Improved skeleton estimation by means of depth data fusion from multiple depth cameras

Marco Carraro[1], Matteo Munaro[1], Alina Roitberg[2], and Emanuele Menegatti[1]

[1] University of Padova, Department of Information Engineering, Via Gradenigo 6/A,
35131 - Padova, Italy
{marco.carraro, matteo.munaro, emg}@dei.unipd.it,
[2] Technische Universität München, Boltzmannstraße 3 85748 Garching bei München
Germany
roitberg@in.tum.de

Abstract. In this work, we address the problem of human skeleton estimation when multiple depth cameras are available. We propose a system that takes advantage of the knowledge of the camera poses to create a collaborative virtual depth image of the person in the scene which consists of points from all the cameras and that represents the person in a frontal pose. This depth image is fed as input to the open-source body part detector in the *Point Cloud Library*. A further contribution of this work is the improvement of this detector obtained by introducing two new components: as a pre-processing, a people detector is applied to remove the background from the depth map before estimating the skeleton, while an alpha-beta tracking is added as a post-processing step for filtering the obtained joint positions over time. The overall system has been proven to effectively improve the skeleton estimation on two sequences of people in different poses acquired from two first-generation Microsoft Kinect.

Keywords: Skeleton estimation, Body parts estimation, multiple depth cameras, PCL, OpenPTrack

1 Introduction

The capability to segment the body parts or, more generally, to estimate a skeleton of a person in an unsupervised way is fundamental for many applications: from health-care to ambient assisted living, from surveillance to action-recognition and people re-identification. The introduction of affordable RGB-D cameras as the Microsoft Kinect, has given a boost to the research in this area and many marker-less skeleton estimation algorithms were born, as Shotton's human pose recognition [1] and Buys' body pose detection [2]. However, all these systems perform better when the subject is seen frontally with respect to the depth camera, mainly because most of the training examples they were trained with referred to this pose. In this work, we want to overcome this problem when multiple cameras are available, thus generating a virtual depth image of the subject warped in frontal view after having fused the depth information coming from the cameras. Moreover, we propose an improvement to Buys' body pose detector [2], here used for skeleton estimation, by adding a preliminary people detection phase for background removal and performing an alpha-beta tracking on the final skeleton joints. The system has been tested on sequences of two freely moving persons imaged by a network composed of two first-generation Microsoft Kinect sensors. Summarizing, the contribution of this work is two-fold:

- We introduce a novel multi-view method to estimate the skeleton of a person based on the fusion of the 3D information coming from all the sensors in the network and a subsequent warping to a frontal pose;
- We improve the body pose detector in [2] by removing background points from the input depth image with a people detection phase and by adding a joint tracking filter to the output of the detector.

The remainder of the paper is organized as follows: Section 2, reviews the state-of-the-art of both single-camera and multi-camera skeleton tracking algorithms, while Section 3 gives an overview of our system. In Section 4, we describe the multi-view data fusion part of our system, while in Section 5 we describe the skeleton estimation algorithm we used and how we improved it. Finally, Section 6 details the experiments we performed and the results we achieved and in Section 7 conclusions are drawn.

2 Related Work

The skeleton of a person gives important cues on what the person is doing (action-recognition)[3,4], who is the person viewed (people re-identification)[5,6,7], what are her intentions (surveillance)[8] and how are her health conditions (health-care)[9]. Furthermore, the wide literature on people tracking [10,11,12] demonstrates its usefulness for both security applications and human-robot interaction. One of the most important works on skeleton tracking is the one by Shotton et al. [1], which trains a random forest to recognize the body parts of a person with a huge training dataset composed of real and synthetic depth images of people. The classifier, licensed by Microsoft for entertainment applications, achieves good performance and works in real-time. The system is released within the Microsoft Kinect SDK and only works with Windows-based computers. Another work released as open-source within the *Point Cloud Library*[13] is the work of Buys et al. [2], which uses an approach similar to Shotton's. In our work, we use this latter body-part detector, that we also improved by adding a people detection pre-processing phase and an alpha-beta tracking algorithm.

