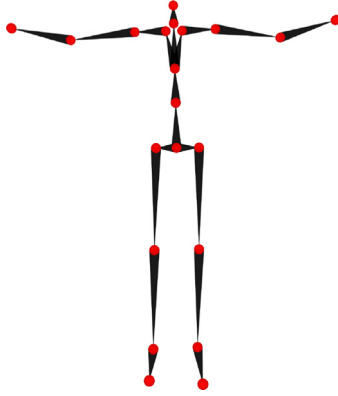


Fig. 3. Example of a Skeleton formed by 21 markers (red) connected by 20 links (black). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1.2. Tracking algorithm

Each time the *Tracker* receives a detection group DG from a camera C_k , it checks the reference frame F where data are expressed. If F differs from the global reference frame G , then the *Tracker* applies the rigid transformation $T_{F \rightarrow G}^C$ to express all data in the global reference frame:

$$\begin{aligned} DG_{C_k}^G &= T_{F \rightarrow G}^C \cdot DG_{C_k}^F \\ &= \{DG_{C_k,j}^G = T_{F \rightarrow G}^C \cdot DG_{C_k,j}^F \mid j = \{0, \dots, J-1\}\} \end{aligned} \quad (5)$$

In a network with k cameras, the transformation matrices $T_{C_k}^G$, where C_k indicates each camera's local reference frame, are the result of a camera network calibration procedure.

Each time a new detection group DG_t is received, the *Tracker* updates the latest available track group TG_{t-1} accordingly. Then, the updated track group TG_t is kept in memory to be used when the next detection group DG_{t+1} arrives. The update process is divided into three parts:

1. update the detected tracks (skeletons that are present both in DG_t and in TG_{t-1});
2. add new tracks (skeletons that are present in DG_t but were not present in TG_{t-1});
3. remove unassociated tracks (skeletons that were present in TG_{t-1} and are not present in DG_t).

The association between detections and tracks is achieved by finding the optimal solution to an assignment problem, where the cost matrix represents the distance between each detection D_j and each track projection \hat{T}_i . The cost matrix $\Delta \in \mathbb{R}^{I \times J}$ is defined as:

$$\Delta = \{\delta_{ij} = \delta_{ij}^l + \delta_{ij}^a \mid i = \{0, \dots, I-1\}, j = \{0, \dots, J-1\}\} \quad (6)$$

where δ_{ij}^l represents the linear distance between the marker positions to associate the i -th track with the j -th detection, δ_{ij}^a the angular distance between the link orientations, I is the total number of tracks at frame $t-1$, and J the total number of detections at frame t . Therefore, δ_t is a function of both the state at time t and at time $t-1$:

$$\delta_{t,i,j} = f(T_{t-1,i}, D_{t,j}) = f(\hat{T}_{t,i}, D_{t,j}) \quad (7)$$

where $\hat{T}_{t,i}$ defines the predicted pose of the i th track at time t exploiting a constant velocity model based on the pose of the i th track at time $t-1$, $T_{t-1,i}$, and $D_{t,j}$ is the pose of the j -th detection at time t .

The linear distances at time t are calculated as:

$$\delta_{ij}^l = \frac{1}{N^\alpha} \sum_{k \in \mathcal{K}} w_k (\|\hat{\mathbf{m}}_{i,k} - \mathbf{m}_{j,k}\| + w_v \cdot \|v_{i,k} - v_{j,k}\|) \quad (8)$$

where \mathcal{K} are the markers in common between the i -th track and the j -th detection, $\hat{\mathbf{m}}_{i,k}$ is the k -th marker's position of the predicted track, $\mathbf{m}_{j,k}$ the corresponding marker's position of the detection, $v_{i,k}$ the linear velocity of the k -th marker of the track, $v_{j,k}$ the corresponding linear velocity of the detection, w_k the weight associated with the marker, and w_v the velocity weight with respect to the position weight. N represents the number of markers in common, and its exponent $\alpha \geq 1$ allows to prefer track-detection matches that have a higher number of markers in common. We removed the subscript t from Eq. (8), since each element refers to the same time frame.

The distance between two markers $\|\hat{\mathbf{m}}_{i,k} - \mathbf{m}_{j,k}\|$ is calculated as the Euclidean distance between their positions.

Similarly, the angular distances at time t are calculated as:

$$\delta_{ij}^a = \frac{1}{N^\alpha} \sum_{k \in \mathcal{K}} w_k (\phi(\hat{\mathbf{l}}_{i,k} - \mathbf{l}_{j,k}) + w_\omega \cdot \|\omega_{i,k} - \omega_{j,k}\|) \quad (9)$$

where \mathcal{K} , in this case, represents the links in common between the i -th track and the j -th detection, $\hat{\mathbf{l}}_{i,k}$ is the k -th link's orientation of the predicted track, $\mathbf{l}_{j,k}$ the corresponding link's orientation of the detection, $\omega_{i,k}$ the angular velocity of the k -th link of the track, $\omega_{j,k}$ the corresponding angular velocity of the detection, w_k the weight associated with the link, and w_ω the velocity weight with respect to the position weight. In this case, N represents the number of links in common, and its exponent $\alpha \geq 1$ allows to prefer track-detection matches that have a higher number of links in common. As for Eq. (8), we removed the subscript t from Eq. (9), since each element refers to the same time frame.

The distance $\phi(\cdot)$ between two links is calculated as the angle between the two quaternions defining their orientations along the shortest path:

$$\phi(q_0, q_1) = 2 \arccos(|q_0 \cdot q_1|) \quad (10)$$

where q_0 and q_1 represent the two unit quaternions describing the links' orientations and \cdot the dot product between the two.

Each marker/link weight is composed of two terms:

$$w = w_m / l \cdot w_v \quad (11)$$

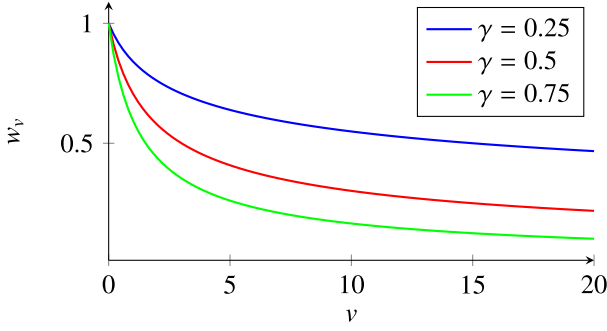


Fig. 4. Velocity weight plot for different values of γ . Velocity expressed in ms^{-1} for positions, rad s^{-1} for orientations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where $w_{m/l} = \kappa_{m/l}$ if the markers/links confidences are used ($\kappa_{m/l}$ represents the marker/link confidence), 1 otherwise; $w_v = (1 + \|v\|)^{-\gamma}$, with $\gamma \in [0, 1]$, if the markers/links velocities are used, 1 otherwise. In this way, the faster a track's marker/link is moving and the lower its distance from the detection is weighted (see Fig. 4). This choice allows not to penalize too much the distances during fast movements.

Once the cost matrix Δ is filled, the Munkres algorithm (Munkres, 1957) computes the optimal set of associations between detections and tracks that minimize the total cost (that is, the overall distance between detections and tracks). At this point, the previous tracks $\mathcal{T}_{G_{t-1}}$ are updated with the new data from the latest detections DG_t . Velocities and accelerations are updated accordingly. If some detections were not associated with any track (e.g., the number of detections at time t is greater than the number of tracks at time $t - 1$, or some detections have a too high distance with respect to any track), then new tracks are initialized accordingly. Finally, if some tracks were not associated with any detection for a certain period of time t_m (e.g., the number of tracks at time $t - t_m$ is greater than the number of detections at time t , or some tracks have a too high distance with respect to any detection), then those tracks are deleted. This choice allows to maintain the correct IDs also in case of brief full-body occlusions.

