Signal Processing 224 (2024) 109598

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

FLK: A filter with learned kinematics for real-time 3D human pose estimation

Enrico Martini [a,*], Michele Boldo [a], Nicola Bombieri [b]

[a] Department of Computer Science, University of Verona, Verona, Italy
[b] Department of Engineering for Innovation Medicine, University of Verona, Verona, Italy

ARTICLE INFO

Keywords:
Human motion refinement
Human pose estimation
Filtering
Denoising
Completion
Kalman filter

ABSTRACT

There is a growing interest in adopting 3D human pose estimation in safety-critical systems, from healthcare to Industry 5.0. Nevertheless, when applied in such settings, these neural networks may suffer from estimation inaccuracy. Besides imprecise or inconsistent annotations in the training dataset, the inaccuracy is caused by poor image quality, rare poses, dropped frames, or heavy occlusions in the scene. In addition, these scenarios often require the software results to have temporal constraints, such as real-time and zero- or low-latency, which make many of the filtering solutions proposed in the literature inapplicable. This paper proposes FLK, a Filter with Learned Kinematics, to refine 3D human motion data in real-time and at zero/low latency. The temporal core combines a Kalman filter and a low-pass filter, which learns the motion model through a recurrent neural network. The spatial core takes advantage of the biomechanical constraints of the human body to provide spatial coherency between keypoints. The combination of the cores allows the filter to adequately address different types of noise, from jittering to dropped frames. We test the filter on motion data from multiple datasets and seven 3D human pose estimation backbones, improving accuracy up to 140 mm with non-Gaussian noise and 53 mm with missing information.

1. Introduction

Human pose estimation (HPE) has mainly been investigated in the last years, and significant advancements have been achieved thanks to different and sophisticated deep learning techniques [1,2]. Because HPE allows for *marker-less* human motion analysis, there is a growing interest to adopt such software platforms also to the healthcare [3–5] as well as industrial domain [6–8].

Nevertheless, the intrinsic inaccuracy of such platforms often leads to noise in the extrapolated positional information of human keypoints or periods during which such information is even missing. As a consequence, data filtering for denoising or completion is a fundamental step before data analysis. Temporal filters based on neural networks and state observers have shown great potential to denoise HPE. However, using these techniques in real-world situations is challenging, especially when the goal is to generate filtering or reconstruction results with minimal delay. We address this by proposing a real-time spatiotemporal filter that uses inter-frame (temporal) and intra-frame (spatial) information to remove jitters, correct wrong 3D poses and complement dropped frames. Our *filter with learned kinematics* (*FLK*) combines an adaptive Kalman filter, a low-pass filter, and a filter that implements skeleton adjustments based on biomechanical constraints. *FLK* selectively activates the Kalman filter by using the concept of *keypoint confidence*, which allows identifying the keypoints more subject to jittering (see Fig. 1).

Kalman filters (KF) [9] have shown great potential and broad applicability for filtering [10,11]. Some variants (e.g., EKF, UKF) extend the applicability of KF to nonlinear systems by linearization or deterministic sampling [12]. They use dynamic system input/output measurements observed over time, statistical noise, and other inaccuracies to estimate a system's unknown inner state. KF consists of two phases: prediction and correction. In the prediction phase, the filter estimates the system's current state based only on the previous observation and the transition function. The correction phase adjusts the estimated state closer to the measured state by steering the estimated state toward the measured values. Several approaches tried to use KF to denoise the outcome of HPE frameworks. Niu et al. [7] used KF to reduce jitter and other inaccuracies caused by occlusions, proposing a reliability index to identify errors in joint detection. Loumponias et al. [13] applied a non-linear KF to recover sequences from a markerless mocap system, where occluded keypoints were considered occluded. Tripathy et al. [14] constrained a KF to maintain the consistent distance between physically linked joints throughout the motion sequence captured by an RGB-D camera. Musunuri et al. [15] compared KFs in reducing nonlinear temporal noise in RGB-D motion capture data. Similarly, Ahmed et al. [16] filtered the RGB-D camera motion capture output to

* Corresponding author.
E-mail address: enrico.martini@univr.it (E. Martini).

https://doi.org/10.1016/j.sigpro.2024.109598
Received 23 October 2023; Received in revised form 3 April 2024; Accepted 28 June 2024
Available online 4 July 2024

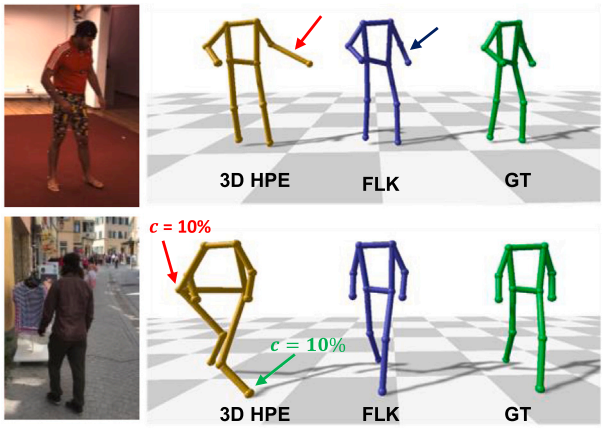

Fig. 1. Pose estimation results: extrapolated by a 3D HPE framework (3D HPE), with emphasis on confidence values of keypoints (c) and spatial incoherence between keypoints w.r.t. human biomechanics; After filtering with the proposed approach (*FLK*); The ground truth (GT).

avoid inconsistent estimations of the limb lengths. The main limitation of these approaches is the choice of the transition function. Since human motion is complex and often unpredictable, the exact knowledge about the transition function cannot be determined, leading to wrong predictions.

