Skeleton estimation and tracking by means of depth data fusion from depth camera networks

Marco Carraro[a], Matteo Munaro[a,b], Emanuele Menegatti[a]

[a] *University of Padova, Via G. Gradenigo 6/A, Padova, Italy*
[b] *Fyusion Inc.*

Abstract

In this work, we describe an approach for estimation and tracking of the skeleton of the human body from camera networks exploiting only depth data. The algorithm takes advantage of multiple views by building and merging together the 3D point clouds. The final skeleton is computed from a virtual depth image generated from this point cloud by means of back-projection to a reference camera image plane. Before the back-projection, the person point cloud is frontalized with respect to the reference camera, so that the virtual depth image represents the person from a frontal viewpoint and the accuracy of the skeleton estimation algorithm is maximized. Our experiments show how the proposed approach boosts the performance with respect to other state-of-the-art approaches. Moreover, the proposed algorithm requires low computational burden, thus running in real-time.

Keywords: multi-view skeletal tracking, markerless human body pose estimation, depth data, frontalization, camera networks

1. Introduction

Human Body Pose Estimation (H-BPE) in single and multi-camera networks is a hot research topic since a long-time, due to its importance to fields as Action Recognition [1], Entertainment [2], Rehabilitation [3], Human-Computer Interaction [4] and Robotics [5]. The most common solution to this problem, in the so-called "motion-capture systems", is to have the user to wear a special

suit or a set of markers that can be easily detected and that approximate the performer motion.

While this solution provides very good results in terms of accuracy (usually with an accuracy of a millimeter), it is often unfeasible to use in real-world scenarios, in particular, when real-time processing of the motion is needed. Moreover, the set of markers to be weared encumbers the user movements, impacting his/her performance. For those reasons, marker-less motion capture systems would be preferred and have been also studied. In particular, many efforts have been put in RGB-based human BPE, since RGB cameras are very common and cheap. To this end, recently, new computational capabilities and deep neural networks allowed to solve this problem with detection accuracy approaching that of marker-based systems [6, 7]

At the same time, RGB-only solutions heavily rely on the lighting of the scene. In many real-world applications as automated assembly in industrial application lines or in theaters, it is not always possible to guarantee enough light to obtain useful RGB information. In the literature, this problem is commonly overcome by using depth images as the input data. Nevertheless, if this data are obtained by passive sensors as stereo-cameras, they suffer from the same aforementioned problems of scarcity of light. On the other hand, active cameras do not rely on the visible light. They project an infrared pattern useful to triangulate and generate depth information, as the Microsoft Kinect v1, or use an array of emitters and measure the phase shift of the returning signal, as the Microsoft Kinect v2 [8].

In this work, we are proposing a marker-less solution to the human BPE problem. In particular, the solution we will describe uses depth-only information to be reliable in most light conditions and it best exploits a single-view state-of-the-art depth-based skeletal tracker by using a frontal view warping of each subject. The novelty of this work is two-fold:

- we enhance the performance of a state-of-the-art depth-based skeletal tracker.

- we propose a novel pose-invariant algorithm for solving the human BPE problem in multi-camera scenarios.

The remainder of the paper is organized as follows: in Section 2 we will review the state-of-the-art of both single and multi-view human body pose estimation. Section 3 describes the algorithm details, while in Section 4 we will validate our approach with experiments done with two different persons. Finally, in Section 5 we will draw our conclusions and describe the future work.

2. Related Work

Research about BPE is active in both single and multi-camera scenarios. In this section we will discuss about both.

2.1. Single-view skeletal tracking

Since it is way easier and cheaper to deal with a single sensor, most of the BPE research is focused on this category. Recent years have seen a general improvement from the quality point of view thanks to advances in the machine learning field, in particular with deep learning solutions. As an example, Convolutional Neural Networks (CNNs) showed great success when trained on very large datasets with sufficient texture information as with RGB images. The impressive quality result is usually paid in terms of the final framerate achievable. Nevertheless, this limitation is going to be leveraged by using new efficient network architectures.

In this context, the work of Cao et al [6] was one of the first to reach real-time performance by using an architecture which jointly computes the body part locations of all the persons in an image together with their Part Affinity Fields (PAFs). This approach is similar to the work of Insafutdinov et al [9], while it is more efficient, since it associates the different body parts using a fast greedy algorithm which exploits the computed PAFs.

On the other hand, the availability of different types of data as the depth information, gives the possibility of being more robust to light conditions, tracking the persons' skeleton more reliably.

In [2], the Microsoft Kinect skeletal tracker is described. The authors trained a random forest to classify the pixels of a depth image as belonging to different human body parts or not. The final algorithm works in real-time and has been adopted by Microsoft in its entertainment applications. The algorithm is closed-source, but is available with Windows-only computers through the Microsoft Kinect SDK.

