Integrating Human Body MoCaps into Blender using RGB Images

Jordi Sanchez-Riera Francesc Moreno-Noguer

Institut de Robòtica i Informàtica Industrial, CSIC-UPC
08028, Barcelona, Spain
Email: jsanchez,fmoreno@iri.upc.edu

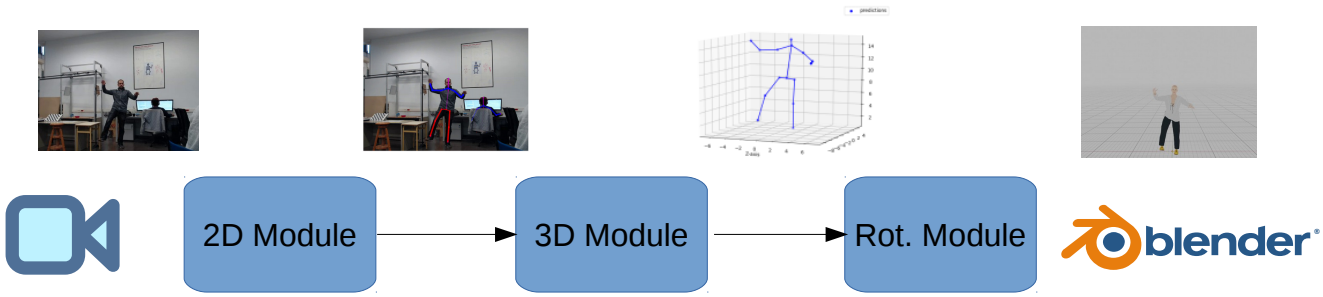


Figure 1. The proposed mocap system is composed of three modules. Given a RGB image the first module detect the body 2D joint positions, then these locations are passed to the second module that infers the 3D joint locations in world coordinates and finally the third module is responsible to communicate with Blender application and convert the 3D joint locations into rotation angles for our 3D human model.

***Abstract*—Reducing the complexity and cost of a motion capture (mocap) system has been of great interest in recent years. Unlike other systems that use depth range cameras, we present an algorithm that is capable of work as a mocap system with a single RGB camera and it is completely integrated in an off-the-shelf rendering software. This makes our system easily deployable in outdoor and unconstrained scenarios. Our approach builds upon three main modules. First, given solely one input RGB image we estimate 2D body pose; the second module estimates the 3D human pose from the previously calculated 2D coordinates and the last module calculates the necessary rotations of the joints given the goal 3D point coordinates and the 3D virtual human model. We quantitaviely evaluate the first two modules using synthetic images, and provide qualitative results of the overall system with real images recorded from a webcam.**

Keywords–MoCap; 2D, 3D human pose estimation; Synthetic human model; Action mimic.

I. Introduction

Motion capture (mocap) systems are used in industry and research to record real motions. The applications of such systems span from animating virtual characters or facial expressions, navigate into virtual reality (VR) environments, to modeling human-human/robot/object interactions. Professional mocap systems are expensive, complex to use and need of some dedicated space to record the motions, usually with multiple cameras. More modern systems, just require to wear a suit which has reflective markers or motion sensors [1], [2]. These systems store the motions into files with a standard format that can be shared with other applications. However, due to the complexity of recording and processing the data from such systems, and that not everyone can afford to acquire a mocap suite, it is not easy to find motion files processed by third parties. Or even in the case we can find public repositories with motion files as in [3] or [4], the motions we can find may not be those we need.

In order to make the mocap recordings more affordable, there have been several attempts to find alternatives to reduce the complexity and time to process the recorded data. A good example can be found in [5], where they only need two calibrated cameras to infer 3D points given by a set of reflective markers on the human body. To eliminate the burden of having to wear body/cloth markers, authors in [6] propose a new system that uses depth images, therefore 3D locations come directly from the camera sensor device. This method, however, still needs a camera calibration process, which is a tedious task. More recently, [7] eliminated the need of camera calibration proposing a system able to infer 3D joint locations from a single RGB camera. At the same time, the irruption of Kinect camera has encouraged homebrew developers to program algorithms using the API device library to achieve an inexpensive mocap system for body [8] or faces [9]. Unfortunately, the official API library is only available on Windows platforms and some of the free alternative libraries are obsolete.