3.2. Skeleton merger

Small errors in the calibration of the camera network, as well as uncertainties in the estimation of body poses intrinsic of any detector, can result in differences in the detected poses depending on the camera and orientation in which a person is seen. By concatenating the detections from each camera without some form of filtering, non-negligible flickering can arise in the computed tracks. Kalman or generic low-pass filters are typically used in cascade to skeletal tracking to obtain smoother trajectories. However, due to the noisy nature of the input data, as well as its limited frame rate, this requires extremely low cutoff frequencies to be set, thus not allowing to correctly track fast movements. To limit the flickering phenomena without penalizing fast movements, we chose to merge the tracks describing the same person's pose seen by different cameras.

The *Skeleton Merger* node is in charge of the fusion of the tracked skeletons obtained by the *Skeleton Tracker* node. Each received track group \mathcal{T}_G is expressed with respect to the same global reference frame G , and contains the set of skeletons detected by one of the cameras C_k at time t . The *Skeleton Merger* does not require the cameras to be time-synchronized. In fact, the node determines the closest set of detections from each detector in time. Hardware time synchronization can lead to more accurate tracked poses, but it is not strictly required in the *Hi-ROS* framework.

Since the cameras are not required to be synchronized, the merged skeletons can belong to slightly different time frames. First, the average

source time \bar{t} is calculated. Then the i -th merged track is computed as:

$$\bar{\mathcal{T}}_i = \frac{\sum_{k \in \mathcal{K}} \gamma_{\mathcal{T}_{C_k,t}} \cdot \mathcal{T}_{C_k,t}}{\sum_{k \in \mathcal{K}} \gamma_{\mathcal{T}_{C_k,t}}} \quad (12)$$

where $\mathcal{T}_{C_k,t}$ is the i -th track detected by camera C_k at time t , $\gamma_{\mathcal{T}_{C_k,t}}$ its confidence (if provided by the detector, 1 otherwise), and \mathcal{K} the set of cameras that can see the track.

To further increase the pose estimation accuracy, before computing the merged skeletons, a procedure to detect outlier markers and/or links is implemented, based on a voting procedure among all detectors. First, possible flipped detections are identified by comparing the orientations of the pelvis. The detections from each camera are split into groups. Each group contains detections where the distance between the pelvis orientations is lower than $\pi/2$, where the distance is calculated following Eq. (10). If more than one group is present, it is assumed that the detections from the largest group are correct, while the others are flipped.

After removing the flipped detections, outlier estimates coming from one or more cameras are detected for each marker and/or link by following an analogous voting procedure. The orientation distances of the links are calculated following Eq. (10), while the position distances of the markers are computed as the Euclidean distance between each pair of markers.

To fuse the skeletons estimated from different cameras, the *Skeleton Merger* node constantly maintains a buffer of track groups, which is reset each time a merged skeleton is computed. Merging is triggered if at least one of the following conditions is met:

1. If the buffer contains a track group with the same source (i.e., the same detector) as the one from the newly received track group, then all the elements in the buffer should be merged, before adding the new track group.
2. If the time difference between the oldest element in the buffer and the newly received track group is greater than a maximum delta, then all the elements in the buffer should be merged, before adding the new track group.
3. If, after adding the new track group to the buffer, the number of tracks to merge is equal to the number of detectors, then the next track group that will be received will necessarily belong to a new time frame, and thus all the elements in the buffer should be merged.

3.3. Skeleton Optimizer

The *Skeleton Optimizer* node allows to further identify outliers present, and to optimize joint positions/orientations to ensure limb length consistency of the tracked skeletons, after a prior calibration of the tracked persons. Therefore, the node can be divided into three parts: calibration (Section 3.3.1), outlier detection (Section 3.3.2), and, finally, optimization (Section 3.3.3).

3.3.1. Calibration

The calibration procedure aims to compute the nominal set of limb lengths for a track \mathcal{T} of a specific person:

$$\bar{\mathcal{L}} = \{\bar{l}_j \mid j = \{0, \dots, J - 1\}\} \quad (13)$$

where J is the total number of links defining a skeleton. The procedure is manually triggered and stores multiple frames of the tracked body poses in a buffer. Then, the average link lengths \bar{l}_j are computed, as well as the corresponding standard deviations. The standard deviation is an indication of the calibration quality. If one or more standard deviations are higher than a maximum threshold, then the calibration quality is considered too low, and a new calibration should be performed. Once proper calibration data are computed for a person, it is possible to save the results for future use (e.g., in a following session involving the same subject). Then, such data can be loaded without the necessity to recalibrate the same person twice.

3.3.2. Outliers detection

If calibration data are present, it is possible to further detect and fix outliers present in the tracked poses. A typical problem in markerless body tracking is the possibility to have non-negligible limb length variability between consecutive time frames. Knowing the correct body dimensions of the tracked person makes it possible to detect the markers that are being estimated in a wrong position.

For each tracked skeleton \mathcal{T} where calibration data is present, the length of each link is compared with the correct one obtained from the calibration. If such a difference is greater than a maximum threshold, then the distal marker forming the link is considered an outlier. At the same time, all links connected to the marker are also considered outliers.

3.3.3. Optimization

Once the highest sources of error are removed (i.e., the markers identified as outliers), the full skeleton undergoes an optimization procedure to minimize limb length variability. Such optimization aims at minimizing an energy cost that takes into account the limb length variability, the marker distances, and the orientation differences. The first term allows to have consistent body dimensions throughout all the acquisition, while the last two terms are required to have a resulting pose that is as close as possible to the original one.

For each pair of markers $\mathbf{m}_p, \mathbf{m}_d$ that form a link l , the energy cost of the length error is defined as:

$$E_l = (\|\mathbf{m}_p^* - \mathbf{m}_d^*\| - \bar{l})^2 \quad (14)$$

where \mathbf{m}_p^* and \mathbf{m}_d^* are the positions of the markers after the optimization, and \bar{l} is the length of the link connecting the two markers computed during the calibration.

The energy cost of the marker position errors is defined as:

$$E_p = \|\mathbf{m}_p^* - \mathbf{m}_p\|^2 + \|\mathbf{m}_d^* - \mathbf{m}_d\|^2 \quad (15)$$

Finally, the energy cost of the link orientation errors is defined following Eq. (10) as:

$$E_o = \phi(l^*, l)^2 \quad (16)$$

where l^* is the orientation of the link after optimization.

The total energy for each pair of markers forming a link is defined as:

$$E = w_l E_l + w_p E_p + w_o E_o \quad (17)$$

The three weights used are then defined as:

$$\begin{aligned} w_l &= (\|\mathbf{m}_p - \mathbf{m}_d\| - \bar{l} / \bar{L})^{0.2} \\ w_p &= \frac{1 - w_l}{2} \\ w_o &= 1 - w_l - w_p \end{aligned} \quad (18)$$

where w_l is the weight relative to the link length error, w_p the weight relative to the marker position error, w_o the weight relative to the link orientation error, \bar{L} is the maximum acceptable length error (links with errors greater than \bar{L} are considered outliers). In this way, the higher the length error, the stronger the optimization will be, while maintaining the original marker positions and link orientations when the link length error is small. Finally, the total energy cost relative to the full skeleton consists of the summation of all the energies related to each j th link:

$$E = \sum_{j=0}^{J-1} E_j \quad (19)$$

The global optimization problem that allows the minimization of the total energy cost is solved by exploiting the Levenberg–Marquardt algorithm implemented in the *Ceres Solver* library (Agarwal et al., 2022).