A similar work was released as open-source in the *Point Cloud Library*[1][10] by Buys et. al [11]. In this work, we adopt this skeletal tracker as part of our pipeline, by enhancing its performance using a prior people detection step.

Another famous skeletal tracker which uses depth information to compute the skeleton is the NiTE skeletal tracker, which is closed-source and licensed by the Israeli PrimeSense, acquired by Apple in 2013. Unfortunately, given its nature, we have no information on how this skeletal tracker works, even if several works proved that it provides worse results compared to the ones obtained with other available skeletal trackers.

In this work, we improve the performances of [11] by combining it with a people detection module. We also enrich the overall results by introducing an alpha-beta tracking module.

2.2. Multi-view skeletal tracking

Intelligent BPE systems must deal with occlusions and should cover larger areas. For this reason, the direction is towards camera networks which can exploit multiple views and enlarge the single camera field-of-view. In this way, it is possible to observe people movements for longer periods enabling BPE to be useful for different applications as behavior analysis [12], long-term people re-identification [13, 14] , as well as action recognition [1].

In [15], each sensor computes the single-view skeleton from the RGB images which are then fused with the skeleton estimated from the 3D model obtained using the visual hull technique. Such model is exploited to refine the skeletons

[1] `https://github.com/pointcloudlibrary/pcl`

obtained.

In [16] and [17], the authors used the single view skeletons obtained with as a feed to compute the final 3D skeleton exploiting the combination of proposal from the PBD algorithm. The skeletons are rendered in 3D by means of projection using each couple of cameras as a stereo couple.

The recent work described in [18][2] is one of the first open-source solutions to give a general approach for the real-time BPE using RGB-D camera networks of different sizes and for multiple people that does not require synchronization between the cameras. Nevertheless, the skeletal tracker used rely on RGB data, therefore, illumination may impact the overall skeleton estimation quality. In order to be robust with different illumination variation the literature rely on depth or point cloud data to compute the skeleton information.

Gao et. al[19] solved the BPE problem by registering a point cloud obtained by two Microsoft Kinects to a 3D model. While the results are very accurate, the work is not feasible to work in real-time given the high computational burden of the approach (roughly 6 seconds per frame).

On the other hand, Yeung et al [20] propose a real-time solution for a camera network composed of two orthogonal Kinects. While the solution is fast, it is not clear if it scales to different number and position of sensors.

Our work exploits multiple sensors and their extrinsic calibration in a novel way. We propose a depth-based fusion of the data gathered by the sensors which are then frontalized with respect to a reference camera. A virtual depth image resulting from the back-projection of the fused points is then fed to a state-of-the-art depth-based skeletal tracker [11] to estimate the body poses of the different persons in the scene. Experimental results show that we improved state-of-the-art performance for different sequences involving different persons.

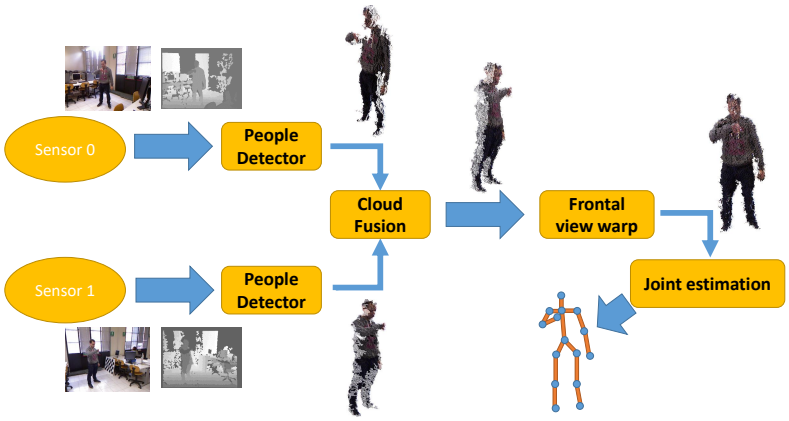

Figure 1: The system overview with a 2-sensor network. At each new frame the sensors in the network provide the master computer with new synchronized point clouds built from the depth images. After an initial fusion, the master computer warps the data to a frontal view in order to obtain the best results and estimate the joint locations of each person in the scene.