Another set of approaches is based on *autoencoder*, which has become popular in recent years due to its ability to learn a compressed representation of the input data [17,18]. To refine 3D skeleton motion, Holden et al. [19] proposed a convolutional autoencoder that projects the motion onto the manifold to remove the error. Bidirectional recurrent autoencoders (BRA) account for temporal relationships and dynamics of human motion [20]. In [21], they proposed a bidirectional recurrent auto-encoder that preserves bone-length consistency by maintaining the distance between keypoints naturally connected by bone. Nakatsuka et al. [22] proposed a variational autoencoder with gated recurrent units (GRU) to solve the problems of left–right switching and missing markers. Zeng et al. [23] proposed *SmoothNet*, a Deep Learning network that is HPE-agnostic. They demonstrated its validity in improving temporal smoothness and precision on 2D HPE, 3D HPE, and SMPL [24]. In the literature, there are also combinations of different approaches, as reported in [25]. Coskun et al. [26] integrated three LSTMs into a KF to capture the high non-linearity that characterizes both human motion and noise models. The primary limitation of learned models lies in their limited ability to generalize. Retraining those models is essential to achieve optimal results with different HPE frameworks, necessitating additional datasets.

In the proposed *adaptive Kalman filter* (AKF) block, the transition function is learned through a recurrent neural network (RNN), providing accurate predictions of the body joints' position. Also, limiting the use of neural networks to the prediction phase makes the filter independent of the training dataset and the HPE backbone. We designed and compared different RNN models and showed that gated recurrent units (GRU) allow the Kalman filter to achieve better results than the long short-term memory (LSTM). We combined AKF with a low-pass filter to adequately address high-frequency *large-error* jitters (with the first) and high-frequency *small-error* jitters (with the second). We also implemented a spatial filter to take advantage of the biomechanical constraints of the human body to guarantee spatial coherency between keypoints.

The main contributions of the work are the following:

- We present *Filter with Learned Kinematics (FLK)*, a three-step denoising and completion algorithm that refines keypoints from any 3D HPE frameworks. It consists of three blocks. The first is an Adaptive Kalman Filter (AKF), which relies on a transition function learned through RNN and overcomes the limitation of traditional Kalman filters by dynamically adjusting only keypoints more prone to jittering. The second, Biomechanical Constraint Adjustment (BCA), is a spatial filter that models the keypoints as a graph and ensures spatial coherency between keypoints. The third is a low-pass filter (LPF) that eliminates small-error jitters caused by temporal incoherencies between frames.
- We propose a quantitative evaluation of *FLK* applied to refine the 3D human poses extrapolated by many state-of-the-art HPE on three popular datasets. In the analysis, we included further and different types of pose errors (i.e., Gaussian and random noise), missing keypoints, and dropped frames.
- We demonstrate the efficiency of the learned transition function using a Gated Recurrent Unit (GRU) through an ablation study, compared to Long-Short Term Memory (LSTM) and the identity function.

The code is publicly available at https://github.com/PARCO-LAB/FLK.

2. Methodology

FLK is a plug-and-play filter that refines the human pose estimated by any 3D HPE framework. The starting point is the set of keypoints (i.e., the skeleton $\hat{s}_k$) representing the 3D coordinates of the human joints extrapolated from an image at frame k. The result is the denoised and completed version of the skeleton, $\hat{s}''_k$. Fig. 2 shows an overview of the filtering pipeline, which blocks are described in more detail in the following sections.

2.1. Adaptive Kalman filter

AKF is the first filtering block applied to correct the spatial position of keypoints estimated by the HPE with high-frequency and *large-error* jitter. It takes advantage of the pose information of the previous frame to identify the target keypoints.

For each keypoint j at frame k, AKF computes a confidence value c as in Eq. (1):

$$c_{k,j} = \frac{1}{\alpha v_{k,j}^2 + 1} \qquad v_{k,j} = \frac{||p_{k,j} - \hat{p}_{k-1,j}||}{t_k - t_{k-1}} \tag{1}$$

where α is a fixed scaling factor, v represents the velocity of the keypoint at frame k, $p_{k,j}$ is the 3D position of the jth keypoint at frame k, $\hat{p}_{k-1,j}$ is the refined position of the jth keypoint at frame $k-1$, and t_k is the timestamp of the k-frame, expressed in seconds. The confidence of each keypoints depends on its velocity, so that slow keypoints (i.e., keypoints with position at time t_k close to their position at time t_{k-1}) have higher confidence values than fast keypoints.

In the proposed solution, *FLK* applies the AKF block to each keypoint with a confidence value below a certain threshold, which we call *Kalman activation threshold* (Θ).

AKF implements a prediction-update approach to correct the keypoint spatial position when activated. Since the 3D position of each body joint is composed of three values, the inner state vector x, as the observation measure y, represents a single coordinate of a keypoint.