Table 1

Skeleton message definition.

Field	Description
<i>id</i>	ID of the person (set to -1 if no tracking is being performed)
<i>src_time</i>	Timestamp of the input data
<i>src_frame</i>	Reference frame of the input data
<i>max_markers</i>	Maximum number of markers that can be present in the skeleton
<i>max_links</i>	Maximum number of links that can be present in the skeleton
<i>confidence</i>	Confidence of the estimation (if available)
<i>bounding_box</i>	Smallest bounding box confining all the markers (if available)
<i>markers[]</i>	Vector of <i>Markers</i> (m)
<i>links[]</i>	Vector of <i>Links</i> (l)

3.4. Skeleton filter

Highly accurate biomechanical analyses of human movement typically require a filtering step in a post-processing phase to enhance the raw acquired data. For this reason, the *Hi-ROS* framework includes state-of-the-art real-time state-space and Butterworth filters to allow smoothing of the tracked bodies' trajectories.

The filter included in this work is based on an open-source implementation released by Pizzolato et al. (2015) under the Apache v.2 license. The filter has been adapted to be used on *SkeletonGroup* messages. Changes in the number of markers or links between consecutive frames are correctly handled, and a custom low-pass filter is implemented for the link orientations. The cutoff frequencies and the order of the Butterworth filter can be freely selected. Typical cutoff frequencies in biomechanics range between 6-10 Hz (Winter, 2009).

3.5. Skeleton messages

This section describes custom-defined messages designed to efficiently store information that describes the poses of multiple people. Such messages are used both as the output of each single camera detector, as well as to exchange data throughout the whole *Hi-ROS* pipeline. The messages are designed to ensure generality: no assumptions are made on the number of keypoints defining a pose, nor on the type of skeleton being used. Each message includes the 3D positions of the estimated keypoints (or markers) defining each person's body, and the orientations of its links (each segment connecting a pair of markers). The usage of a unique interface for the communication of data within the proposed framework, together with the choice of exploiting the ROS middleware, allows to easily support different cameras and body pose detection algorithms. In fact, to exploit all the tools and algorithms proposed in this work, it is sufficient to provide a bridge to convert the output of the specific detector into a *SkeletonGroup* message. Finally, ROS enables the usage of a large variety of cameras, where drivers are already available, either developed by the manufacturers or by the community, exploiting a unique interface independently of the specific hardware being used.

In detail, the *SkeletonGroup* message defined within *Hi-ROS* contains 2 fields: *header*: containing the publication time and the reference frame the data refer to; *skeletons[]*: vector of *Skeletons* (**S**).

The *Skeleton* message contains 9 fields, as described in Table 1. Each *Skeleton* is composed of a series of markers connected by links. Markers can represent either a person's estimated joint centers or actual markers applied on the body, while links allow to define the body hierarchy. The *src_time* and *src_frame* fields store information on the input source used for the pose estimation (e.g., the source time of the input image and the corresponding camera reference frame, respectively).

Markers and *Links* are defined as follows. The *Marker* message consists of 4 fields, as described in Table 2.

The *Link* message is similar to the *Marker* message, but also includes information on the parent and child marker IDs that form the link. Therefore, it consists of 6 fields, as described in Table 3.

Table 2

Marker message definition.

Field	Description
<i>id</i>	ID of the marker
<i>name</i>	Name of the marker (if available)
<i>confidence</i>	Confidence of the estimation (if available)
<i>center</i>	<i>KinematicState</i> of the marker's center

Table 3

Link message definition.

Field	Description
<i>id</i>	ID of the link
<i>name</i>	Name of the link (if available)
<i>parent_marker</i>	ID of the parent marker connected by the link
<i>child_marker</i>	ID of the child marker connected by the link
<i>confidence</i>	Confidence of the estimation (if available)
<i>center</i>	<i>KinematicState</i> of the link's center

Table 4*KinematicState* message definition.

Field	Description
<i>pose.position</i>	Position of the object
<i>pose.orientation</i>	Orientation of the object
<i>velocity</i>	Linear and angular velocities of the object
<i>acceleration</i>	Linear and angular accelerations of the object

A *KinematicState* allows to store information on the 3D pose, velocity, and acceleration of an object, as defined in Table 4. The *KinematicState* can be used to describe a marker's position (thereby leaving the *orientation* field empty), a link's orientation (thereby leaving the *position* field empty), or a system of reference (using both the *position* and *orientation* fields).

4. Experiments

The accuracy, tracking robustness, and real-time performance of the developed framework were assessed on a public dataset (Guidolin et al., 2022). The dataset includes RGB-D and markerless motion capture data of multiple sequences with up to 4 people interacting in a limited area, recorded using a calibrated network consisting of 5 Azure Kinect cameras. Markerless body poses are estimated by using the *Microsoft Azure Kinect Body Tracking SDK* detector. Each camera runs an instance of such detector, providing full-body pose data describing all people seen by the specific camera. The skeleton model used consists of 32 joints, where each joint includes data describing its position and the orientation of the child joint. Detailed information on the skeleton representation can be found in Guidolin et al. (2022). The full-body poses estimated by each detector are then used as input for the proposed framework, and the tracked skeletons are compared to the ones estimated by the *Xsens MVN Analyze* software.

The single-person sequences present in the dataset are characterized by specific actions (e.g., sitting, walking, jogging, waving hands, etc.), while multi-person sequences contain movements that led the participants to occlude each other multiple times during the experiments. The dataset also includes body poses estimated by exploiting full-body *Xsens MVN Awinda* inertial suits for up to 2 people simultaneously. No optoelectronic data are present due to the fact that the required markers attached to the body are highly reflective in the infrared domain, resulting in a strong distortion in the Kinects' depth and, consequently, in a poor estimation of body poses. However, the software used for the estimation of the body poses via inertial motion capture (that is, *Xsens MVN Analyze*), allows to obtain an accuracy comparable to that of state-of-the-art optoelectronic systems, as demonstrated in Schepers et al. (2018).

To assess the accuracy of the proposed framework, the markerless motion capture data retrieved from the dataset were converted into

SkeletonGroup messages and used as input for the *Skeleton Tracker*, as depicted in Fig. 2. We then compared the inertial body poses to the ones obtained by exploiting different configurations of the proposed modules:

1. configuration T: using the *Skeleton Tracker* only, which purely combines the raw detections from each camera;
2. configuration TMF: using *Tracker*, *Merger*, and *Filter*, thus avoiding the limb length optimization step;
3. configuration TMOF: exploiting the full pipeline, consisting of *Tracker*, *Merger*, *Optimizer*, and *Filter*.

The three configurations were chosen to allow an in-depth analysis of the *Tracker*, *Merger*, and *Optimizer*. Configuration T represents "raw" tracking (i.e., assigning a unique ID to all the people present in the scene, without any form of enhancement of the estimated body poses). Configurations TMF and TMOF, on the other hand, both include an initial tracking step, which is required to use the *Merger* and the *Optimizer*, and a final filtering step. In this way, we can analyze both the effects of including the optimizer (TMOF against TMF) and of including the merger (TMF against T).

To compare the accuracies obtained by the three configurations analyzed, for each trial we calculated the RMSE and the standard deviation (SD) between the estimated anatomical joint angles that define the pose of a person and the ones computed by inertial motion capture as:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1} (j_i - \hat{j}_i)^2}{N}}$$

$$SD = \sqrt{\frac{\sum_{i=0}^{N-1} (j_i - \mu)^2}{N}}$$
(20)

where j_i represents the i th joint angle estimated using the proposed system, \hat{j}_i the corresponding joint angle computed by *MVN Analyze*, N the total number of angles, and μ the mean joint angle value.