3. Algorithm Details

Figure 2 shows the overview of the system proposed. The input of the algorithm comes from a camera network composed of a set of sensors $\mathbf{C} = \{C_1, C_2, ..., C_N\}$ with $N \geq 1$, where each C_i provides at time t a frame $\mathcal{F}_i(t) = \{D_i(t), I_i(t)\}$, where $D_i(t)$ is the depth image and $I_i(t)$ is the RGB image. It is worth noting that the RGB image is here used just to colorize the point cloud, but it has no effect on the algorithm results, thus it can be omitted from $\mathcal{F}_i(t)$. Starting from this input, each camera C_i locally computes the point cloud $\mathcal{P}_i(t)$ using its intrinsic calibration matrix K_i. In particular, given a value d_i of a point $(x, y) \in D_i(t)$, it is possible to compute its 3D projection p_i as shown in Equation 1. As a final step, each camera sends the point cloud to the master computer. This node is in charge of computing the multi-view result by fusing the different views. The algorithm requires a set of synchronized clouds

[2] https://github.com/openptrack/open_ptrack_v2

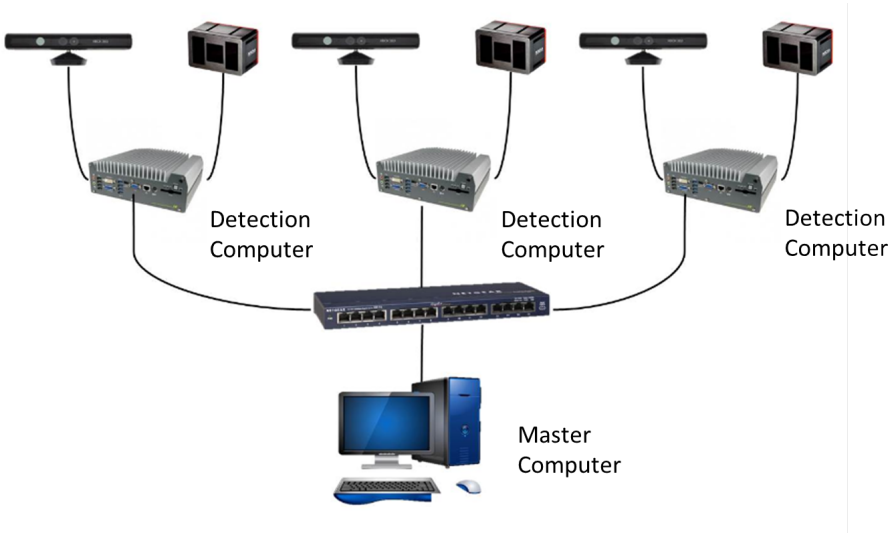

Figure 2: The topology of the camera network. Each sensor is connected to a computer which is sending its depth information through the network. One of the computers, here called master computer, is in charge of reading the information from the others and computing the multi-view body pose estimation. Image courtesy of [21]

$\mathbf{P}(t) = \{{}^{C_i}\mathcal{P}_i(t) \,|\, 1 \leq i \leq N\}$ coming from each $C_i \in \mathbf{C}$. Whenever a new $\mathbf{P}(t)$ arrives, the master node transforms the different clouds in $\mathbf{P}(t)$ in the common world reference frame using the transformation explained in Section 3.1 and then computes the multi-view results as explained in the remainder of the paper. In order to validate our approach, for the sake of simplicity, the results and each figure we reported are referred to a 2-camera network, but it is worth noting that the same approach can be easily applied to a larger number of sensors.

$$\forall D_i(t) \quad \mathcal{P}_i(t) = \{\mathbf{p_i} = (X, Y, Z, R, G, B) \,|\, (X, Y, Z) \in \mathbb{R}^3, (R, G, B) \in \mathbb{N}^3\}$$

$$d_i \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K_i \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \; p_i = \begin{cases} X = \frac{(x - c_x) d_i}{f_x} \\ Y = \frac{(y - c_y) d_i}{f_y} \\ Z = d_i \end{cases}, \quad K_i = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

3.1. Network Calibration

In order to exploit the different sensors in a camera network, the set of roto-translations between all of them needs to be known. This procedure is usually referred to as extrinsic calibration of the camera network. In this work, we use the library described in [21] which extrinsically calibrates the network by using a checkerboard and then refines the results by exploiting the detection of a person in the scene. As a result of this procedure, we fix a world reference frame $\mathcal{W}$, which is placed with the y-axis pointing to the projection of a reference camera, here referred to as C_0. The set of transformations available in the system at this point are indicated in the remainder of the paper as $\mathfrak{T}_{\mathsf{C}_i}^{\mathcal{W}}\big|_{0\leq i\leq N}$. In particular, each transformation $\mathfrak{T}_{\mathsf{C}_i}^{\mathcal{W}}$ is the roto-translation to be applied to a point to change its reference system from C_i to $\mathcal{W}$. Wherever necessary, we will indicate a point cloud with the term ${}^{\mathcal{W}}\mathsf{P}$ to show that the point cloud P is expressed in the $\mathcal{W}$ reference system. it is worth noting that the reference camera C_0, which in our tests corresponds to a real camera, can also be a virtual one C^*, given that the user can define its position with respect to other cameras.