Differently from standard KF, in which the covariance of the process noise Q and the covariance of the observation noise R are fixed and found through manual tuning, in the proposed solution Q and R are calculated at runtime by considering the current velocity of keypoints, as in Eq. (2):

$$Q_{k,j} = (\alpha - 1)e^{-\alpha v_{k,j}} + 1 \qquad R_{k,j} = \alpha v_{k,j}^2 \tag{2}$$

Eq. (2) describes the relation between $Q_{k,j}$ and $R_{k,j}$ with the velocity v of keypoint j at frame k, so that the covariance of the noise increases differently along with the velocity.

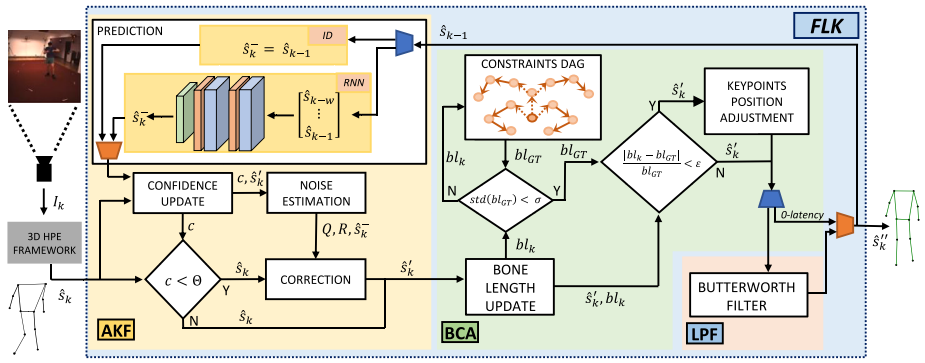

Fig. 2. Overview of the *FLK* filtering pipeline, which consists of an adaptive Kalman filter (AKF), a biomechanical constraint adjustment (BCA), and a low-pass filter (LPF). The input is $\hat{s}_k$, provided by a 3D HPE framework. The output is the refined version of the estimated skeleton, $\hat{s}''_k$.

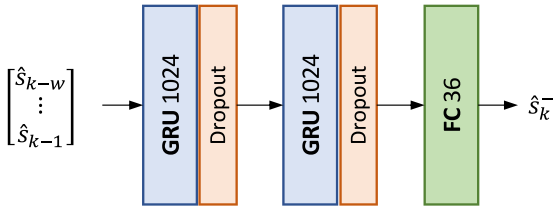

Fig. 3. RNN architecture.

Prediction. The prediction phase estimates the keypoint position based only on the motion model without knowing the actual position provided by the HPE framework. The critical challenge is to find the transition function that best approximates the evolution of the system state. We tested two models: *identity* and *RNN*. With the identity model, we assume that the human body is static. Following this assumption, we modeled the transition function as in Eq. (3):

$$\hat{x}^-_{k,j,a} = \hat{x}_{k-1,j,a} \tag{3}$$

where $\hat{x}^-_{k,j,a}$ is the prediction of the a-coordinate ($a \in \{x, y, z\}$) of joint j at frame k.

To deal with the complexity and non-linearity of human motion patterns, we implemented a second transition function that consists of an RNN with gated recurrent units (GRU), represented in Fig. 3. It predicts each current pose starting from a sequence of past poses. After each GRU layer, we applied a dropout layer with a keep probability of 0.9 to prevent overfitting during the training phase. We included a fully connected (FC) layer at the network's end. Since the prediction involves 12 keypoints, the model tested in experimental results uses an FC layer of 36 units.

Correction. In this phase, AKF uses the results provided by the HPE framework to refine the state estimated by the prediction phase. To do that, *FLK* compute the correction as in Eq. (4):

$$\hat{x}_{k,j,a} = K_{k,j} \cdot y_{k,j,a} + (1 - K_{k,j}) \cdot \hat{x}^-_{k,j,a} \tag{4}$$

where $K_{k,j}$ is the Kalman gain. The coordinate position $\hat{x}_{k,j,a}$ is obtained by averaging the prediction $\hat{x}^-_{k,j,a}$ (given by the identity or RNN model) with the current noisy position $y_{k,j,a}$. This average is weighted by $K_{k,j}$, so a higher gain weights more incoming measurements, while a small gain increases the previous state prediction weight (Fig. 4).

Considering, for example, $\Theta = 75\%$ and $\alpha = 10^{-2}$, we can derive from Eq. (1) that the minimum joint velocity to enable the Kalman filter is $v = 5.77$ m/s. At this velocity, the resulting refined position of each joint coordinate is composed of the current measurement for 35.6% and the prediction for 64.4%.

2.2. Biomechanical constraint adjustment

The biomechanical constraint adjustment (BCA) is the second block of the pipeline. It provides spatial coherency between keypoints that are biomechanically constrained. BCA uses a directed acyclic graph (DAG) to represent the human body, where nodes are the joint positions and edges are links between two adjacent joints. Each edge can be either flexible, such as the one between shoulder and hip, or fixed, such as the one between shoulder and elbow.

BCA requires each bone length to minimize the variation in rigid segments during tracking. During the first iterations, when the size of the bones is not yet known, the algorithm calculates the Euclidean distances between all the adjacent keypoints provided by AKF and stores such values in the DAG.