Similar to [7], we propose an algorithm that is able to infer 3D body joint locations from a single RGB camera, and at the same time we integrate it to a Blender 3D modeling software [10] that allows us to generate realistic renders using Makehuman [11] 3D human model. This allows easy saving and exporting captured motions into an industry standard motion file, which could be used by other software applications. Moreover, eliminating the need of a depth sensor device, as opposite to [6], [8], we can use our system both indoors and outdoors. As illustrated in Figure 1, our algorithm is composed by three modules. The first module takes the images from

the camera and estimates the 2D joint articulations of the body. We then estimate the 3D human pose locations from the 2D detected joints, and finally the third module calculates the rotation of each joint to map from the virtual 3D human body joints to the 3D pose locations estimated by the second module.

The rest of the paper is organized as follows. In the next section is discussed the related work for the first and second implemented modules, in section III is described the developed algorithm as well as its integration to Blender, then in section IV are presented the synthetic and real performed experiments, and finally in section V are drawn the conclusions.

II. Related Work

One of the key parts for a mocap system is to have a reliable human pose detector. Exists many literature on human pose estimators, and in this section we will review the most significant algorithms in 2D human pose detection and 3D human pose estimation.

2D human pose detection. When estimating 2D human pose from a single RGB image, there are two possible different approaches. One, known as bottom-up approach, consist into find some body joint locations and then, try reason about the best configuration to match the body structure. The other one, known as top-down approach, starts from localizing the whole body region to later detect body parts. A very popular bottom-up algorithm was introduced by [12], and improved later in [13] adding optical flow information of the body parts. Both algorithms can run in several platforms, e.g. PC, phone, tablet, at very high frame rates. Top-bottom approaches, also can detect 2D poses of multiple persons at high frame rates. Most of them run first a human detector [14] to define the region where to find the body features, which make these algorithm more prone to have problems when two persons overlap to each other. Most popular top-bottom algorithms are [15], [16], and their respective faster and improved versions [17], [18]. Finally, another very popular approach [19], combines multiple bottom-up top-down layers together, named hourglass, to detect the pose at multiple resolutions. For our system, we decide to use the AlphaPose [16] method because outperforms the other above mentioned algorithms, can also run in real time, and the skeleton that retrieves is more similar to our 3D human skeleton model used in Blender, see Figure 2 a, b.

3D human pose estimation. Estimating 3D human pose from a single RGB image is an important challenge. Some of the first approaches, use previously detected 2D joint locations to train a network capable of inferring the 3D joint coordinates [20], [21], [22]. 2D and 3D detections can then be combined to make 3D pose estimation more robust [23]. However, one of the problems that arises when training the networks for these methods is the lack of labeled data. It is not easy to find ground truth for 3D human poses, for this reason, [24] proposes a method to combine labeled images with images on the wild to train a network that estimates 2D joint locations and their depth. Authors in [25], go one step further and estimate directly 3D human pose from a single image using a synthetic dataset of 5M labeled images. These kind of networks are quite complex [26], and using extra information such as temporal correlations can help to improve their performance [27]. Most recent algorithms, not only can find a 3D human pose estimate but, also can recover the whole body mesh [28] or even the shape parameters [29]. We decide to use the method described in [20] due to its simplicity for training. The network described is relatively simple, and can be trained with thousands of samples instead of millions. Moreover, network inference is very fast.