Let q_D^G and q_P^G be two quaternions that describe, respectively, the orientations of a distal segment and of a proximal segment, expressed with respect to the same reference frame G . The rotation of the joint connecting the two segments can be calculated as:

$$q_D^P = q_P^{G*} \otimes q_D^G \quad (21)$$

where \otimes indicates the quaternion multiplication, and $*$ the complex conjugate. Anatomical joint angles are then obtained by computing the corresponding Euler angles, following the zxy sequence. The International Society of Biomechanics (ISB) defines flexion/extension as the rotation about the z axis, abduction/adduction about the x axis, and internal/external rotation about the y axis (Wu et al., 2002, 2005).

Joint angles were preferred to keypoint positions due to the fact that a comparison between keypoint distances could lead to erroneous results. This depends on two main factors. First, the two systems (markerless motion capture and inertial motion capture) place their keypoints in different positions on the body. In fact, different motion capture systems and body pose estimation algorithms typically rely on unrelated models (if any). As a result, comparing different systems requires some sort of arbitrary mapping between the models. The second reason for focusing the analysis on the joint angles depends on the drift phenomena typical of inertial sensors, that can lead to increasing errors in the absolute position of the analyzed persons over time. However, the impact of drifting on the estimation of the joint angles is negligible, as demonstrated by Schepers et al. (2018).

The dataset used in this work is split into two sections: single-person sequences and multi-person sequences. Single-person sequences include the following actions: bending, crossing arms, jogging, jumping, N-pose, pointing, sitting, squatting, T-pose, throwing, walking, and waving. Multi-person sequences, on the other hand, include the following actions: people walking in a circle, people crossing, walking

Table 5

List of single-person and multi-person actions used for the experiments.

N people	Action	Description
1	walk	Walking at self-selected speed
1	squat	Squatting
1	bend	Bending down
1	sit	Sitting on a chair
1	jog	Jogging in place
1	jump	Jumping in place
1	cross_arms	Crossing arms
1	point	Pointing to different directions
1	wave	Waving hands
1	throw	Pretending to throw an object
2 – 4	free_static	Free movements while in the same place
2 – 4	free_dynamic	Free movements while changing positions
2 – 4	circle	Walk in a circle
2 – 4	cross	Switch positions while walking in a circle
2 – 4	in_out	Enter and exit from the cameras field of view
4	eight	Walk forming an eight

forming an *eight* shape, free dynamic movements, free static movements, entering/exiting from the scene, and static poses. More details can be found in Guidolin et al. (2022). Out of the 19 different actions, we removed N-pose, T-pose, and static sequences from our analysis since they are present for calibration purposes, and focused the analysis on active movements only. Table 5 describes all the actions taken into account for the accuracy assessment of the proposed framework.

For each recorded frame, we extracted the values of the following joint angles: left/right shoulder flexion/extension, left/right elbow flexion/extension, left/right hip flexion/extension, and left/right knee flexion/extension. The choice was driven by the fact that such angles are among the most informative, being the ones with the highest ranges of motion during typical movements. Only flexion/extension is taken into account because adduction/abduction and internal/external rotation cannot be estimated with high accuracy even by dedicated systems. Similarly, the estimates of such angles computed from the *Azure Kinect Body Tracking SDK* available in the dataset can be very noisy.

For each action (including 1, 2, 3, or 4 people), we calculated the mean RMSE and the SD of all the 8 joint angles taken into account (Tables 6 and 7). The accuracy analysis is divided between single-person sequences (Section 4.1) and multi-person sequences (Section 4.2). The tracking robustness of the proposed system is discussed in Section 4.3, while real-time performance is analyzed in Section 4.4.

4.1. Single-person sequences

The results obtained across all single-person sequences are reported in Table 6. The table outlines the accuracies achieved using configuration T, TMF, and TMOF, as well as the improvement gains obtained when using TMF and TMOF with respect to pure tracking when compared to the results provided by inertial motion capture.

The raw output of the *Azure Kinect Body Tracking SDK* detectors shows a mean RMSE between the values of the anatomical joint angles computed from the estimated body poses and the ones calculated by *MVN Analyze* of $(18.77 \pm 15.23)^\circ$ across all single-person sequences. The RMSE is reduced by 21.74 % when also including *Merger* and *Filter*, going from 18.77° to 14.69° . However, an even higher improvement in accuracy can be observed in the SD, with a decrease of 34.54 %, going from 15.23° to 9.97° . It is important to mention that some joint angles show a constant difference between inertial and markerless data. This can be noticed mainly on the upper part of the body and appears to be caused by a systematic shift in the positioning of the elbow keypoints estimated by the body pose detection algorithm used in the dataset. For this reason, better insight on the accuracy improvement obtained when exploiting the full *Hi-ROS* pipeline is obtained by comparing the SD of the errors, rather than the RMSE. The SD can, in fact, be seen as the

RMSE after compensating the constant shift. Finally, SD allows us to have an indication of how much flickering is present in the estimated body poses. In fact, the higher the SD and the more flickering is present.

When also including the *Optimizer*, the improvement with respect to pure tracking is still of 21.74 % on the RMSE, and 34.60 % on the SD, which is slightly better with respect to the TMF case. This behavior can be explained by two causes. The first is that the raw estimated body keypoints are already placed in positions such that the body dimensions do not change much between consecutive frames; thus, the effect of the *Optimizer* on the final poses is limited. The second reason is that the link lengths do not directly influence the estimation of the joint angles. As seen in Eq. (21), only the relative orientation of the distal and proximal links contributes to the final value of the angle. Thus, the accuracy of the estimated joint angles might remain substantially unchanged, while at the same time the estimated keypoint positions can be more consistent and respect the person's body dimensions throughout the whole experiment.

Analyzing specific actions, we can see that *walking* and *waving* are the most accurate, achieving accuracies of 6.45° and 6.08° respectively, and the ones with the highest error reductions (44.44 % and 53.65 % respectively). This means that, in these sequences, the impact of *Merger*, *Optimizer*, and *Filter* is maximized, allowing the correct detection and fix of flipped detections and outliers. *Pointing*, on the other hand, is the most challenging action. This can be explained by the fact that during the *pointing* sequences there is one arm that is constantly occluding important parts of the body, such as the chest and/or the shoulder, creating difficulties in the correct estimation of the pose and, at the same time, limiting the impact of outlier detection. It is worth noting that such poses are also critical with respect to inertial motion capture. In fact, the shoulder joint kinematics is typically the most difficult to estimate correctly (Jackson et al., 2012). Therefore, the highest errors in the *pointing* sequences may also be caused by errors in the poses estimated with the inertial suit. *Walking* and *waving*, on the other hand, are more accurately tracked by the camera network, being occlusions minimal.

As depicted in Fig. 5, which shows an example of joint angles estimated using the inertial suit and the complete *Hi-ROS* pipeline (TMOF) during a *wave* sequence (shoulders and elbows) and during a *jog* sequence (hips and knees), the estimates of the two systems are extremely similar, even in challenging actions as jogging.