3.2. Single-view skeletal tracking

The work presented in [11] is one of the few open-source depth-based skeletal trackers. One of the main contribution of this paper is to enhance the results of [11] by introducing a people detection module prior to the joints estimation. Although the part based classification provided by [11] is background independent, it is difficult to associate the right joints where the person is, since also the background pixels are classified as body parts. Background subtraction may be of interest to solve this issue, but it is only applicable to static cameras and static backgrounds. For this reason, we decided to exploit the people detection performed by *OpenPTrack* [21, 22, 23], which is background independent and relies only on depth information to preserve the light invariant capability. As depicted in Figure 2, the people detection module is run by each detector and the results are sent to the master computer which is in charge of fusing them.

In particular, the master computer needs the cloud data belonging to the persons in the scene. To this end, the people detector message is enriched with the point cloud data belonging to each person detected. Since the amount of data are big and it may result in network congestion, a sub-sampling is performed by means of voxelization. Indeed, a point cloud of 640x480 points is usually encoded with 20MB. Since we need the data of all the persons in the scene for each camera, the data exchanged via network for a frame can easily be of the order of hundreds of MegaBytes. By using a voxelization of 0.02 cm as leaf size, we are approximating each cloud by keeping its surface information, resulting in a point could encoded with approximately 0.6MB.

3.3. Multi-view Data Fusion

(a) (b) (c) (d)

Figure 3: An example of the cloud fusion performed by the master computer with a 2-camera network. (a) and (b) depict the single-view point clouds taken from the two cameras, while (c) and (d) show two different views of the fused point cloud.

At this point, the data are fused and skeletal estimation and tracking is performed. Given the set of clouds $\mathbf{P}(t)$, each point cloud is referred to the world reference system $\mathcal{W}$ and they are merged by means of the Iterative Closest Point algorithm [24] to overcome the possible calibration misalignments. Formally,

given the set $\mathbf{P}(t)$ the multi-view fusion algorithm computes the point cloud $P_f(t)$ obtained as explained in Equation 2:

$$ {}^{\mathcal{W}}P_f(t) = \left(\mathfrak{T}_{C_0}^{\mathcal{W}} \cdot {}^{C_0}P_0(t)\right) \bigoplus_{1 \leq i < N} \left(\mathfrak{T}_{ICP}^{i} \cdot \mathfrak{T}_{C_i}^{\mathcal{W}} \cdot {}^{C_i}P_i(t)\right) \tag{2} $$

where the operator $\oplus$ operates on two point clouds and returns a new one which is composed of the union of the two set of points. The transformation $\mathfrak{T}_{ICP}^{i}$ is the one obtained by the ICP algorithm [24] to align the cloud ${}^{\mathcal{W}}P_i$ to the reference cloud ${}^{\mathcal{W}}P_0$.

As explained in Section 4, the entire algorithm achieves real-time framerates using a two Kinect v1 sensor network. Nevertheless, the computational burden required by Equation 2 is linear in the number of cameras and persons in the scene. In order to cope with this problem, it is possible to remove the ICP pre-alignment of the clouds, thus removing the $\mathfrak{T}_{ICP}^{i}$ element from Equation 2. Indeed, ICP is particularly effective for sensors that have large errors in depth estimation when the target distance increases. For smaller camera networks or camera networks that use other types of sensors (e.g. the Kinect v2), this step can be skipped, reducing the computational load of the entire pipeline.

3.4. Frontal view warping

Depth-based skeletal trackers are not yet able to achieve the great performance shown by their RGB counterparts in presence of occlusions and non-frontal views. In order to exploit at the best the single-view algorithm, we perform a frontal view warping of the point cloud $P_f(t)$. Given the point cloud $P_f(t)$, the goal of the frontal view warping algorithm is to compute $P_{fv}(t)$ which is frontal with respect to the reference camera C_0. In order to achieve this goal, we project the points of $P_f(t)$ to the x-y plane of the world coordinate system, obtaining the cloud P_f^{xy}:

$$ P_f^{xy} = \left\{ p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \widehat{p} \quad | \quad \forall \widehat{p} \in P_f(t) \right\} \tag{3} $$