Intelligent surveillance systems rely more and more on camera network cooperation. Indeed, more cameras are able to cover more space and from multiple views, obtaining better 3D shapes of the subject and decreasing the probability of occlusions. Recent works relies on camera collaboration in network to enhance skeletal estimation. In [14], the skeleton obtained by single RGB images is fused with the skeleton estimated from a 3D model composed with the visual

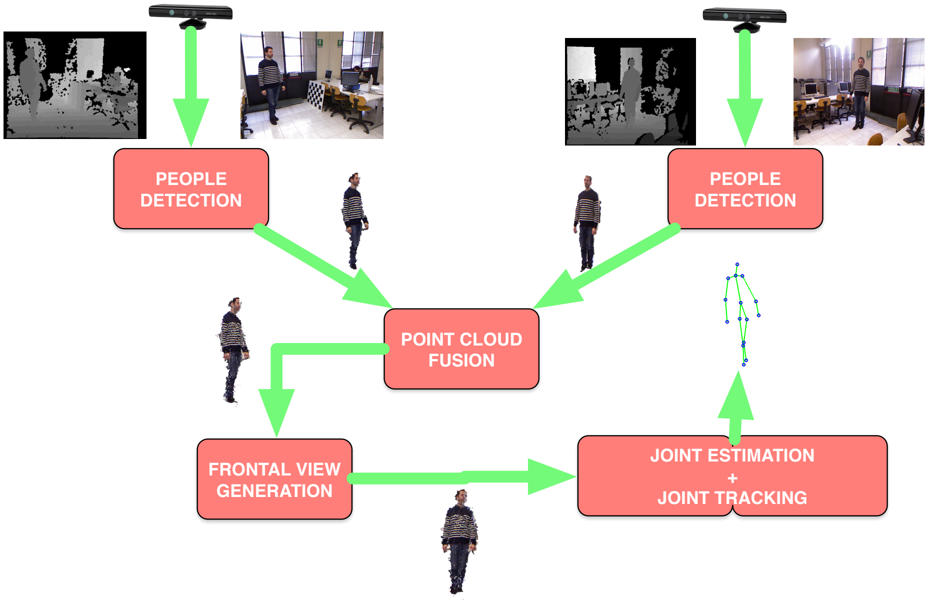


Fig. 1: Overview of the proposed system.

hull technique. The visual hull is used for refining the pose obtained from the single images. In [15], a skeleton is computed for every camera from a single image and then these estimated are projected to 3D and intersected in space. The work by Gao et al. [16] addresses this problem by registering a 3D model to the scanned point cloud obtained by two Kinects. This work is very accurate but unfeasible for real-time purposes given the 6 seconds needed to process each frame. The work of Yeung et al. [17] proposes a solution to the same problem with two Kinects that can be used in real-time. In particular, they uses two orthogonal Kinects and fuse the skeletons obtained from the Microsoft SDK with a constrained optimized framework.

In this work we exploit the multi-view information at the depth level, leaving the skeleton estimation as the last part of the pipeline. In this way, we are able to obtain better skeletons also when the single ones are potentially noisy or when they have some not tracked joints. Moreover, we minimize the skeleton estimation error by warping the fused data to a frontal view, given that the skeleton estimation is best performed from frontally viewed persons.

3 System Overview

Fig. 1 provides an overview of our system. In this work, a network composed of two first-generation Microsoft Kinect is considered, but the extension to a higher number of cameras is straightforward. At each new frame, the Kinects compute the 3D point cloud of the scene and the people detector segments only the points belonging to the persons in the scene. Afterwards, we transform the point clouds to a common reference frame given that the network is calibrated and we fuse the point cloud data after performing a fine registration with the Iterative Closest Point(ICP) algorithm [18]. The multi-view cloud obtained is then rotated and reprojected to a virtual image plane so as to generate a depth map of the persons seen from a frontal view. Then, body parts detection is performed on this virtual depth map and the joint position is computed from the body segmentation and tracked with an alpha-beta tracking filter. The obtained skeleton can then be reprojected to either of the original images. In Sections 4 and 5, we will better review each step of the proposed method.