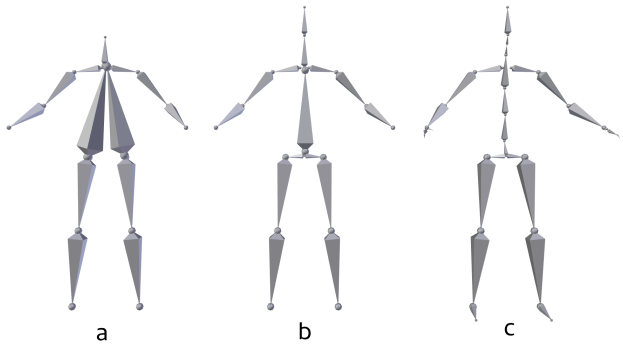


Figure 2. Different skeletons configurations. a) Left, skeleton given by CPM [12] method; b) Middle, Alpha Pose skeleton used to compute angle rotations; c) Right, skeleton used for Makehuman model to which captured motion is transferred.

III. Method

The proposed mocap system is divided into three interconnected modules. These modules will process the RGB images coming from a webcam, and finally control a 3D human model inside Blender software. We first explain the first two modules responsible to infer 3D body position from a single image. Then, we describe how these 3D joints are transformed into rotations for our 3D human model and how these rotations are passed to Blender to render the virtual model.

A. Estimate body 3D joints

Find 2D joint locations from an image. We follow Alpha Pose [16] to obtain 2D human pose estimations from a single RGB image. The method is fast, outperforms other state-of-the-art algorithms and the returned body pose is similar to the Makehuman 3D human model skeleton that we use in Blender. The method starts from some human region proposals, then these region proposals are passed through two different components. The first one is the symmetric spatial transfer network, that generates a set of pose proposals and then, a parametric pose non-maximum-suppression module selects the most plausible pose estimations. A third component, the pose-guided proposals generator, is used to augment the training data and improve the network performance.

Infer 3D joint coordinates from 2D body locations. Given a set of 2D points $x \in \mathbb{R}^2$ obtained in the previous module, we want to find a regression function that estimates the 3D points $y \in \mathbb{R}^3$ and minimizes the error over a set of 3D body poses. The regression function will be modeled by a neural network defined in [20]. This network is composed by two consecutive blocks that contain a linear layer with batch normalization, and a Rectified Linear Unit followed by a dropout layer. We use default parameters to train this network.

B. Animate a 3D model

Converting 3D coordinates to joint rotations. Our 3D human model is controlled by a hierarchical structure called skeleton. This skeleton can be seen as a directed graph, where there is a root node and non or several children for each node. Each node is a segment with its start position defines the rotation pivot point and the end position is the start of the child node. For each node (also named bone or joint), a bind matrix M_{bj} encodes the joint position and rotation of the skeleton at rest pose. Also, a pose matrix M_{pj} encodes the amount of rotation of each joint respect the rest skeleton pose. Thus, a skeleton pose $P(\theta)$ will be defined by a set of rotation parameters $\theta \in \mathbb{R}^{3K}$ for each one of the K joints that have direct correspondence with the 3D virtual model, Figure 2 c.

We want to calculate the rotation for each joint of our skeleton that matches a set of estimated 3D joint locations. The 3D joint locations, Figure 2 b, and our model skeleton, Figure 2 c, can have a different structure. Therefore, we can only find rotations for the skeleton joints that are equivalent in both structures. In order to calculate the rotations of each joint, we will define a source vector v_s starting at the parent of the joint to beginning of the child of the joint, and a target vector v_t defined by the 3D point coordinates. If $v = v_s \times v_t$, $s = ||v||$ and $c = v_s \cdot v_t$, the rotation matrix to match the two vectors v_s, v_t is defined by:

$$R = I + [v]_x + [v]_x^2 \frac{1-c}{s^2} \ , \tag{1}$$

where $[v]_x$ is the skew-symmetric cross product matrix of v.