Before discussing multi-person sequences, it is important to perform a more in-depth analysis of the *Skeleton Optimizer*. In fact, while its impact is overshadowed by the *Merger* when using state-of-the-art hardware and granting optimal views of the persons on the scene, the *Optimizer* is a powerful tool, designed to be used in the most challenging scenarios. When less accurate hardware is adopted, resulting in a noisy estimated depth, or if the tracked persons are far from the sensors, the estimated keypoint positions can lead to significantly varying limb lengths. Although the data recorded in the dataset do not suffer from these problems, the same cannot be ensured in different scenarios. When this is the case, the *Optimizer* allows to minimize the impact of erroneous estimates and to guarantee meaningful body poses even in the most challenging scenarios. This is especially important in applications where correct absolute positions of the estimated body keypoints are mandatory, such as in the human-robot interaction field. In this case, a wrong estimation of a hand keypoint, for example, might indicate that the operator is far from the moving robot, while in reality, their hand is closer to a dangerous area. The optimization would allow to identify the wrongly estimated keypoints and fix their positions to respect the anatomical constraints, thus correcting the body pose and avoiding risks for the operator.

4.2. Multi-person sequences

The results of multi-person sequences are reported in Table 7 and show similar errors with respect to single-person ones. It is worth

Table 6

Framework accuracy in single-person sequences when using the *Tracker* only (T), *Tracker, Merger*, and *Filter* (TMF), and *Tracker, Merger, Optimizer*, and *Filter* (TMOF). Data are presented as RMSE (SD).

N people	Action	T [°]	TMF [°]	TMOF [°]	Δ T-TMF [%]	Δ T-TMOF [%]
1	bend	17.98 (13.48)	14.39 (8.77)	14.36 (8.76)	-19.99% (-34.92%)	-20.13% (-34.99%)
1	cross_arms	17.70 (13.53)	13.82 (8.44)	13.80 (8.46)	-21.91% (-37.64%)	-22.05% (-37.51%)
1	jog	19.15 (17.45)	14.39 (12.20)	14.40 (12.19)	-24.89% (-30.08%)	-24.81% (-30.12%)
1	jump	21.60 (18.76)	16.40 (13.00)	16.44 (13.05)	-24.10% (-30.72%)	-23.90% (-30.43%)
1	point	24.39 (20.14)	21.13 (15.94)	21.21 (16.02)	-13.36% (-20.86%)	-13.02% (-20.46%)
1	sit	16.52 (13.37)	12.40 (8.38)	12.40 (8.28)	-24.98% (-37.26%)	-24.98% (-38.08%)
1	squat	17.31 (14.76)	14.42 (11.31)	14.28 (11.10)	-16.73% (-23.41%)	-17.53% (-24.82%)
1	throw	19.58 (16.11)	14.25 (9.14)	14.26 (9.16)	-27.24% (-43.26%)	-27.16% (-43.15%)
1	walk	16.15 (11.61)	12.98 (6.45)	12.98 (6.45)	-19.61% (-44.44%)	-19.61% (-44.44%)
1	wave	17.36 (13.12)	12.70 (6.08)	12.75 (6.16)	-26.83% (-53.65%)	-26.55% (-53.09%)
1	mean	18.77 (15.23)	14.69 (9.97)	14.69 (9.96)	-21.74% (-34.54%)	-21.74% (-34.60%)

Table 7

Framework accuracy in multi-person sequences when using the *Tracker* only (T), *Tracker, Merger*, and *Filter* (TMF), and *Tracker, Merger, Optimizer*, and *Filter* (TMOF). Data are presented as RMSE (SD).

N people	Action	T [°]	TMF [°]	TMOF [°]	Δ T-TMF [%]	Δ T-TMOF [%]
2	circle	16.88 (12.93)	14.00 (8.50)	14.10 (8.62)	-17.06% (-34.27%)	-16.47% (-33.34%)
2	cross	17.00 (13.41)	13.90 (8.96)	13.99 (9.04)	-18.24% (-33.23%)	-17.71% (-32.60%)
2	free_dynamic	16.66 (13.59)	13.05 (8.42)	13.07 (8.49)	-21.68% (-38.03%)	-21.54% (-37.51%)
2	free_static	16.58 (13.91)	12.31 (8.05)	12.35 (8.10)	-25.76% (-42.15%)	-25.51% (-41.77%)
2	in_out	16.56 (13.04)	14.32 (11.66)	14.39 (11.72)	-13.50% (-10.54%)	-13.08% (-10.09%)
2	mean	16.74 (13.38)	13.52 (9.12)	13.58 (9.19)	-19.24% (-31.84%)	-18.88% (-31.32%)
3	circle	16.81 (13.21)	14.22 (9.17)	14.28 (9.28)	-15.40% (-30.62%)	-15.02% (-29.71%)
3	cross	17.67 (14.04)	15.68 (11.28)	15.82 (11.44)	-11.24% (-19.65%)	-10.45% (-18.50%)
3	free_dynamic	18.31 (15.28)	13.79 (9.54)	13.85 (9.64)	-24.68% (-37.55%)	-24.36% (-36.89%)
3	free_static	17.59 (13.21)	14.87 (8.99)	14.78 (8.94)	-15.44% (-31.91%)	-15.98% (-32.31%)
3	in_out	16.59 (13.97)	14.71 (12.27)	14.81 (12.35)	-11.33% (-12.21%)	-10.76% (-11.64%)
3	mean	17.39 (13.94)	14.65 (10.25)	14.71 (10.33)	-15.76% (-26.47%)	-15.41% (-25.90%)
4	circle	18.48 (14.66)	16.51 (12.19)	16.69 (12.37)	-10.67% (-16.83%)	-9.71% (-15.62%)
4	cross	17.91 (14.22)	15.76 (11.60)	15.84 (11.73)	-12.03% (-18.45%)	-11.59% (-17.54%)
4	eight	16.49 (13.03)	14.82 (10.50)	14.95 (10.64)	-10.13% (-19.38%)	-9.32% (-18.36%)
4	free_dynamic	18.96 (16.29)	15.41 (12.04)	15.44 (12.09)	-18.75% (-26.11%)	-18.57% (-25.80%)
4	free_static	17.61 (14.30)	14.53 (10.43)	14.43 (10.35)	-17.51% (-27.12%)	-18.05% (-27.65%)
4	in_out	17.35 (14.08)	15.57 (12.08)	15.55 (12.06)	-10.27% (-14.16%)	-10.36% (-14.36%)
4	mean	17.80 (14.43)	15.43 (11.47)	15.48 (11.54)	-13.31% (-20.51%)	-13.03% (-20.03%)