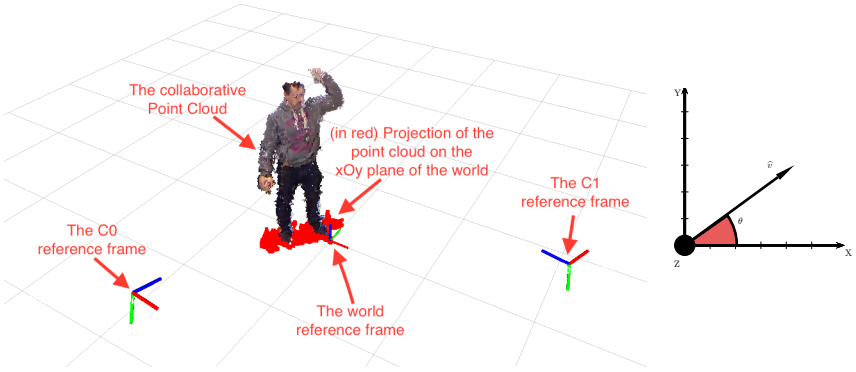

Figure 4: The frontal view-warping algorithm. On the left, the fused point cloud in native colors is shown on top of its projection on the xOy plane of the world reference system P_f^{xy} (shown in red). On the right, the principal component $\widehat{v}$ translated to the origin is shown. The angle to be applied to frontalize the cloud is θ.

and given the covariance matrix $\Sigma_{P_f^{xy}(t)}$ of $P_f^{xy}(t)$:

$$\Sigma_{P_f^{xy}(t)} = \frac{1}{||P_f^{xy}(t)||} \sum_{i=1}^{||P_f^{xy}(t)||} (p_i - \overline{p})(p_i - \overline{p})^T, \quad \overline{p} = \frac{1}{||P_f^{xy}(t)||} \sum_{i=1}^{||P_f^{xy}(t)||} p_i \quad (4)$$

we compute the two principal components [25] v_1, v_2 of P_{fv}^{xy}:

$$\exists \lambda_1, \lambda_2 \in \mathbb{R} \quad | \quad v_i \lambda_i = \Sigma_{P_f^{xy}(t)} \lambda_i \, , 1 \leq i \leq 2 \quad (5)$$

The two vectors v_1 and v_2, by definition, lie on the axis of P_f^{xy} which, by construction, represents an oval shape. The transformation needed to obtain P_{fv} can be described as a rotation around the z-axis of the world reference frame of the cloud P_f^{xy}. The magnitude of the rotation is given by the angle θ which is the angle between the $u_x = (1, 0, 0)$, versor of the orthonormal basis $[u_x, u_y, u_z]$ that identifies $\mathcal{W}$, and the vector $\widehat{v} = \max(v_1, v_2)$. Mathematically, given θ:

$$\theta = \arccos\left(\frac{\widehat{v} \cdot u_x}{||\widehat{v}||}\right) \quad (6)$$

and knowing $\overline{p}$, the P_f^{xy} centroid already defined in Equation 4, the transformation to be applied can be formulated as expressed in Equation 7.

$$\mathfrak{T}_{fv} = \begin{pmatrix} \cos\theta & \sin\theta & 0 & -\overline{p}_x \\ -\sin\theta & \cos\theta & 0 & -\overline{p}_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

The final point cloud $P_{fv}(t)$ is finally obtained by applying the transformation $\mathfrak{T}_{fv}$ to the original fused point cloud $P_f(t)$:

$$P_{fv}(t) = \mathfrak{T}_{fv} \cdot P_f(t) \tag{8}$$

It is worth noting that the arms may introduce noise in the frontalization of a specific pose, since they usually are not aligned with the human orientation. Nevertheless, the points generated by the arms are less than those generated by the body, resulting in a lower contribution of the calculation of the eigenvectors v_1 and v_2. As a future work, we will remove the biggest part of this noise by projecting the points belonging to a convex hull around the center of the body of a person.

3.5. Synthethic Depth Generation

The single-view depth-based skeletal tracker described by Buys [11] uses a random forest trained to classify each pixel of a depth image as part of a body part of a human limb[3]. In order to generate the same input data, we backproject $P_{fv}(t)$ to the C_0 camera image plane obtaining a virtual depth image as explained in Equation 9.

$$D^*(t) = \{d_{ij} = (i,j) \in \mathbb{N}^2 \,|\, i \in (0,480),\, j \in (0,640)\} \tag{9}$$

Since we perform a voxelization of the point clouds to avoid congestions in the network, at this point it may result that the virtual depth image contains holes.

[3]We are using the term *limb* to improperly refer to a human body part, i.e. the part between two adjacent joints

For overcoming this problem that seriously affects the overall performance of the skeletal tracker, we applied the following depth filling formula:

$$\forall d_{ij} \in D^*(t), \qquad d_{ij} = \begin{cases} z, & \exists (x, y, z) \in P_{fv} \,|\, K_0 \cdot [x, y, z]^T = z[i, j, 1]^T \\ \overline{z}, & \exists (x, y, \overline{z}) \in P_{fv} \,|\, K_0 \cdot [x, y, \overline{z}]^T = \overline{z}[\overline{i, j}, 1]^T, \\ & \text{with } (\overline{i, j}) = \underset{\widehat{(i,j)}}{\arg\min} \|(i, j) - \widehat{(i, j)}\| < \epsilon \\ +\infty, & \text{otherwise} \end{cases} \tag{10}$$

The depth filling defined by Equation 10 by scanning each point d_{ij} of the depth image $D^*(t)$. If d_{ij} is the backprojection of a point in the cloud (we do know this by using the intrinsics of the reference camera), it assumes its z value. Otherwise, the algorithm looks for the closest backprojected point in its neighborhood (defined by parameter ϵ. If this is found, its value is associated to d_{ij}, otherwise, d_{ij} assumes a very high value. In this way, these points for which we have no cloud information cannot interfere with the joint estimation algorithm explained in Section 3.6.