After a few iterations, when all measured distances converge on the corresponding consistent average value (i.e., the standard deviation is below a certain threshold σ), BCA starts to correct the joint positions. Given a measured bone length $\mathrm{bl}_{k,ij}$ (between keypoints i and j) and the corresponding reference bone length $\mathrm{bl}_{GT,ij}$, BCA activates only if $\mathrm{bl}_{k,ij}$ differs from $\mathrm{bl}_{GT,ij}$ by a constant threshold ϵ. When activated, it adjusts keypoint j as in Eq. (5):

$$\hat{p}_{k,j} = p_{k,i} + \frac{\mathrm{bl}_{GT,ij} \cdot (p_{k,j} - p_{k,i})}{\mathrm{bl}_{k,ij}} \tag{5}$$

where i and j are ordered by following the DAG representation. Eq. (5) corrects j such that the distance between i and j matches the reference bone length $\mathrm{bl}_{GT,ij}$.

2.3. Low-pass filter

FLK implements a low-pass filter (LPF) to correct high-frequency *low-error* jitters. In particular, we integrated a Butterworth filter [27]. The Butterworth filter, widely utilized in signal processing, is known for its simplicity and efficacy. With a maximally flat frequency response in the passband, the Butterworth filter ensures shape preservation without introducing distortions. Furthermore, it provides a smooth roll-off from the passband, contributing to a consistently predictable response. The chosen implementation of the Butterworth filter can be found in [28].

Overall, the LPF block complements the AKF filter by ensuring the quality and reliability of the skeleton data by eliminating noise and inconsistencies in the temporal dimension. *FLK* deactivates this block to provide zero-latency results.

3. Experimental evaluation

Datasets. We run the *FLK* evaluation on different datasets, each containing 3D pose annotations assumed as ground truth. *Human3.6M* [29] is one of the most extensive and widely adopted benchmarks for HPE. It comprises 3.6 million frames obtained from four digital cameras

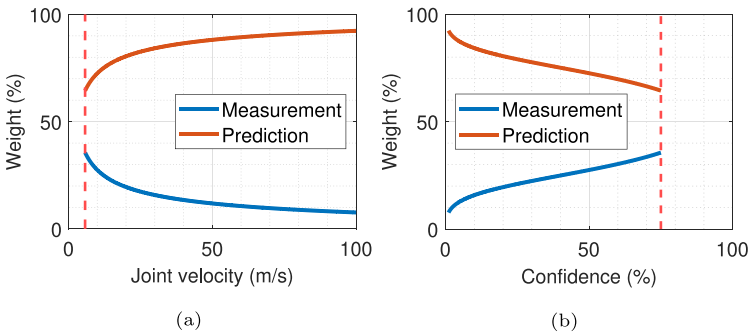

Fig. 4. Weight of measurement and prediction in the average to obtain the coordinate estimation. The red dotted line represents the *Kalman activation threshold* Θ, set to 75%. As the joint velocity increases (and the confidence drops), the adaptive filter weighs the model prediction more than the actual measurement.

Table 1
Experimental results on Human3.6M using different HPE frameworks as backbone. Each row corresponds to a refinement method. Each pair of columns represents a different input. The error is expressed as MPJPE and Accel.

Input	Synthetic (25%)		GT + FCN		OP + FCN		GT + Ray3D		CPN + Ray3D	
Evaluation metrics	MPJPE ↓	Acc. ↓	MPJPE ↓	Acc. ↓	MPJPE ↓	Acc. ↓	MPJPE ↓	Acc. ↓	MPJPE ↓	Acc. ↓
3D HPE framework	183.2	551.2	49.2	1.3	78.5	19.0	84.9	2.2	114.6	15.8
+ EMA	105.2	122.7	61.6	**0.9**	88.4	3.0	94.8	1.0	123.1	2.6
+ Linear KF (0th-order)	105.2	122.8	61.6	**0.9**	88.4	3.0	94.8	1.0	123.1	2.6
+ Linear KF (1st-order)	107.4	126.8	56.2	**0.9**	82.6	3.2	91.0	1.1	120.8	2.7
+ Linear KF (2nd-order)	110.0	129.8	54.8	**0.9**	80.8	3.3	90.4	1.1	119.6	2.8
+ SmoothNet [23] ($T = 64$)	78.2	78.8	49.8	**0.9**	76.7	1.2	85.1	**0.9**	113.7	1.1
+ **FLK (id)**	44.2	**75.7**	**47.8**	**0.9**	76.4	**1.0**	**84.4**	**0.9**	**113.6**	**1.0**
+ **FLK (GRU)**	**41.0**	**75.7**	**47.8**	**0.9**	**76.2**	**1.0**	**84.4**	**0.9**	**113.6**	**1.0**

with a 50 Hz frame rate. In our experiments, we trained the models included in the comparison on all actions performed by subjects 1, 5, 6, 7, and 8, while we used the first video of each action performed by subjects 9 and 11 as the test set. *AIST Dance Video Database* [30] is a dataset containing 10 389 video clips of different dancers. We trained on 868 sequences from the first 20 actors during our experiments and reported results using 470 sequences from the last ten actors, according to the dataset authors' recommendations. The *3D People in the Wild (3DPW)* dataset [31] is a collection of images captured with a moving phone camera in outdoor real-world environments. It includes diverse scenes, clothing, and poses, making it a challenging benchmark for HPE software. For the train/test division, we followed the protocol suggested by the authors.

Evaluation metrics. To evaluate the accuracy of *FLK*, we adopted the mean per joint position error (MPJPE) expressed in mm. We used the mean per joint acceleration error (Acc.) expressed in mm/s^2 to quantify the jitter problem, as in [23,32,33].