Before we can calculate the rotation matrix is necessary to have the two sets of 3D points in the same coordinate system. Therefore, the skeleton 3D joint locations need to be transformed from local X_j^L coordinates to world X_j^W coordinates. In the case that the skeleton joint has no father, we will use Equation 2, otherwise we will use Equation 3, where M_{Tf} is the M_T matrix already calculated for the father of the current bone.

$$X_j^W = M_{bj} \cdot M_{pj} \cdot X_j^L = M_T \cdot X_j^L \tag{2}$$

$$X_j^W = M_{Tf} \cdot M_{bf}^{-1} \cdot M_{bj} \cdot M_{pj} \cdot X_j^L \tag{3}$$

Finally, to obtain the skeleton joint coordinates in Blender coordinates, we need to use the defined world matrix M_w.

$$X_j^G = M_w \cdot X_j^W \tag{4}$$

Transfer rotations to Blender. The final goal is to be able to transfer detected motion human poses from a videos or a webcam to a 3D human model in Blender, thus, apply this motion to other applications by storing it into mocap files or just simply use it as it is. To this end, we design a communication protocol with [30] library between Blender and the image stream. From one side, we will have a process that will take the images from the stream and calculate the 3D joint coordinates of the body. On the other side, we will have a python script that will take the 3D joint locations from the stream, the rest 3D joint locations of the 3D human model and calculate the body joint rotations. Therefore, we will use a server-client structure where the server will provide 3D joint locations at desired frame rate, while the client will be limited to listen the data from the server.

IV. Experiments

We perform two different kind of experiments. First we generate two sets of synthetic data, to train and evaluate the 3D human pose estimation, and then, we record several real sequences with a webcam to evaluate the whole algorithm performance.

A. Generate synthetic data

For training and evaluation purposes we generate two different kind of datasets with Blender [10] and Makehuman [11]. The first dataset is used to train the 3D pose estimation module, while the second dataset is used to evaluate the overall performance of the proposed mocap system. Note that for the 2D joint estimation module, a dataset is not needed since we use the weights trained from [16].

The first dataset consists of 6 human models (3 men and 3 women) with different shapes and sizes, performing a total of 54 motions obtained from [3] website. Therefore, we have a total of 324 sequences of approximately 100 frames each. For each sequence, the ground truth for 3D joint locations as well as their 2D projections on a camera image are stored. We set the camera resolution to be of 640×480 for both datasets.

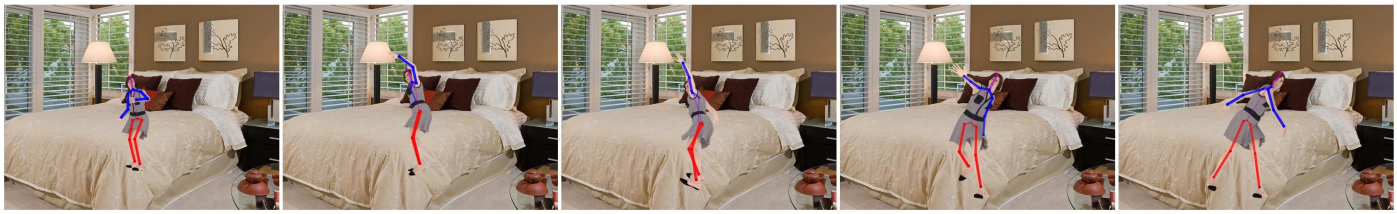
For the second dataset, we use 4 human models (2 men and 2 women) different from the ones used in the first dataset. We also use 10 motion sequences that were not included in the first dataset. In this case, the information gathered includes also the renderized RGB images of the sequences apart from the 3D and 2D ground truth locations. To make the images more close to the real world, in each generated sequence a random image is added as a background.

We also generate several real sequences each one about 30 seconds long, to evaluate qualitatively the proposed mocap system. In concrete, we record five sequences with a webcam at 640×480 image resolution with an actor performing different movements in the center of the image. The recorded images have more than one person in the scene, therefore other person detections are removed from the image with a simple post-processing that consist into keep the person with biggest bounding box in the center of the image.