4 Multi-view data fusion

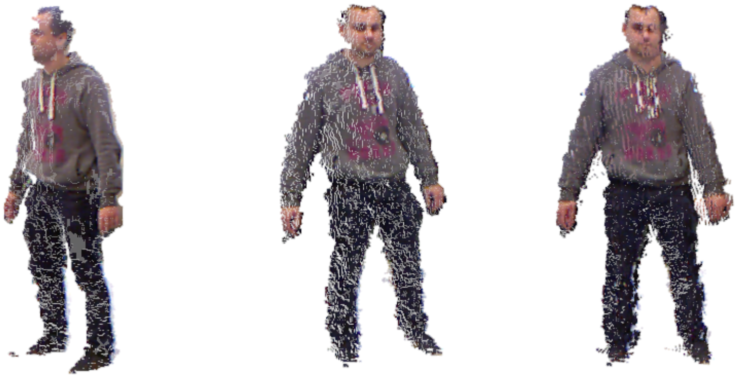
State-of-the-art body part detectors [1,2] perform poorly in presence of occlusions. This case often occurs when a person is side-viewed by a camera, and having more cameras in the scene does not ensure that one of them sees the person completely. For this reason, our system exploits the perception network to perform data fusion and frontal view generation in order to provide to the body part detector a more complete depth image of the person in the scene, thus improving the final performance.

4.1 People detection for background removal

The body part detector in [2] poorly estimates the lower body parts of a person when the ground plane is visible under the feet of the person or when the person is too close to the background. To overcome this problem, we added a people detection phase as a preprocessing for background removal. In this way, we build a new depth image where all the background points are set to a big depth value (e.g. 10 meters), so that the Random Forest in [2] can easily discard them from belonging to the foreground person. As for the people detector, we exploit the RGB-D people detection in [10,11], that is publicly available in the Point Cloud Library and allows to robustly detect people and provide the point cloud points belonging to them. Then, these 3D points are reprojected to 2D to create a masked image which can be used instead of the entire depth image, improving the output of the original body part detector.

4.2 Point cloud fusion

The information coming from multiple cameras is here exploited at the depth level, fusing the point clouds by means of the Iterative Closest Point algorithm. In particular, considering the network of two cameras C_0, C_1 we used for the experiments, we first obtain the segmented point clouds P_0, P_1 by means of the people detector and then, given the extrinsic parameters of the network, we refer these point clouds to a common *world* reference frame. After this transformation,



(a) (b) (c)

Fig. 2: An example of the depth data fusion process. In a) the point cloud obtained from the C_1 camera, in b) the point cloud obtained from the C_0 camera and in c) the final fused cloud.

the resulting point clouds P_1^w and P_0^w are finely registered by means of an ICP algorithm in order to account for depth estimation errors intrinsic of the sensors [19] or possible inaccuracies in the extrinsic calibration of the network. In formulas, we obtain the point clouds:

$$P_0^w = \mathfrak{T}_0^w(P_0) \tag{1}$$

$$P_1^w = \mathfrak{T}_1^w(P_1) \tag{2}$$

$$P_{total}^w = P_0^w \oplus \mathfrak{T}_{ICP}(P_1^w) \tag{3}$$

where, $\mathfrak{T}_i^j$ represents the transformation from the i reference frame to the j reference frame and $\mathfrak{T}_{ICP}$ is the transformation obtained by performing ICP with the point cloud P_0^w as the target cloud and the P_1^w as the source cloud. In Fig. 2, an example of this process is shown. In order to lower the time to compute $\mathfrak{T}_{ICP}$, we calculate this transformation by using two downsampled versions of P_0^w and P_1^w and by limiting to 30 the number of iterations.

4.3 Frontal view generation

The best skeleton estimation comes from frontal-viewed persons. For this reason, we want to warp the total point cloud P_{total}^w obtained at Section 4.2 to be frontal with respect to the camera we chose as a reference, here C_0. In order to obtain this result, as shown in Fig. 3, we project the points of P_{total}^w to the ground, that is the xOy plane of the *world* reference frame, thus obtaining a 2D shape that usually resembles an ellipsoid O of points. We then calculate the principal components [20] of O in order to find a vector $\widehat{v}$ with the same direction of the