FLK parameters. We set all the parameters of *FLK* and kept the same values throughout the experimental results. In the AKF block, we chose $\Theta = 75\%$ and $\alpha = 10^{-2}$ after an offline grid search. For the prediction phase, we trained the RNN to predict the common keypoint position among the evaluated backbones: shoulders, elbows, wrists, hips, knees, and ankles. We trained the network by using the Adaptive Moment Estimation (Adam) [34] optimizer, with a learning rate of 10^{-5}. Since the network predicts the actual joint positions given a sequence of past skeletons without noise, we used the ground truth of different datasets for training. We empirically designed a 64-frame-long sequence as it provides a good trade-off between accuracy and processing time.

In the BCA block, $\sigma = 2$ cm and $\epsilon = 2 \cdot 10^{-2}$. In the LPF, the $3rd$-order Butterworth filter has a critical frequency of 2 Hz. This last block has been deactivated to generate the results at zero latency (Table 3).

Compared methods. To assess the performance of the proposed filter, we performed an extensive comparison using different state-of-the-art filtering techniques. For the traditional filters, we implemented a moving average filter, weighted moving average filter, Savitzky–Golay filter [35], and Butterworth low-pass filter. For the state-observers, we tested three variants of linear KF with different transition matrices. During the prediction phase, the 0th-order KF assumes that all keypoints do not move. The $1st$-order KF assumes that all keypoints keep the velocity constant, while the $2nd$-order KF assumes that all keypoints keep the acceleration constant. For each filter used in the comparison, we fine-tuned all fixed parameters to achieve the best results through grid search on the synthetic error dataset. Finally, we compared *FLK* with SmoothNet [23], which outperforms all the previous denoising solutions. We tested SmoothNet by using the pre-trained models provided by the authors. For this reason, Table 1 reports four variants of SmoothNet, which differ on the window length required to clean the data. We tuned each filtering method to obtain the best results via grid search. For all filters, we set the maximum permitted latency to 64 frames (1.28 s), which is the latency of the most accurate version of SmoothNet.

3.1. Comparison with existing solutions

Synthetic noise. The first test consists of a version of the ground truth corrupted by two noise types: additive white Gaussian noise (with a signal-to-noise ratio set to 45) and asymmetric noise. The second is an additive white Gaussian noise of power $8 \cdot 10^{-2}$ dBW with values below zero being halved. While the first noise is applied to each dataset value, the second affects only a pre-determined percentage (i.e., 5% and 25%). The first two columns on Table 1, i.e., *Synthetic (5%)* and *Synthetic (25%)*, show the accuracy of the filters on this type of noise. Both *FLK* versions outperform by far all the existing filtering methods, improving the quality of corrupted input in both precision (up to 77.6%) and smoothness (up to 87.7%).

Root-relative 3D human pose estimation. For the second test, we considered the task of improving the root-relative 3D HPE, in which the position and orientation of the human body are relative to the pelvis joint. We selected the fully-connected network (FCN) presented in [36] as relative 3D HPE since all SmoothNet models we used are trained and tested on FCN. In Table 1, columns *GT + FCN* and *OP + FCN* report the filtering results on FCN using the ground truth and OpenPose [37] as

Table 2
Experimental results on 3DPW and AIST using different HSE frameworks as backbone. Each row corresponds to a refinement method. Each pair of columns represents a different input. The error is expressed as MPJPE and Accel.

Input	SPIN (3DPW)		PARE (3DPW)		EFT (3DPW)		VIBE (AIST)		TCMR (AIST)	
Evaluation metrics	MPJPE ↓	Acc. ↓	MPJPE ↓	Acc. ↓	MPJPE ↓	Acc. ↓	MPJPE ↓	Acc. ↓	MPJPE ↓	Acc. ↓
3D HPE framework	101.8	37.2	80.0	27.9	92.6	35.3	104.4	31.5	104.5	6.6
+ EMA	105.2	9.0	86.2	7.9	96.9	8.7	114.0	6.8	113.7	4.6
+ Linear KF (0th-order)	105.2	9.0	86.2	8.0	96.9	8.7	113.9	6.8	113.7	4.6
+ Linear KF (1st-order)	104.8	9.4	85.6	8.2	96.7	9.1	114.0	7.2	114.1	4.7
+ Linear KF (2nd-order)	105.0	9.7	85.6	8.5	96.9	9.4	114.3	7.6	114.6	4.7
+ SmoothNet [23] ($T = 32$)	102.4	7.0	81.5	6.8	93.0	6.9	123.9	**4.4**	130.0	4.4
+ **FLK (id)**	99.3	**6.7**	**77.2**	**6.5**	**87.0**	**6.6**	**101.7**	4.5	**103.5**	**4.3**
+ **FLK (GRU)**	**99.2**	**6.7**	**77.2**	**6.5**	87.1	**6.6**	105.9	4.5	104.1	**4.3**

Table 3
Experimental results with zero-latency filters applied to the output of Ray3D with 2D CPN [38] on Human3.6M. Each row corresponds to a refinement method. Each column represents an action taken from the Human3.6M dataset [29]. The error is expressed as MPJPE.