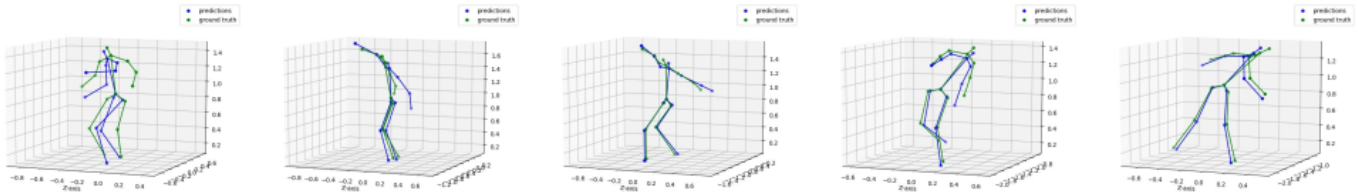
B. Evaluation

The images generated in the second dataset are passed to the Alpha Pose [16] detector network. This network returns the estimated 2D locations of 16 body joints, see 2. The estimated locations are compared with ground truth locations using the PCKh@0.5 [31] measure. This measure calculates the percentage of correct keypoints when the threshold is 50% of the head bone link. The curves for the values of each action can be seen in Figure4. We can observe that most of the actions have similar performance except for two actions: "teeter" and "picking up". In the case of "teeter" action, the motions of arms and legs are small but fast, making it seem like the person is shaking and this provokes several misdetections. In the case of "picking up" action, the model bends the upper part of the body occluding the other bottom half, making the 2D estimation module fail. However, the overall performance among all the actions, is quite accurate.

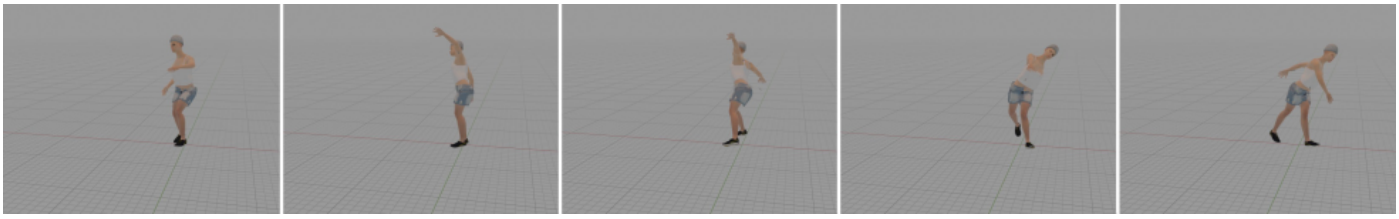
Unlike the 2D joint detection module, the 3D pose estimation module is trained from the scratch. The reason for that, is that we want to have 3D joint estimations referenced with



(a) 2D joint detections



(b) 3D pose estimation



(c) Renderized 3D human model

Figure 3. This figure shows the process for the motion capture from the acquisition of the images with a single RGB camera to the motion transfer to a 3D human model. In a) is show the 2D joint detections. In b) is show the ground truth 3D joint positions (in green) and the correspondent estimations by the 3D joint detection module (in blue). In c) is show the renderized images of the 3D human model. The input images are synthetized with Blender software. Ground truth is available.

	boxing	goalie throw	jumping jacks	looking around	picking up	talking	teeter	walking	walking 2	zombie kicking	average
error(m)	0.0961	0.0989	0.1299	0.1585	0.1399	0.0952	0.1416	0.1621	0.1535	0.1288	0.1304

TABLE I. This table shows the mean per joint position error (MPJPE) for each one of the sequence motions in the second dataset. The values are expressed in meters. We can observe that the sequence with less error is boxing while the sequence with the biggest error is walking.