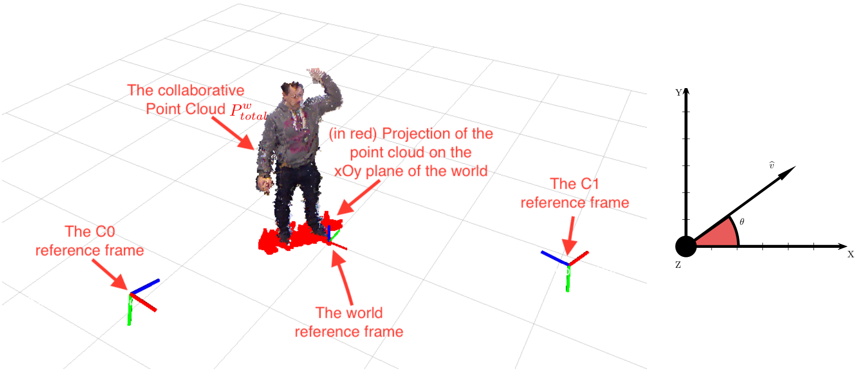

The collaborative Point Cloud P^{w}_{total}
(in red) Projection of the point cloud on the xOy plane of the world
The C1 reference frame
The C0 reference frame
The world reference frame
Y
$\widehat{v}$
θ
Z
X

Fig. 3: The frontal view generation phase. On the left there are the reference system of our Kinects and the common world reference system, the collaborative cloud obtained with ICP after the people detection phase and the same cloud projected on the xy plane of the world reference frame (visible in red). On the right is shown a top-view of the world reference frame, the vector representing the principal component $\widehat{v}$ and the angle θ which is used for the frontal view warping. Best viewed in color.

major axis of O, that is then used to rototranslate the original P^{w}_{total} with M:

$$M = \begin{bmatrix} \mathrm{R} & \mathrm{T} \\ 0^{3\times 1} & 1 \end{bmatrix} \tag{4}$$

where R is the rotation matrix which rotates a cloud of θ around the *world* z-axis and T the translation to bring the final point cloud to be centered on the *world* reference frame. In order to compute R, we need to compute θ, which is the angle between $\widehat{v}$ and $u_x = (1, 0, 0)$. In formulas, we have:

$$\theta = arccos\left(\frac{\widehat{v} \cdot u_x}{|\widehat{v}||u_x|}\right) \tag{5}$$

$$R = \begin{pmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6}$$

$$T = -\begin{pmatrix} k_x \\ k_y \\ 0 \end{pmatrix}, \qquad K = (k_x, k_y, k_z) = \frac{\sum_{i=0}^{|P^{w}_{total}|} P^{w}_{total}(i)}{|P^{w}_{total}|} \tag{7}$$

where K is the centroid of the total point cloud before the rototranslation. We can now obtain the desired frontal-view point cloud as:

$$P^{w}_{fv} = \{p = (x_p, y_p, z_p)^T \mid \exists q \in P^{w}_{total}, p = Mq\} \tag{8}$$

5 Body skeleton estimation

In this work, we use the algorithm in [2] to perform body part detection that is open source and available in the Point Cloud Library. This detector takes as input a depth image, that is then classified by a Random Forest. For this reason, the multi-view and frontal point cloud P^w_{fv} obtained in Section 4.3 has to be projected to 2D in order to create a virtual depth-image $D_{virtual}$ that could be processed by the body part detector.

5.1 Virtual depth image generation

In this work, the virtual depth image $D_{virtual}$ is estimated by projecting the points in P^w_{fv} to the image plane of the C_0 camera, that has been taken as a reference. However, this process often leaves some holes in the generated image. We thus implemented a hole-filling procedure that fills the holes with the nearest valid points until a threshold distance. In formulas:

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

$$d_{ij} = \begin{cases} P^0_{fv}(i,j), & (i,j) \text{ provides a valid point in } P^0_{fv} \\ P^0_{fv}(\overline{i,j}), & (\overline{i,j}) = argmin\{||(i,j) - (\widehat{i,j})|| < t | P^0_{fv}(\widehat{i,j}) \text{ is valid}\} \\ 10000, & \text{otherwise} \end{cases} \tag{10}$$

In Fig. 4, a comparison of a sample image with and without the hole-filling procedure is shown. This hole-filled depth map is then provided as input of the body part detector.

Fig. 4: On the left, the resultant point cloud re-projection to the image plane of the reference camera. On the right, the re-projection after the hole-filling procedure.