No frame missing	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg.
Ray3D [39] (2D CPN [38])	161.4	84.8	141.7	150.4	119.2	113.2	82.0	73.9	161.5	**140.7**	121.1	91.8	91.5	93.2	92.2	114.6
+ Linear KF (0th-order)	163.5	89.1	150.9	153.7	126.1	117.8	86.1	81.4	162.1	142.1	126.1	96.2	106.3	128.2	116.8	123.1
+ Linear KF (1st-order)	163.8	88.8	147.4	153.4	123.0	116.5	85.5	79.3	162.5	141.9	124.6	95.0	102.8	117.6	109.7	120.8
+ Linear KF (2nd-order)	163.9	88.8	145.8	153.8	121.8	116.5	85.4	78.9	162.7	142.3	123.8	94.6	101.4	109.5	104.3	119.6
+ SmoothNet [23]	161.6	85.4	142.0	**150.2**	119.3	113.7	82.7	74.8	**161.3**	140.8	121.1	92.3	92.4	94.6	93.0	115.0
+ **FLK (id)**	**161.0**	**84.6**	**141.6**	**150.2**	**118.9**	**112.8**	**81.7**	**73.7**	161.6	141.2	**120.4**	**91.4**	91.6	91.9	91.2	114.3
+ **FLK (GRU)**	**161.0**	**84.6**	**141.6**	150.3	**118.9**	**112.8**	81.8	**73.7**	161.7	141.3	120.5	**91.4**	**91.4**	**91.8**	**91.1**	**114.2**
30% of frames missing, at least 10 consecutive frames lost per gap.																
Ray3D [39] (2D CPN [38])	163.1	**88.9**	148.4	155.7	124.9	117.1	**85.8**	78.6	163.1	**141.7**	125.8	97.0	103.0	117.4	114.1	121.6
+ Linear KF (0th-order)	166.0	94.5	161.0	161.2	134.0	124.4	91.8	88.3	164.2	144.0	133.0	104.0	123.1	163.5	148.7	133.4
+ Linear KF (1st-order)	166.6	94.6	155.2	160.3	128.5	122.2	91.4	85.7	164.8	144.1	129.8	101.6	115.7	143.0	133.4	129.1
+ Linear KF (2nd-order)	167.4	95.6	153.0	161.3	127.1	123.2	92.1	85.7	165.4	145.4	129.0	101.5	115.0	131.0	124.7	127.8
+ SmoothNet [23]	163.4	89.7	149.8	155.9	125.5	117.6	86.9	79.4	**163.0**	141.8	126.3	97.9	105.2	122.4	117.9	122.8
+ **FLK (id)**	163.0	89.0	149.9	156.0	125.3	**117.0**	85.9	78.7	163.2	142.3	125.8	97.6	107.6	126.4	120.6	123.2
+ **FLK (GRU)**	**162.7**	89.4	**143.9**	**153.6**	**121.4**	117.1	86.6	**78.0**	163.3	143.6	**123.0**	**95.8**	**98.8**	**97.0**	**96.0**	**118.0**
30% of frames missing, at least 20 consecutive frames lost per gap																
Ray3D [39] (2D CPN [38])	**166.2**	**95.6**	159.6	165.4	135.4	**128.1**	95.6	**90.1**	**166.0**	**148.9**	136.6	**104.6**	125.3	167.6	145.6	135.4
+ Linear KF (0th-order)	168.8	101.4	172.2	173.2	145.3	136.0	101.4	99.7	167.1	151.5	145.0	111.7	146.6	215.6	183.3	147.9
+ Linear KF (1st-order)	171.0	104.5	163.2	171.8	139.3	134.8	103.1	97.7	168.4	152.1	141.7	112.1	141.0	188.6	162.1	143.4
+ Linear KF (2nd-order)	174.2	108.9	161.1	176.6	139.5	138.4	107.0	102.5	170.7	155.2	143.2	114.4	145.0	174.4	151.0	144.1
+ SmoothNet [23]	166.6	96.6	161.5	166.1	136.9	129.2	97.4	91.7	**166.0**	149.4	137.7	106.3	129.5	176.4	152.4	137.6
+ **FLK (id)**	166.4	95.9	163.0	166.9	136.4	129.5	**95.4**	91.1	166.4	150.4	137.6	105.6	134.8	193.4	160.4	139.6
+ **FLK (GRU)**	168.8	101.4	**149.2**	**162.3**	**128.7**	132.4	103.6	95.8	168.5	155.1	**132.3**	107.6	**119.1**	**114.0**	**107.3**	**129.8**

2D pose estimator, respectively. Also, in these cases, both *FLK* versions outperform SmoothNet and all the other filtering techniques, reducing the FCN error, on average, by 2.8% and 2.9%, respectively.

Absolute 3D human pose estimation. We evaluated the performance of *FLK* on improving ray-based 3d human pose estimation (Ray3D) [39], which is the state-of-the-art solution for *absolute* 3D HPE. Differently from FCN, Ray3D provides the skeletons in the World Coordinate System. As a two-stage approach, Ray3D needs the 2D poses as input. Following the authors' recommendations, we trained Ray3D using the 2D ground truth and Cascaded Pyramid Network (CPN) [38]. Columns *GT + Ray3D* and *CPN + Ray3D* on Table 1 show that *FLK* improves Ray3D, in average, of 0.5 mm and 1.0 mm, respectively.