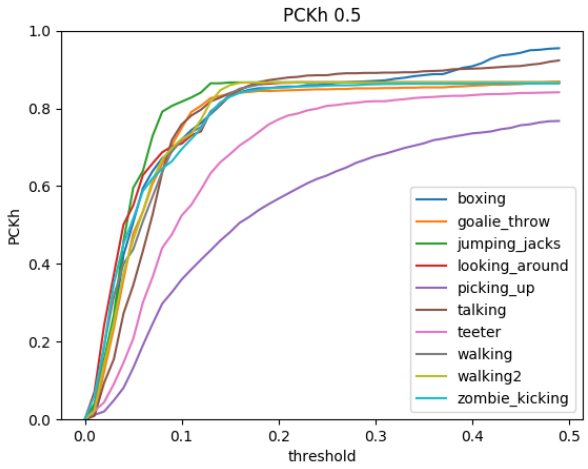
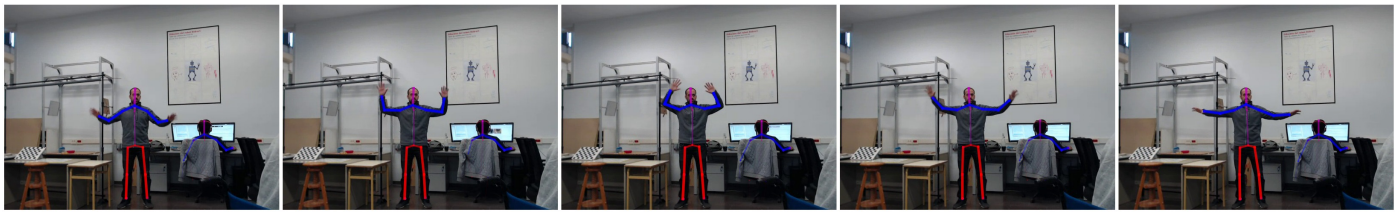


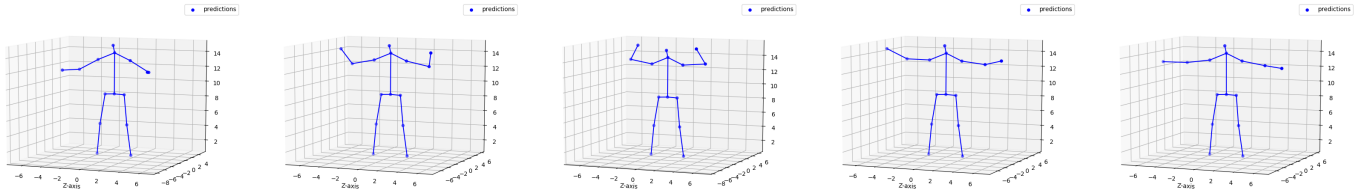
Figure 4. This figure shows the percentage of correct keypoints at 50% of the head bone link for the different actions generated in the second dataset. Most of the actions perform very similar, being "teeter" and "picking up" the ones with worst performance.

the the Blender coordinate system for a simpler calculation of rotations. Therefore, the ground truth 2D and 3D joint locations are normalized according the new generated training dataset. This means that all sequences are reoriented taking the "hips" joints as reference, and values for all joints are rescaled to values from 0 to 1. Once the 3D joint estimation network is trained and weights are obtained, we proceed to evaluate the same 10 sequences as before. In this case, to evaluate the performance of the network we use the mean per joint position error (MPJPE) given in meters [32]. For evaluation, the ground truth 2D joint locations are replaced by the estimations obtained in the 2D joint estimation module. The results can be observed in TableI. The first thing to notice is that even in the previous model the actions "teeter" and "picking up" are the ones with worst performance, in the current module the worst performance actions are "walking" and "walking 2". Therefore, the error from the previous module is not propagated as we could expect. The action with less error correspond to "boxing" which is the action where the model has less joints moving. The mean performance of all the actions is very low with an error of 0.13 meters.

In Figure 3, we show the results for 5 frames of the



(a) 2D joint detections



(b) 3D pose estimation

(c) Renderized 3D human model

Figure 5. This figure shows the process for the motion capture from the acquisition of the images with a single RGB camera to the motion transfer to a 3D human model. In a) is show the 2D joint detections. In b) is show the ground truth 3D joint positions (in green) and the correspondent estimations by the 3D joint detection module (in blue). In c) is show the renderized images of the 3D human model. We use images recorded by a webcam. Ground truth is not available.

sequence "goalie throw" for a woman actor. In each row are presented the results for each one of the proposed modules. The top row shows the results of the Alpha Pose algorithm. In the middle row we show the ground truth 3D coordinates (in green) and the 3D estimated joint values (in blue). Finally in the bottom row we show a renderized model in Blender. We can observe that the original motion is very close to the renderized motion.