5.2 Joint estimation

The body part detector [2] assigns one of the 24 labels defined at training time to each pixel of the depth map and calculates the blobs of the coherent voxels with the same label.

From this preliminary segmentation, we then compute the positions of the skeleton joints in two steps. At first, we address the problem of the possibility of a single label being assigned to multiple coherent groups of voxels. This issue can be solved by combining the coherent voxel groups into a single blob or sorting them by their size with the largest one being selected for the joint calculation. Although the results of these simple methods were satisfying, the body part positions were imprecise in certain cases, especially as it comes to the smaller body parts such as hands and elbows. An improvement was achieved by building an optimal tree of the body parts, starting from the *Neck* as the root blob and further recursively estimating the child-blobs. This method is based on a predefined skeleton structure, which settles whether two body parts are connected as well as certain constraints regarding the expected size of the limbs.

In the second phase, the 3-D position of the joint is calculated from the selected blob. In most cases, the 3D centroid of the corresponding blob point cloud provides a good estimate for the joint position. An exception to this is the *Hip* blob, which also contains a large part of the torso. Besides, the *Shoulder* and *Elbow* joints are special cases which are described below.

Shoulders The shoulder position is calculated from the corresponding chest blob. Inside the blob point cloud, we estimate the voxel V_y_max with the maximum Y-value. We further build a sub-group of voxels belonging to the chest blob and having the distance to the V_y_max below a certain threshold (10 cm) and use the centroid of this sub-blob as the final position.

Elbows If the elbow blob was detected, we use the normal approach calculating the centroid of the blob. Otherwise, we estimate the point inside the arm blob, which has the longest distance from the previously estimated Shoulder joint.

Hips We define a certain threshold and build a sub-group of voxels belonging to the lower part of the hip blob. The centroid of this sub-group is used as the result position.

5.3 Joint tracking over time

The *Alpha-Beta filter* detailed in Algorithm 1 was implemented on top of the standard joint calculation to assure a consistent and continuous motion over time. This deterministic approach estimates the new position based on the predicted and measured position, with the weight of the measured position given by the parameter α, while β shows the weight of the velocity update.

Careful tuning of the α and β parameters is necessary to achieve the best results. Additionally, we have modified the update parameters for the hand joints, which usually have higher velocities then other body parts.

Algorithm 1 Applying the Alpha-Beta filter for joint tracking at a timestep t

```
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: $V_k = V_k + (\beta * R_k)/dt$
```

6 Experiments

We tested the different steps of the proposed approach with two series of RGB-D frames recorded from a network of two first-generation Microsoft Kinect. In these sequences, two different freely-moving persons were performing different movements. For measuring the accuracy, we considered the following skeleton estimation error:

$$\epsilon = \frac{\sum_{frames} \frac{\sum_{joints} ||pos_{estim} - pos_{actual}||}{N_{joints}}}{N_{frames}} \tag{11}$$

where the ground truth for joint position pos_{actual} has been manually annotated. The system used for testing the methods proposed is an Ubuntu 14.04 machine with an Intel core i7-4770 CPU and a NVidia Geforce GTX 670 GPU. In Table 1, we reported a quantitative comparison of skeleton estimation with our methods and the original one in terms of 11. In this table, we report also a baseline multi-view approach at the skeleton level in which each fused skeleton $\widehat{S}$ is the average of the single-view skeletons S_0 and S_1. While the original method [2] is independent from the background for the body pose estimation, the joint estimation algorithm is not and this cause the large ϵ obtained in our tests. Adding a people detection step, thus improve exponentially the performance gained by the joint estimator and our joint-tracking filter mantains the performance while smoothing the joints estimated. Our novel multi-view approach outperforms both the single-view skeleton estimation and the baseline multi-view method we used. The results achieved are from 20% to 33% better than the single ones and up to 24% better than the baseline skeleton-based multi-view approach. Furthermore, the computational burden needed for computing a skeleton is around 100 ms (60 ms for computing the virtual depth image plus 40 ms for the PCL skeleton computation) allowing the real-time usage of the proposed approach. In Fig. 5, we reported a qualitative comparison of skeleton estimation with these techniques.

	first person	second person
[2]	97.82	139.50
ours single-view with PD	34.35	35.01
ours single-view with PD and JT	34.50	35.35
baseline multi-view at the skeleton level	30.50	29.65
proposed multi-view approach at the depth level	**23.26**	**28.22**

Table 1: The performance achieved by the original method [2] and our method. PD stands for people detection and JT for joint tracking.