Figure 5 shows an example of the depth backprojection and hole-filling algorithm.

Figure 5: An example of the hole-filling algorithm for the virtual depth image generation. On the left, the original backprojected virtual depth image. On the right, the same image after the application of the algorithm. For visualization purposes, we changed to black all the valid depth value, while in white we show missing or very high depth values.

3.6. Joint Estimation and Tracking

The output of the body part detector [11] is a function $\mathcal{F} : \mathsf{D} \rightarrow \mathcal{L}$, where $\mathcal{L}$, defined at training time, is a set of 24 labels each one associated to a different body part. In this way, is particularly difficult to detect the joint position for each person in the scene, since also the pixel belonging to clutter or background are classified as belonging to a body part.

Nevertheless, by using the improvements already explained in the previous sections, the algorithms classifies an image where it exactly knows where the pixels belonging to a person are, thus improving the overall accuracy. By applying [11] to $\mathsf{D}^*(\mathsf{t})$, we calculate the body joints locations following two steps. Firstly, we consider the problem of a single label that could be assigned to multiple coherent groups of voxels. Two simple methods to solve this problem are to group the coherent voxels into a single blob or to sort the voxels by their size and consider only the largest one for the joint position calculation. However, whenever the distinction of the smaller and bigger body parts are not so clear, as it commonly happens with hands or elbows, such methods do not provide good results. For this reason, we consider an optimal tree of the body parts, starting from the *Neck* as the root joint and recursively estimating the child-blobs. The method is therefore based on a pre-defined skeleton structure, which defines whether a body part is linked to another one as well as an expected size of the body limbs.

The 3D centroid of each blob is then used for the computation of the joint location. In most cases, the centroid is a good estimation of the joint location itself, but a different algorithm is used for estimating the hip, shoulder and elbow joints as described below:

- **Hip:** The original hip blob returned by the body part detector is generally big and includes part of the torso. The hip centroid is therefore selected by considering only the lower part of the initial hip blob;

Algorithm 1 .

Input: $Xm = [Xm_1, ..., Xm_n]$ - measured values of m body joints; $Xp = [Xp_1, ..., Xp_n]$ - previous values of m body joints; $V = [V_1, ..., V_n]$ - velocities of the body joints

Output: $X = [X_1, ..., X_n]$ - estimated joint positions; $V = [V_1, ..., V_n]$ - updated velocities

For each body joint k in $\{1, n\}$:

Calculate the predicted position: $X_k = Xp_k + V_k * dt$

Difference between measured and predicted: $R_k = Xm_k - X_k$

New joint position value: $X_k = X_k + \alpha * R_k$

New joint velocity value: $V_k = V_k + (\beta * R_k)/dt$

- **Shoulder:** Given the chest blob, we estimate the sub-voxel V_{y_max} with the maximum Y-value. We further embed voxels belonging to the chest blob to V_{y_max} while mantaining the maximum distance wrt V_{y_max} below a threshold (i.e. 10 cm). Finally, we use the centroid of this sub-blob as the location of the shoulder;

- **Elbow:** The elbow is one of the smallest part returned by the body part detector. Moreover, for some frames, the elbow is not detected at all. In this case, we consider as the elbow location, the point of the arm blob which has the longest distance from the previously estimated shoulder point. Otherwise, we use the normal approach already explained.

The joint estimation algorithm is further refined with a alpha-beta tracking algorithm described in Algorithm 1. In this way, we ensure a continous motion and smoothing of the joint locations over time. Each new joint position is estimated starting from the predicted and measured position, weighted with the α parameter, and its velocity update, weighted with the β parameter. Those parameters are different only for the hands joints which normally have higher velocity and therefore faster trajectories.

4. Experiments

	head	neck	shoulder	elbow	wrist	hip	knee	ankle	torso
single-view [11]	31,73	39,09	175,57	86,88	144,88	139,35	132,27	107,42	109,65
single-view ours	**11,35**	8,96	18,82	28,44	80,35	38,68	22,73	**36,93**	51,74
multi-view [16, 17]	15,48	15,43	17,81	**26,74**	61,19	48,48	24,46	43,53	62,05
multi-view ours	11,59	**7,02**	**17,49**	30,87	**44,57**	**19,68**	**21,50**	39,18	**20,95**

	overall
single-view [11]	107,43
single-view ours	33,11
multi-view [16, 17]	35,02
multi-view ours	**23,65**

Table 1: Numerical results of the proposed algorithms with respect to the state-of-the-art. The numbers expressed are obtained using Equations 11 and 12 respectively for the top and bottom part of the table. All the numbers are in millimeters.