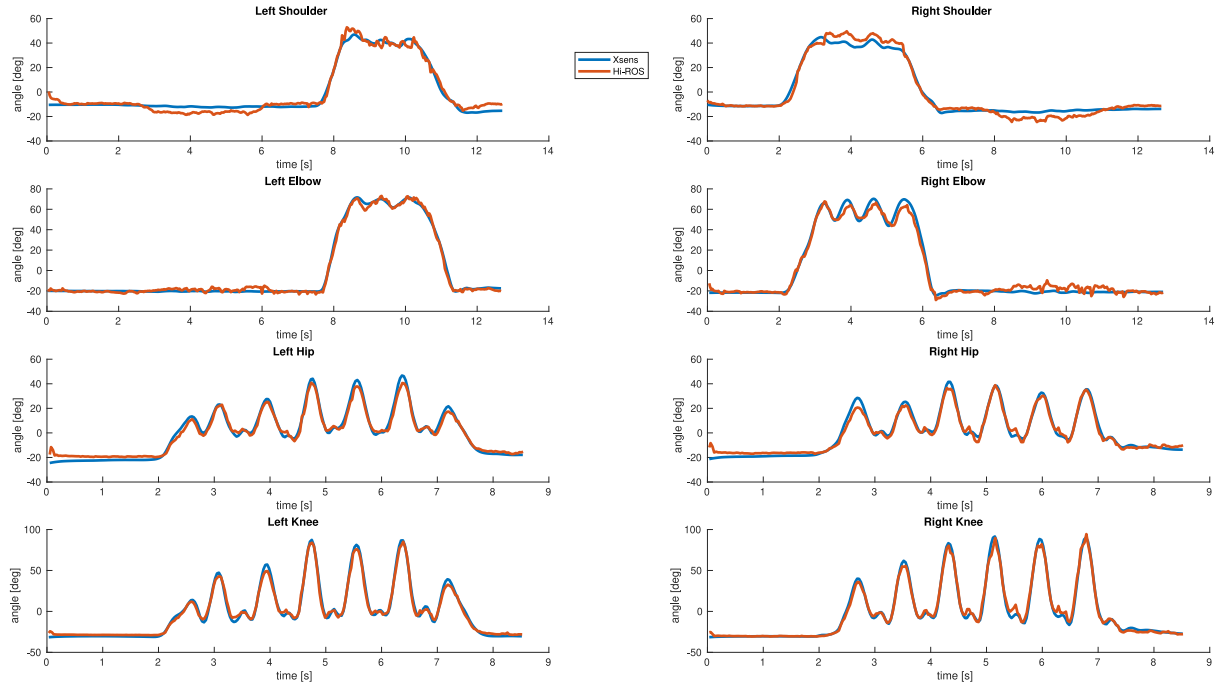


Fig. 5. Joint angles calculated by *Xsens* (in blue) and *Hi-ROS* (in orange). Shoulder and elbow angles refer to a *wave* sequence, while hip and knee angles refer to a *jog* sequence. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

noticing that the average error for the sequences with 2 persons is even lower than the errors obtained in single-person sequences. The reason behind this can be explained by the fact that the increased difficulty arising from the higher number of people present in the scene is compensated for by the different typologies of actions being performed.

As depicted in Table 7, all three configurations show the same behavior, depending on the number of people in the scene, where the errors slightly increase as the number of persons in the scene increases. This was expected and is dependent on the fact that the more people are tracked in the same confined space, the more occlusions will be present. When a body is partially occluded, erroneous estimations of part of the body keypoints might occur. Additionally, it is easier for a detected skeleton to be completely occluded by others and, as a result, to be considered as a new person when it reappears. The *Azure Kinect Body tracking SDK* tends to spawn new skeletons in an incorrect pose, resulting in temporary larger errors during the first few frames. However, the *Tracker* is still able to assign the correct IDs even in these cases.

In multi-person sequences, the errors obtained from pure tracking start from $(16.74 \pm 13.38)^\circ$ when tracking 2 persons, increase to $(17.39 \pm 13.94)^\circ$ when 3 people are present, and reach $(17.80 \pm 14.43)^\circ$ with 4 people in the scene. The same trend is obtained when using TMF and TMOF. The accuracy improvement with respect to raw tracking decreases as the number of people increases. With 2 people in the scene, the RMSE is reduced by 19.24% when using TMF and by 18.88% when using TMOF, while the SD is reduced by 31.84% (TMF) and 31.32% (TMOF). With 3 people in the scene, the RMSE is reduced by 15.76% when using TMF and by 15.41% when using TMOF, while the SD is reduced by 26.47% (TMF) and 25.90% (TMOF). Finally, with 4 people in the scene, the RMSE is reduced by 13.31% when using TMF and by 13.03% when using TMOF, while the SD is reduced by 20.51% (TMF) and 20.03% (TMOF). Here, the results of the *Optimizer* are slightly less accurate than those of the *Merger*. As in single-person sequences, the difference is negligible, since the different positions of the keypoints after the optimization do not directly impact the estimation of the joint angles, which depend on the orientations of the body links.

The smaller improvement with respect to single-person sequences may be due to the fact that, overall, a lower number of cameras are seeing the persons at each time frame. Although this difference does not affect the pure tracking accuracy (configuration T), which consists only of the association of the raw detections obtained by each camera, this affects *Merger* and *Optimizer*. In fact, outlier filtering can be less effective when a lower number of cameras can see the persons in the scene. However, the increase in accuracy is still important, with a 20% error reduction when using TMOF with respect to *T* even when tracking 4 people in close proximity.

When analyzing single actions, in TMF and TMOF we can notice a consistent behavior among the different sequences, where *free_static* is typically the most accurate action (with errors ranging from 8.05° to 10.43°) and *in_out* is the least accurate one (with errors ranging from 11.66° to 12.08°). This was expected since during *free_static* sequences people are standing in the same place during the whole experiment, whereas in *in_out* sequences people are constantly exiting and reentering the acquisition area. However, in the T configuration, the behavior is almost inverted. In this case, the *in_out* sequences are among the most accurate, while the highest errors occur in less demanding sequences, such as *free_static* and *free_dynamic*. A possible explanation for this is the presence of flipped detections. In fact, as introduced in Section 3, it is possible for one or more cameras to detect a body pose flipped by 180° . If such a person is moving, there is a high chance that the pose will be fixed after a few frames. However, if the person is standing still, the orientation of the detection might remain flipped throughout the whole experiment. In this sense, the fact that people are constantly moving in *in_out* sequences helps to reduce the impact of such flipped detections. On the contrary, in the *free_static* and *free_dynamic* sequences, this cannot happen. The same behavior is not present in the TMF and TMOF configurations. In fact, *Merger* and *Optimizer* are designed to correctly detect and remove both flipped detections and outlier markers/links.

4.3. Tracking robustness

When tracking multiple people, it is important to be able to assign a consistent ID to each person throughout the whole experiment. Extreme proximity of 2 or more persons, as well as awkward poses, can cause tracking algorithms to switch the assigned IDs. In order to assess the robustness of the proposed framework on assigning consistent IDs to all the tracked people, the output of multi-person sequences was manually checked. The tracked skeletons (and their IDs) were projected onto the point cloud obtained by merging the 5 Kinects (Fig. 1). The correctness of each ID was then manually controlled for all the analyzed sequences.

The IDs were correctly assigned to each participant, independently of the number of people present in the scene and of the proximity of the tracked people. There are some cases, especially in *in_out* sequences, where some persons exit from the field of view of all cameras simultaneously for small windows of time. Then, when a person reenters the scene, the system gives them a new ID. However, even in these cases, the IDs are correctly assigned and kept by each track during the time windows in which they are visible to at least one camera. It is worth noticing that *cross* and *eight* sequences are extremely challenging in this regard, since people are constantly moving across each other, while also being partially occluded to one or more cameras. Finally, it is important to remark that person re-identification after one or multiple subjects are not seen by any camera for prolonged periods of time is out of the scope of this work, where the *Tracker* is the first step to achieve, together with *Merger*, *Optimizer*, and *Filter*, highly accurate markerless full-body pose estimation and tracking in real-time. Thus, we can conclude that the *Hi-ROS* framework is robust and allows to correctly track all the persons in the scene.

4.4. Real-time performance

The average computation time required to process a single frame was measured for all the nodes of the proposed framework, i.e., tracking time, merging time, optimizing time, and filtering time. Table 8 reports the computation times obtained when varying the number of people being tracked. The tests were run on an Intel Core i7-1165G7 CPU @ 2.80 GHz laptop with 16 GB of RAM.

As expected, the results show that the optimization is the most demanding task, ranging from 4.16 ms when a single person is being tracked to 8.33 ms when tracking 4 people. Tracking, merging, and filtering times are negligible, requiring respectively 0.19 ms, 0.55 ms, and 0.11 ms to track one person, and 0.86 ms, 0.88 ms, and 0.21 ms to track 4 people.