In Figures 5,6, we show the results for two real sequences when the actor is performing some random movements in the center of the scene. We can observe in the first row, that in the recorded images our 2D human pose detection module is able to find two different persons in the scene. Since for our proposed mocap we want only to focus on one person, we apply a simple image processing consisting into keep biggest bounding box near to the center of the image. In the second row we show the inferred 3D human position for the only person that we want to detect. In the third row is shown our virtual 3D model once calculated rotations are applied.

V. Conclusion

We presented a system capable to perform as a mocap system with only a single RGB camera and free source software that can run in several platforms. The system is based in three components that calculates 2D human body joint locations, then from these locations infer the 3D joint world coordinates and finally, the 3D joints are transformed to joint body rotations for our virtual human model. The first two components are evaluated quantitatively with synthetic data, where the third component is evaluated qualitatively. The overall system is also evaluated in real video images with several sequences performed by a person. We show that we can mimic the movements of a person with the potential to run the whole system in real time. In the future, we could use this mocap system to perform dedicated actions for any kind of topic and extract several ground truth data like in [33].

References

[1] "XSens: MotionCapture," 2019, URL: https://www.xsens.com/ [accessed: 2019-01-02].

[2] "Vicon: MotionCapture," 2019, URL: https://www.vicon.com/ [accessed: 2019-01-02].

[3] "Adobe Mixamo," 2019, URL: https://www.mixamo.com/ [accessed: 2019-01-02].

[4] "CMU Motion Files," 2019, URL: http://mocap.cs.cmu.edu/ [accessed: 2019-01-02].

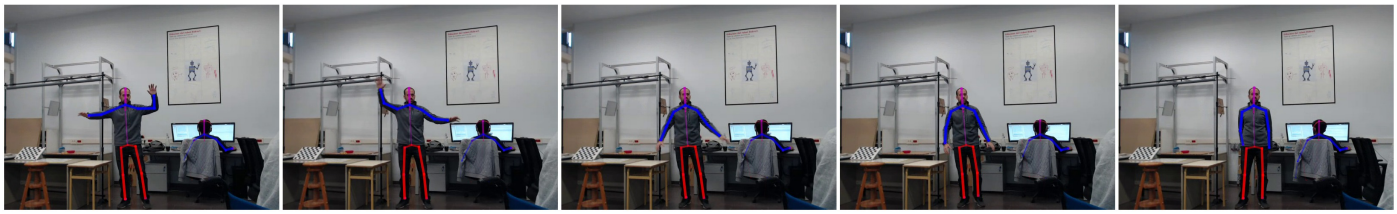
[5] J. Chai and J. K. Hodgins, "Performance animation from low-dimensional control signals," ACM Trans. Graph., vol. 24, no. 3, 2005.

[6] D. K. S. T. Varun Ganapathi, Christian Plagemann, "Real-time human pose tracking from range data," in ECCV, 2012.

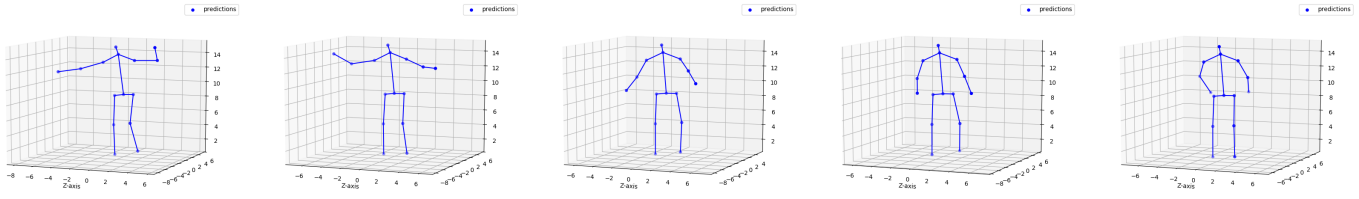
[7] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single rgb camera," vol. 36, no. 4, 2017.