In order to evaluate the method proposed in this work, we recorded a set of RGB-D frames from a 2-Kinect v1 sensor network. In particular, the set of frames involves two different persons performing different movements in front of the cameras. For the sake of comparison, we manually annotated the ground truth for the set of recorded frames taken from the reference camera. For each joint j, let $\mathcal{B}$ be the set of dataset frames and $L_j(i)$, $G_j(i)$, respectively, the location of the joint j on frame i returned by the algorithm considered and its ground truth, the numerical comparison is performed by means of the reprojection error defined in Equation 11.

$$e_j = \frac{1}{||\mathcal{B}||} \sum_{i \in \mathcal{B}} ||G_j(i) - L_j(i)|| \tag{11}$$

Since the methods considered are very different between each other, we highlighted the overall performance of each of them in the last part of Table 1 by averaging e_j over all the joints and both sequences:

$$e^* = \frac{1}{||\mathcal{B}||} \sum_{j=0}^{||\mathcal{B}||} e_j \tag{12}$$

Figures 6 and 7 show some example frames of the different tested methods. All the 3D skeletons obtained by the different methods have been reprojected on the reference camera C_0.

The results achieved highlight how the original depth based skeletal tracker is good at estimating body parts, but not at distinguishing between background and foreground. Indeed, the proposed single-view skeletal tracker greatly improved the performance achieved, as shown in the first two rows of Table 1.

In order to evaluate the accuracy of the proposed multi-view algorithm, we used as a baseline a state-of-the-art multi-view approach described in [16, 17]. This approach computes the final 3D skeleton of each person in the scene by means of triangulation of the single-view 2D ones. In order to be as fair as possible in our comparison, we used the same improved skeletal tracker as a feed for both algorithms. The results reported in Table 1 show that the performance obtained by our approach is around 30% better than the baseline. Indeed, the strong point of our approach is that the data are fused and warped in order to get the best possible result from the single-view skeletal tracker. Approaches based on the fusion of single-view skeletons, like the baseline, are not able to distinguish between noisy and good joints.

Regarding runtime, our algorithm is able to generate multi-view skeletons at about 10 fps, while using a non-optimized version of the code on a Intel i7 equipped with a NVidia GTX 670 GPU.

5. Conclusions

In this work, we proposed several improvements regarding the subject of depth-based single and multi-view skeletal tracking algorithms.

The first contribution is an extension to Buys' skeletal tracker [11] with a prior people detection phase. While the original algorithm is able to correctly classify the pixels of each person in the scene, it is not able to distinguish between pixels belonging to the background of an image. On the contrary, the proposed algorithm is able to generate a virtual depth image that keeps the depth value

of the pixels belonging to the persons in the image, while removing the other ones. The results achieved demonstrate how our approach greatly enhances the accuracy in terms of re-projection error of the detected skeleton joints.

The second contribution of this paper is about multi-view depth-based skeletal tracking. The proposed approach extracts the best performance from the single-view skeletal tracker by presenting a frontal-view warping of the fused data coming from the camera network. The results highlight how this approach can overcome a state-of-the-art multi-view skeletal tracker algorithm [16, 17] that exploits a fusion at the skeleton level rather than at the depth data level. The overall improvement is around 30% in terms of the re-projection error described in Equation 11.

A limitation of the current approach is the fact that it does not address the case of multiple persons together in the same scene. To this aim, it is anyway possible to combine the proposed approach together with a state-of-the-art data association algorithm [18, 21, 22].

Given the low computational burden required, the algorithms described in this work can be used for building real-time marker-less body pose estimation camera networks that work also in dimmer scenes and with a low number of cameras. Moreover, given the nature of the algorithm, the more cameras are in the network, the more informative and accurate will be the final depth image to perform body pose estimation. In particular, a better coverage of the field-of-view will result in better accuracy of the overall algorithm.

References

[1] Y. Du, W. Wang, L. Wang, Hierarchical recurrent neural network for skeleton based action recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1110–1118.

[2] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moore, Real-time human pose recognition in parts from single depth images, Communications of the ACM 56 (1) (2013) 116–124.

[3] A. Aguado, I. Rodríguez, E. Lazkano, B. Sierra, Supervised+ unsupervised classification for human pose estimation with rgb-d images: a first step towards a rehabilitation system, in: Converging Clinical and Engineering Research on Neurorehabilitation II, Springer, 2017, pp. 795–800.