Human shape estimation. We also tested FLK in denoising 3D keypoints from Human Shape Estimation (HSE) backbones. HSE focuses on predicting human 3D shapes and poses from 2D images, providing information about the skeletal structure, articulation, and surface geometry as output. The standard representation for this task is the *Skinned Multi-Person Linear* model (SMPL) [24]. It includes the root-relative 3D positions of a predefined set of anatomical joint locations. For the experiments presented in Table 2, we used 5 different HSE backbones: SPIN [1], PARE [40], EFT [41], VIBE [32], TCMR [42]. FLK performs better than the other methods, yielding similar results across the various datasets. However, it is worth noting that in the 3DPW dataset, both the GRU and identity versions perform similarly. In contrast, when testing on the AIST dataset, the identity version outperforms the GRU version. This is due to the high-velocity movements within the AIST dataset, presenting challenges for learning the movement patterns by the RNN.

3.2. Comparison of 0-latency filters

To simulate a possible application of the denoising solutions on a real-time system, in Table 3, we consider only 0-latency filters. A filter is 0-latency if, given a skeleton $\hat{s}_k$ at time k, it provides the filtered $\hat{s}'_k$ before $\hat{s}_{k+1}$ arrives. SmoothNet, in the original implementation, is not a 0-latency filter, as when skeleton $\hat{s}_k$ arrives, the filter output is the refined skeleton $\hat{s}'_{k-T}$. According to the authors' suggestions, we implemented a 0-latency version of SmoothNet by considering only the last frame in the denoising process. For the sake of space, we present the results obtained starting from the outputs of Ray3D with CPN. The first section of Table 3 reports all the actions of the dataset. On average, *FLK* with GRU achieves the best results, being the only filter that can smooth Ray3D without introducing latency.

Gap filling. In many scenarios, occlusions and corrupted frames often result in total or partial absence of information. To test the robustness of *FLK* in such contexts, we randomly removed consecutive frames from the Human3.6M videos. We discarded 30 percent of the frames in the first test with bursts of at least ten successive frames (≥0.1 s) and, in the second test, with bursts of at least 20 consecutive frames (≥0.4 s). While filters based on KF are designed to handle periods with no information, SmoothNet cannot deal with missing data. To compare the solutions, we replaced each missing frame with a copy of the last available frame for SmoothNet.

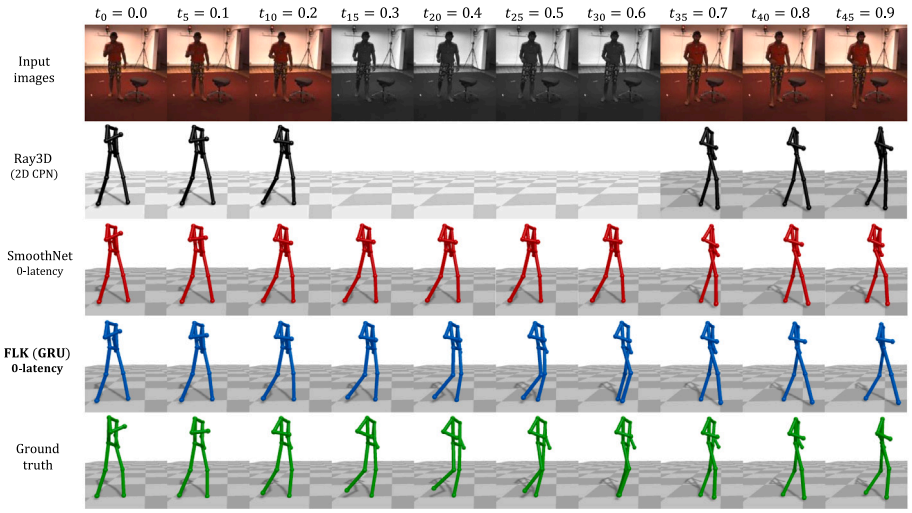

Fig. 5. Qualitative example of gap filling with the 0-latency constraint. Skeletons in black are estimated by Ray3D [39], with CPN [38] as 2D HPE. For 20 consecutive frames (from t_{11} to t_{30}) no skeleton is given to the filters. The 0-latency version of SmoothNet [23] (in red) performs poorly since it does not handle the case of missing data, while *FLK* (in blue) produces more reliable results, compared to ground truth (in green).

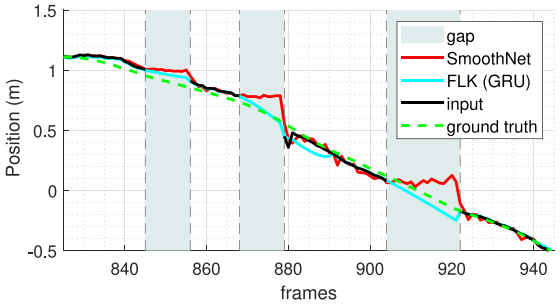

Fig. 6. Displacement of a keypoint coordinate in the presence of gaps. The input (in black) is the result of Ray3D using CPN. Both SmoothNet (in red) and *FLK* (in cyan) are in their best 0-latency versions.

The last two sections of Table 3 show the results of our analysis in the presence of missing frames. *FLK* with GRU is the only filter that can improve Ray3D on average. In particular, the improvement of *FLK* is more evident when applied in non-static scenarios (i.e., where the majority of keypoints have non-zero velocity). Examples of clear improvements are on *WalkD* (up to 4.9%), *Walk* (up to 31.9%), and *WalkT* (up to 26.3%). Fig. 5 shows qualitatively the effectiveness of *FLK* in recovering missing information while the subject is walking.