Table 8 also reports estimates of the computation times required to track more than 4 people. Such data are calculated by assuming a linear trend between the number of people being tracked and the computation time required by each node. It is important to note that the total time to process each frame is not equal to the sum of the computation times of all nodes. In fact, each node runs in parallel. Therefore, taking the 4 people scenario as an example, the average time required to process each frame would be equal to 8.33 ms (the time required by the most computationally intensive node), which corresponds to a theoretical maximum frame rate of ~ 120 Hz. On the other hand, the average delay between input detections and output tracks corresponds to the sum of the times of all the nodes. In this case, it would be equal to 10.28 ms.

Finally, since the usage of *Merger*, *Optimizer*, and *Filter* is optional, faster performance can be obtained, as an example, by removing the *Optimizer*, with the downside, however, of not guaranteeing consistent limb lengths during the experiments. We can see that when tracking 10 people, the computation time required by the *Optimizer* is 16.81 ms, corresponding to a maximum frame rate of ~ 59 Hz. However, without the *Optimizer*, the computation time required for each frame would decrease to 2.22 ms, allowing to reach a maximum frequency of ~ 450 Hz.

Table 8

Computation times required to process a single frame. Times to track 1 to 4 people are calculated on the dataset trials, while times to track 5 and 10 people are estimated from the previous ones assuming a linear trend between the number of tracked people and the computation time required. Data are presented as mean (SD).

N people	Tracker [ms]	Merger [ms]	Optimizer [ms]	Filter [ms]
1	0.19 (0.05)	0.55 (0.08)	4.16 (0.95)	0.11 (0.03)
2	0.37 (0.14)	0.71 (0.28)	6.44 (2.13)	0.17 (0.06)
3	0.63 (0.26)	0.85 (0.43)	7.52 (3.10)	0.19 (0.08)
4	0.86 (0.39)	0.88 (0.55)	8.33 (3.76)	0.21 (0.10)
5	1.08	1.03	10.01	0.25
10	2.22	1.59	16.81	0.40

5. Conclusions

In this paper, we presented *Hi-ROS*, an open-source framework for real-time accurate assessment of human motion. This work aims at putting the basis to enable accurate human motion analysis in everyday and industrial environments, without the necessity of a dedicated laboratory and expensive equipment. This is made possible by the open-source nature of the project, the ability to use the framework independently of the quantity and typology of sensors being adopted, and its efficiency that allows to obtain real-time performance, even when a large number of people are present in the scene. The proposed system offers a series of algorithms to perform real-time multi-camera multi-person tracking, consisting of 4 main modules:

1. a *Skeleton Tracker* node to perform pure frame-by-frame tracking of the detected body poses;
2. a *Skeleton Merger* node to fuse detections from multiple cameras and detect outliers;
3. a *Skeleton Optimizer* node to perform a global optimization that ensures consistency of each person's body dimensions;
4. a *Skeleton Filter* node to perform real-time filtering of the estimated marker positions and link orientations.

All the developed nodes communicate with each other by means of efficient custom-defined *Skeleton Messages*.

The accuracy and real-time performance of the proposed system were assessed on a public dataset. The dataset includes a variety of sequences with up to 4 people interacting, recorded by exploiting a calibrated camera network consisting of 5 Azure Kinect cameras. Body poses are captured for up to 2 people simultaneously using an accurate inertial motion capture suit (*Xsens MVN Awinda*). The results show that, by exploiting the *Hi-ROS* framework, it is possible to reduce body pose estimation errors by up to 27.24 % when compared to a pure tracking-by-detection approach. When increasing the number of people in the scene, the framework is still capable of reducing the estimation errors by up to 25.76 % with 2 persons, by up to 24.68 % with 3 persons, and by up to 18.75 % with 4 persons. Robustness analyses show that correct IDs are assigned to each tracked person in all the dataset sequences. Even during periods of close proximity of 2 or more people, no ID switch is reported. Real-time performance was also analyzed by reporting the computational time required by each node of the framework to process a frame. The complete *Hi-ROS* pipeline is able to produce real-time results, reaching 240 Hz when tracking a single person, or 120 Hz when tracking 4 people. Depending on the configuration being used, even faster performance can be achieved.

The analyses performed suffer from two limitations. The first one is the possibility of having inertial body kinematics data only for up to 2 persons simultaneously. This is a limitation of the dataset, where the inertial suits were not available for more people. For this reason, the results reported in Table 7 for multi-person sequences are always obtained by comparing the joint angles of 2 subjects only. In sequences with more than 2 people interacting, it is not possible to know the inertial poses of the other persons. Another limitation depends on the nature of the body poses estimated by inertial motion capture. By using inertial suits, in fact, it is not possible to directly compare the

Euclidean distances between the keypoints estimated using the two motion capture systems. Consequently, all the analyses were conducted on the joint angles describing the poses, rather than on the estimated body keypoints.

Future work aims to exploit raw inertial data (i.e., raw IMU orientations), together with markerless motion capture, as input to drive a common musculoskeletal model of the people being analyzed. This should allow to further improve the system's accuracy, while also minimizing the drifting phenomena that are typical of pure inertial motion capture systems. This approach will also allow to directly compare both the estimated joint angles and body keypoints, since they are all defined on the same shared model.

CRedit authorship contribution statement

Mattia Guidolin: Conceptualization, Methodology, Software, Validation, Data curation, Writing – original draft, Visualization. **Luca Tagliapietra:** Conceptualization, Methodology, Software. **Emanuele Menegatti:** Conceptualization, Writing – review & editing, Supervision. **Monica Reggiani:** Conceptualization, Methodology, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Datasets related to this article can be found at <https://doi.org/10.17605/OSF.IO/YJ9Q4>, an open-source online data repository hosted at OSF (Guidolin et al., 2022).

Acknowledgments

The authors would like to express their gratitude to the funding agencies that supported their research. M. Guidolin and M. Reggiani carried out this study within the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from European Union - Next Generation EU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10-2022, PE00000004), CUP: C93C22005280001. E. Menegatti acknowledges the support of European Union - Next Generation EU, in the context of The National Recovery and Resilience Plan, Investment Partenariato Esteso PE8 “Conseguenze e sfide dell'invecchiamento”, Project Age-IT, CUP: C93C22005240007. Finally, we thank the editorial team and reviewers for their time and effort in reviewing our work. Their insightful comments and suggestions have greatly improved the quality of this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.cviu.2023.103694>.