[8] "Homebrew Mocap Studio," 2019, URL: https://www.youtube.com/watch?v=1UPZtS5LVvw [accessed: 2019-01-02].

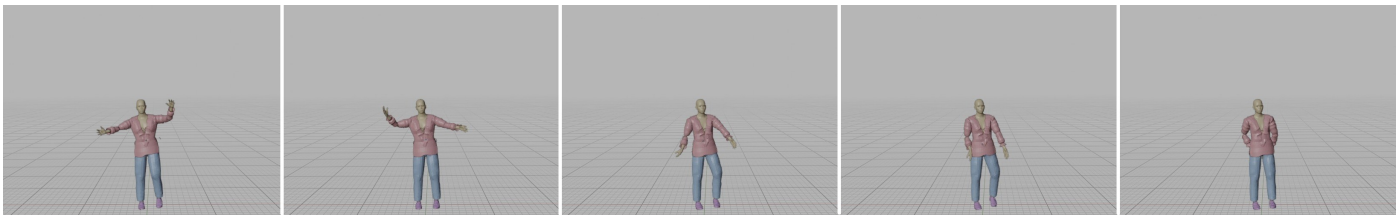
[9] "Face Mocap Studio," 2019, URL: https://brekel.com/brekel-pro-face-2/ [accessed: 2019-01-02].



(a) 2D joint detections



(b) 3D pose estimation



(c) Renderized 3D human model

Figure 6. This figure shows the process for the motion capture from the acquisition of the images with a single RGB camera to the motion transfer to a 3D human model. In a) is show the 2D joint detections. In b) is show the ground truth 3D joint positions (in green) and the correspondent estimations by the 3D joint detection module (in blue). In c) is show the renderized images of the 3D human model.

[10] "Blender," 2019, URL: https://www.blender.org/ [accessed: 2019-01-02].

[11] "Makehuman," 2019, URL: http://www.makehumancommunity.org/ [accessed: 2019-01-02].

[12] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in CVPR, 2016.

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

[14] R. Girshick, "Fast r-cnn," in IEEE International Conference on Computer Vision (ICCV), 2015.

[15] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[16] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation," in ICCV, 2017.

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

[18] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, "Pose Flow: Efficient online pose tracking," in BMVC, 2018.

[19] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in ECCV, 2016.

[20] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," in ICCV, 2017.

[21] F. Moreno-Noguer, "3d human pose estimation from a single image via distance matrix regression," in IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[22] E. Simo-Serra, C. Torras, and F. Moreno-Noguer, "3d human pose tracking priors using geodesic mixture models," vol. 122, no. 2, 2017, pp. 388–408.

[23] C.-H. Chen and D. Ramanan, "3d human pose estimation = 2d pose estimation + matching," in CVPR, 2017.

[24] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3d human pose estimation in the wild: A weakly-supervised approach," in ICCV, 2017.

[25] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen, "Synthesizing training images for boosting human 3d pose estimation," in 3DV, 2016.

[26] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression," in The European Conference on Computer Vision (ECCV), 2018.

[27] M. Lin, L. Lin, X. Liang, K. Wang, and H. Chen, "Recurrent 3d pose sequence machines," in CVPR, 2017.

[28] R. A. Güler, N. Neverova, and I. Kokkinos, "DensePose: Dense Human Pose Estimation In The Wild," in Conference on Computer Vision and Pattern Recognition (CVPR) 2018, 2018.

[29] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in Computer Vision and Pattern Regognition (CVPR), 2018.

[30] "ZMQ," 2019, URL: https://zeromq.org [accessed: 2019-01-02].

[31] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[32] L. Sigal, A. Balan, and M. J. Black, "HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," International Journal of Computer Vision, vol. 87, no. 1, 2010.

[33] A. Pumarola, J. Sanchez-Riera, G. P. T. Choi, A. Sanfeliu, and F. Moreno-Noguer, "3dpeople: Modeling the geometry of dressed humans," in International Conference in Computer Vision (ICCV), 2019.