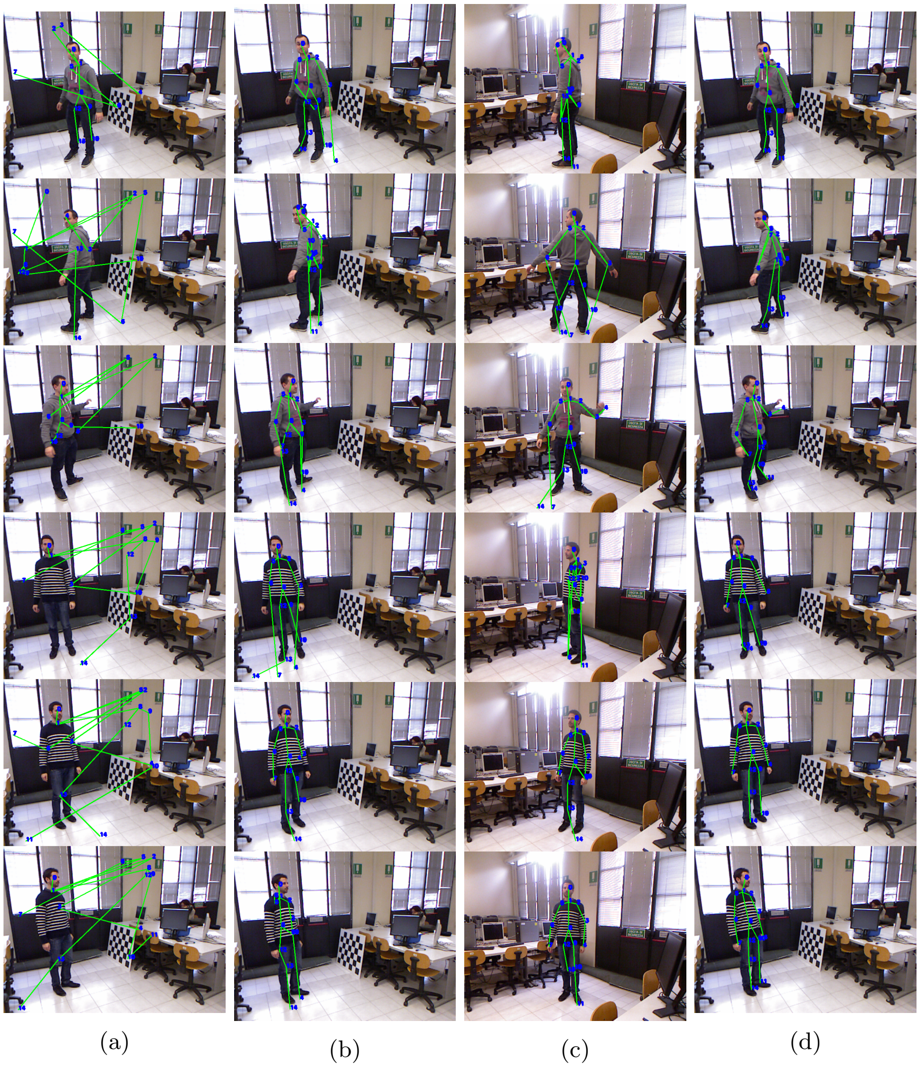
7 Conclusions

In this work, we addressed the problem of human skeleton estimation and tracking in camera networks. We proposed a novel system to fuse depth data coming from multiple cameras and to generate a frontal view of the person in order to improve the skeleton estimation that can be obtained with state-of-the-art algorithms operating on depth data. Furthermore, we improved single-camera skeletal tracking by exploiting people detection for background removal and joint tracking for filtering joint trajectories. We tested the proposed system on hundreds of frames taken from two Kinect cameras, obtaining a great improvement with respect to state-of-the-art skeletal tracking applied to each camera. The proposed approach can be also applied to real-time scenarios given the low computational burden required.