References

- Agarwal, S., Mierle, K., Team, T.C.S., 2022. Ceres solver. URL: <https://github.com/ceres-solver/ceres-solver>.
- Battini, D., Berti, N., Finco, S., Guidolin, M., Reggiani, M., Tagliapietra, L., 2022. WEM-Platform: A real-time platform for full-body ergonomic assessment and feedback in manufacturing and logistics systems. *Comput. Ind. Eng.* 164, 107881.
- Bochinski, E., Eiselein, V., Sikora, T., 2017. High-speed tracking-by-detection without using image information. In: 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance. AVSS, IEEE, pp. 1–6.
- Bragança, S., Costa, E., Castellucci, I., Arezes, P.M., 2019. A brief overview of the use of collaborative robots in industry 4.0: Human role and safety. In: Arezes, P.M., Baptista, J.A.S., Barroso, M.P., Carneiro, P., Cordeiro, P., Costa, N., Melo, R.B., Miguel, A.S., Perestrelo, G. (Eds.), *Occupational and Environmental Safety and Health*. Springer International Publishing, Cham, ISBN: 978-3-030-14730-3, pp. 641–650.
- Bulmann, S., Behnke, S., 2021. Real-time multi-view 3D human pose estimation using semantic feedback to smart edge sensors. *arXiv preprint arXiv:2106.14729*.
- Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., Sheikh, Y., 2019. OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. *IEEE Trans. Pattern Anal. Mach. Intell.* 43 (1), 172–186.
- Chen, L., Ai, H., Chen, R., Zhuang, Z., Liu, S., 2020. Cross-view tracking for multi-human 3D pose estimation at over 100 FPS. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR*, pp. 3279–3288.
- Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., Sun, J., 2018. Cascaded pyramid network for multi-person pose estimation. In: *IEEE Conference on Computer Vision and Pattern Recognition. CVPR*, pp. 7103–7112.
- Chu, H., Lee, J.-H., Lee, Y.-C., Hsu, C.-H., Li, J.-D., Chen, C.-S., 2021. Part-aware measurement for robust multi-view multi-human 3D pose estimation and tracking. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR*, pp. 1472–1481.
- Fender, A., Müller, J., 2018. Velt: A framework for multi RGB-D camera systems. In: 2018 ACM International Conference on Interactive Surfaces and Spaces. pp. 73–83.
- Guidolin, M., Menegatti, E., Reggiani, M., 2022. UNIPD-BPE: Synchronized RGB-D and inertial data for multimodal body pose estimation and tracking. *Data* 7 (6), 79.
- Jackson, M., Michaud, B., Tétreault, P., Begon, M., 2012. Improvements in measuring shoulder joint kinematics. *J. Biomech.* 45 (12), 2180–2183.
- Jiao, Y., Yao, H., Xu, C., 2020. PEN: Pose-embedding network for pedestrian detection. *IEEE Trans. Circuits Syst. Video Technol.* 31 (3), 1150–1162.
- Kadkhodamohammadi, A., Gangi, A., de Mathelin, M., Padoy, N., 2017. A multi-view RGB-D approach for human pose estimation in operating rooms. In: 2017 IEEE Winter Conference on Applications of Computer Vision. WACV, IEEE, pp. 363–372.
- Lim, S., D'Souza, C., 2020. A narrative review on contemporary and emerging uses of inertial sensing in occupational ergonomics. *Int. J. Ind. Ergon.* 76, 102937.
- Liu, G., Tian, G., Li, J., Zhu, X., Wang, Z., 2018. Human action recognition using a distributed RGB-Depth camera network. *Sensors* 18 (18), 7570–7576.
- Malaguti, A., Carraro, M., Guidolin, M., Tagliapietra, L., Menegatti, E., Ghidoni, S., 2019. Real-time tracking-by-detection of human motion in RGB-D camera networks. In: 2019 IEEE International Conference on Systems, Man and Cybernetics. SMC, IEEE, pp. 3198–3204.
- Martínez-González, A., Villamizar, M., Canévet, O., Odobez, J.-M., 2018. Real-time convolutional networks for depth-based human pose estimation. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 41–47.
- Mehta, D., Sotnychenko, O., Mueller, F., Xu, W., Elgharib, M., Fua, P., Seidel, H.-P., Rhodin, H., Pons-Moll, G., Theobalt, C., 2019. XNect: Real-time multi-person 3D human pose estimation with a single RGB camera. *arXiv preprint arXiv:1907.00837*.
- Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.-P., Xu, W., Casas, D., Theobalt, C., 2017. VNect: Real-time 3D human pose estimation with a single RGB camera. *ACM Trans. Graph.* 36 (4), 1–14.
- Microsoft, 2021. Azure kinect body tracking SDK documentation. <https://microsoft.github.io/Azure-Kinect-Body-Tracking/release/1.x.x/index.html>. (Accessed 25 March 2023).
- Moon, S., Park, Y., Ko, D.W., Suh, I.H., 2016. Multiple kinect sensor fusion for human skeleton tracking using Kalman filtering. *Int. J. Adv. Robot. Syst.* 13 (2), 65.
- Munaro, M., Basso, F., Menegatti, E., 2016. OpenPTrack: Open source multi-camera calibration and people tracking for RGB-D camera networks. *Robot. Auton. Syst.* 75, 525–538.
- Munkres, J., 1957. Algorithms for the assignment and transportation problems. *J. Soc. Ind. Appl. Math.* 5 (1), 32–38.
- Newell, A., Yang, K., Deng, J., 2016. Stacked hourglass networks for human pose estimation. In: *European Conference on Computer Vision*. Springer, pp. 483–499.
- Pizzolato, C., Lloyd, D.G., Sartori, M., Ceseracciu, E., Besier, T.F., Fregly, B.J., Reggiani, M., 2015. CEINMS: A toolbox to investigate the influence of different neural control solutions on the prediction of muscle excitation and joint moments during dynamic motor tasks. *J. Biomech.* 48 (14), 3929–3936.
- Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., 2009. ROS: An open-source robot operating system. In: *ICRA Workshop on Open Source Software*, Vol. 3, no. 3.2. Kobe, Japan, p. 5.
- Reddy, N.D., Guigues, L., Pishchulin, L., Eledath, J., Narasimhan, S.G., 2021. TesseTrack: End-to-end learnable multi-person articulated 3D pose tracking. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR*, pp. 15190–15200.
- Rim, B., Sung, N.-J., Ma, J., Choi, Y.-J., Hong, M., 2020. Real-time human pose estimation using RGB-D images and deep learning. *J. Internet Comput. Serv.* 21 (3), 113–121.
- Ryselis, K., Petkus, T., Blažauskas, T., Maskeliūnas, R., Damaševičius, R., 2020. Multiple kinect based system to monitor and analyze key performance indicators of physical training. *Human-Centric Comput. Inf. Sci.* 10 (1), 1–22.
- Schepers, M., Giuberti, M., Bellucci, G., et al., 2018. Xsens MVN: Consistent tracking of human motion using inertial sensing. *Xsens Technol* 1–8.
- Toshev, A., Szegedy, C., 2014. DeepPose: Human pose estimation via deep neural networks. In: *IEEE Conference on Computer Vision and Pattern Recognition. CVPR*, pp. 1653–1660.
- Vysocky, A., Novak, P., 2016. Human-robot collaboration in industry. *MM Sci. J.* 9 (2), 903–906.
- Winter, D.A., 2009. *Biomechanics and Motor Control of Human Movement*. John Wiley & Sons.
- Wu, G., Van der Helm, F.C., Veeger, H.D., Makhosous, M., Van Roy, P., Anglin, C., Nagels, J., Karduna, A.R., McQuade, K., Wang, X., et al., 2005. ISB recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion—Part II: Shoulder, elbow, wrist and hand. *J. Biomech.* 38 (5), 981–992.
- Wu, G., Siegler, S., Allard, P., Kirtley, C., Leardini, A., Rosenbaum, D., Whittle, M., D D'Lima, D., Cristofolini, L., Witte, H., et al., 2002. ISB recommendation on definitions of joint coordinate system of various joints for the reporting of human joint motion—Part I: ankle, hip, and spine. *J. Biomech.* 35 (4), 543–548.
- Zhao, Y., Yuan, Z., Chen, B., 2019. Accurate pedestrian detection by human pose regression. *IEEE Trans. Image Process.* 29, 1591–1605.
- Zhou, L., Lannan, N., Fan, G., Hausselle, J., 2021. Human motion enhancement via joint optimization of kinematic and anthropometric constraints. *EAI Endorsed Trans. Bioeng. Bioinform.* 1, e1.