[4] K. Ehlers, K. Brama, A human-robot interaction interface for mobile and stationary robots based on real-time 3d human body and hand-finger pose estimation, in: Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on, IEEE, 2016, pp. 1–6.

[5] F. Han, X. Yang, C. Reardon, Y. Zhang, H. Zhang, Simultaneous feature and body-part learning for real-time robot awareness of human behaviors, arXiv preprint arXiv:1702.07474.

[6] Z. Cao, T. Simon, S.-E. Wei, Y. Sheikh, Realtime multi-person 2d pose estimation using part affinity fields, in: CVPR, 2017.

[7] E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, B. Schiele, S. I. Campus, Arttrack: Articulated multi-person tracking in the wild, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 4327, 2017.

[8] S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni, E. Menegatti, Performance evaluation of the 1st and 2nd generation kinect for multimedia applications, in: Multimedia and Expo (ICME), 2015 IEEE International Conference on, IEEE, 2015, pp. 1–6.

[9] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, B. Schiele, Deepercut: A deeper, stronger, and faster multi-person pose estimation model, in: European Conference on Computer Vision, Springer, 2016, pp. 34–50.

[10] R. B. Rusu, S. Cousins, 3d is here: Point cloud library (pcl), in: Robotics and automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 1–4.

[11] K. Buys, C. Cagniart, A. Baksheev, T. De Laet, J. De Schutter, C. Pantofaru, An adaptable system for RGB-D based human body detection and pose estimation, Journal of visual communication and image representation 25 (1) (2014) 39–52.

[12] S. Wu, H. Yang, S. Zheng, H. Su, Y. Fan, M.-H. Yang, Crowd behavior analysis via curl and divergence of motion trajectories, International Journal of Computer Vision 123 (3) (2017) 499–519.

[13] R. Vezzani, D. Baltieri, R. Cucchiara, People reidentification in surveillance and forensics: A survey, ACM Computing Surveys (CSUR) 46 (2) (2013) 29.

[14] M. Munaro, A. Basso, A. Fossati, L. Van Gool, E. Menegatti, 3d reconstruction of freely moving persons for re-identification with a depth sensor, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, 2014, pp. 4512–4519.

[15] A. Kanaujia, N. Haering, G. Taylor, C. Bregler, 3d human pose and shape estimation from multi-view imagery, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on, IEEE, 2011, pp. 49–56.

[16] M. Lora, S. Ghidoni, M. Munaro, E. Menegatti, A geometric approach to multiple viewpoint human body pose estimation, in: Mobile Robots (ECMR), 2015 European Conference on, IEEE, 2015, pp. 1–6.

[17] S. Ghidoni, M. Munaro, A multi-viewpoint feature-based re-identification system driven by skeleton keypoints, Robotics and Autonomous Systems 90 (2017) 45–54.

[18] M. Carraro, M. Munaro, J. Burke, E. Menegatti, Real-time marker-less multi-person 3d pose estimation in rgb-depth camera networks, arXiv preprint arXiv:1710.06235.

[19] Z. Gao, Y. Yu, Y. Zhou, S. Du, Leveraging two kinect sensors for accurate full-body motion capture, Sensors 15 (9) (2015) 24297–24317.

[20] K.-Y. Yeung, T.-H. Kwok, C. C. Wang, Improved skeleton tracking by duplex kinects: A practical approach for real-time applications, Journal of Computing and Information Science in Engineering 13 (4) (2013) 041007.

[21] M. Munaro, F. Basso, E. Menegatti, Openptrack: Open source multi-camera calibration and people tracking for rgb-d camera networks, Robotics and Autonomous Systems 75 (2016) 525–538.

[22] M. Munaro, A. Horn, R. Illum, J. Burke, R. B. Rusu, Openptrack: People tracking for heterogeneous networks of color-depth cameras, in: IAS-13 Workshop Proceedings: 1st Intl. Workshop on 3D Robot Perception with Point Cloud Library, 2014, pp. 235–247.

[23] M. Munaro, E. Menegatti, Fast rgb-d people tracking for service robots, Autonomous Robots 37 (3) (2014) 227–242.

[24] P. J. Besl, N. D. McKay, et al., A method for registration of 3-d shapes, IEEE Transactions on pattern analysis and machine intelligence 14 (2) (1992) 239–256.

[25] C. R. Rao, The use and interpretation of principal component analysis in applied research, Sankhyā: The Indian Journal of Statistics, Series A (1964) 329–358.

Figure 6: Some results of the single-view algorithm. On the left, the original skeletal tracker algorithm, while, on the right, the proposed one.

Figure 7: Some results of the multi-view algorithm. On the left, the baseline approach, while, on the right, our proposed multi-view algorithm.