References

1. J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, “Real-time human pose recognition in parts from single depth images,” *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.
2. K. Buys, C. Cagniart, A. Baksheev, T. De Laet, J. De Schutter, and C. Pantofaru, “An adaptable system for RGB-D based human body detection and pose estimation,” *Journal of visual communication and image representation*, vol. 25, no. 1, pp. 39–52, 2014.
3. A. Roitberg, A. Perzylo, N. Somani, M. Giuliani, M. Rickert, and A. Knoll, “Human activity recognition in the context of industrial human-robot interaction,” in *Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA)*, pp. 1–10, IEEE, 2014.
4. J. Wang, Z. Liu, Y. Wu, and J. Yuan, “Mining actionlet ensemble for action recognition with depth cameras,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1290–1297, IEEE, 2012.
5. M. Munaro, S. Ghidoni, D. T. Dizmen, and E. Menegatti, “A feature-based approach to people re-identification using skeleton keypoints,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 5644–5651, IEEE, 2014.

6. M. Munaro, A. Basso, A. Fossati, L. Van Gool, and 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*, pp. 4512–4519, IEEE, 2014.
7. M. Munaro, A. Fossati, A. Basso, E. Menegatti, and L. Van Gool, “One-shot person re-identification with a consumer depth camera,” in *Person Re-Identification*, pp. 161–181, Springer, 2014.
8. J.-W. Hsieh, Y.-T. Hsu, H.-Y. M. Liao, and C.-C. Chen, “Video-based human movement analysis and its application to surveillance systems,” *Multimedia, IEEE Transactions on*, vol. 10, no. 3, pp. 372–384, 2008.
9. G. Diraco, A. Leone, and P. Siciliano, “An active vision system for fall detection and posture recognition in elderly healthcare,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pp. 1536–1541, IEEE, 2010.
10. M. Munaro, F. Basso, and E. Menegatti, “Tracking people within groups with RGB-D data,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2101–2107, IEEE, 2012.
11. M. Munaro and E. Menegatti, “Fast RGB-D people tracking for service robots,” *Autonomous Robots*, vol. 37, no. 3, pp. 227–242, 2014.
12. M. Munaro, C. Lewis, D. Chambers, P. Hvass, and E. Menegatti, “RGB-D human detection and tracking for industrial environments,” in *Intelligent Autonomous Systems 13*, pp. 1655–1668, Springer, 2016.
13. R. B. Rusu and S. Cousins, “3d is here: Point cloud library (pcl),” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, IEEE, 2011.
14. A. Kanaujia, N. Haering, G. Taylor, and 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*, pp. 49–56, IEEE, 2011.
15. M. Lora, S. Ghidoni, M. Munaro, and E. Menegatti, “A geometric approach to multiple viewpoint human body pose estimation,” in *Mobile Robots (ECMR), 2015 European Conference on*, pp. 1–6, IEEE, 2015.
16. Z. Gao, Y. Yu, Y. Zhou, and S. Du, “Leveraging two kinect sensors for accurate full-body motion capture,” *Sensors*, vol. 15, no. 9, pp. 24297–24317, 2015.
17. K.-Y. Yeung, T.-H. Kwok, and C. C. Wang, “Improved skeleton tracking by duplex kinects: A practical approach for real-time applications,” *Journal of Computing and Information Science in Engineering*, vol. 13, no. 4, p. 041007, 2013.
18. P. J. Besl and N. D. McKay, “Method for registration of 3-D shapes,” in *Robotics-DL tentative*, pp. 586–606, International Society for Optics and Photonics, 1992.
19. S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni, and E. Menegatti, “Performance evaluation of the 1st and 2nd generation Kinect for multimedia applications,” in *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, pp. 1–6, IEEE, 2015.
20. C. R. Rao, “The use and interpretation of principal component analysis in applied research,” *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 329–358, 1964.



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

Fig. 5: Some sample frames of the dataset we used for testing the proposed approach. Each row represents a frame. The different columns represent:
a) [2] on the C_0 stream;
b) ours with PD and JT on the C_0 stream;
c) ours with PD and JT on the C_1 stream;
d) our multi-view approach re-projected on the C_0 camera.