Fig. 6 shows how *FLK* and SmoothNet fill the gaps at the coordinate level. While SmoothNet tends to maintain the current position, *FLK* understands the motion pattern and consequently follows more closely the correct trajectory (green dotted line).

3.3. Ablation study

We implemented a transition function based on LSTM [26] and compared the results achieved with our function based on GRU. The network implements three stacked LSTM layers, each one with 1024 units. A dropout with a keep probability of 0.7 follows all layers. The network's end consists of two fully connected layers of 1024 units with a *relu* activation function and a final layer with 36 units for the coordinate regression (i.e., one unit per coordinate of each keypoint). The optimizator, learning rate, and initialization are the same as in the GRU model. The only difference is, according to [43], the forget biases, which were initialized to 1.

To test the advantages of the GRU, we compared the three configurations of FLK: id (i.e., no prediction), LSTM, and GRU. Table 4 shows that, as the gaps increase, the GRU version outperforms the LSTM and id versions. For example, in the last section of the table, where there are 30% of missing frames with at least 20 consecutive lost frames per gap, the GRU version achieves better results (21 mm improvement on average). This version is the only one that, in this test, enhances the accuracy of Ray3D.

3.4. Performance analysis

The primary purpose of the filter is to operate in real-time; that is, the total refinement time required by the *FLK* pipeline has to be less than 20 ms (i.e., >50 Hz). We performed all our experiments on a desktop PC with a CPU Intel i7-7700K, a GPU Nvidia RTX 2070 Super, and 16 GB RAM. The version of *FLK* with GRU requires a training time of 653 s and an average computation time of 14.4 ms. On the other hand, the identity version of *FLK* has an average computation time of 1.2 ms. Regarding the 0-latency versions, SmoothNet has a computation time comparable to *FLK* with GRU, while Kalman filters have a computation time similar to the identity version of *FLK*. The computation burden of the other approaches entirely depends on the smoothing window (e.g., the SMA filter with 32-frame windows has a latency of 320 ms).

4. Future work

We plan to refine the biomechanical model to better capture the synergies of human motion by incorporating learned spatial information into the filtering process. In addition, exploring techniques to reduce computational overhead and improve efficiency may be useful, particularly for meeting the real-time requirements of embedded safety-critical systems. We also plan to incorporate a tracking module into the filtering pipeline, designing a *FLK* version that can effectively refine multi-person 3D HPE frameworks.

5. Conclusion

In this work, we proposed *FLK*, a spatio-temporal filter to refine 3D human motion in real-time and at low latency. The filter uses a motion model learned by an RNN and human anatomical constraints to provide a denoised estimation. We showed that *FLK* outperforms other existing solutions in refining the position of different 3D HPE frameworks. We also proposed a 0-latency version of *FLK* that not only improves the current state-of-the-art method for absolute 3D HPE but is also robust to periods on which input frames are unavailable.

Table 4
Results of *FLK* at 0 latency with different transition functions on the results provided by Ray3D [39] with 2D CPN [38]. Each row corresponds to FLK with a specific transition function. Each column represents an action taken from the Human3.6M dataset [29]. The error is expressed as MPJPE.

No frame missing	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg.
Ray3D [39] (2D CPN [38])	161.4	84.8	141.7	150.4	119.2	113.2	82.0	73.9	**161.5**	**140.7**	121.1	91.8	91.5	93.2	92.2	114.6
+ FLK (id)	**161.0**	**84.6**	**141.6**	**150.2**	**118.9**	**112.8**	**81.7**	**73.7**	161.6	141.2	**120.4**	**91.4**	91.6	91.9	91.2	114.3
+ FLK (lstm)	161.1	**84.6**	**141.6**	150.3	**118.9**	113.1	**81.7**	**73.7**	161.8	141.5	121.7	**91.4**	**91.4**	**91.6**	**90.9**	114.4
+ **FLK (GRU)**	**161.0**	**84.6**	**141.6**	150.3	**118.9**	**112.8**	81.8	**73.7**	161.7	141.3	120.5	**91.4**	**91.4**	91.8	91.1	**114.2**
30% of frames missing, at least 10 consecutive frames lost per gap.																
Ray3D [39] (2D CPN [38])	163.1	88.9	148.4	155.7	124.9	117.1	**85.8**	78.6	**163.1**	**141.7**	125.8	97.0	103.0	117.4	114.1	121.6
+ FLK (id)	163.0	**89.0**	149.9	156.0	125.3	**117.0**	85.9	78.7	163.2	142.3	125.8	97.6	107.6	126.4	120.6	123.2
+ FLK (lstm)	165.9	94.6	148.7	161.3	125.6	126.2	91.6	83.5	166.0	146.1	130.3	101.2	105.0	100.6	100.2	123.1
+ **FLK (GRU)**	**162.7**	89.4	**143.9**	**153.6**	**121.4**	117.1	86.6	**78.0**	163.3	143.6	**123.0**	**95.8**	**98.8**	**97.0**	**96.0**	**118.0**
30% of frames missing, at least 20 consecutive frames lost per gap																
Ray3D [39] (2D CPN [38])	166.2	95.6	159.6	165.4	135.4	**128.1**	95.6	**90.1**	**166.0**	**148.9**	136.6	**104.6**	125.3	167.6	145.6	135.4
+ FLK (id)	**166.4**	**95.9**	163.0	166.9	136.4	129.5	**95.4**	91.1	166.4	150.4	137.6	105.6	134.8	193.4	160.4	139.6
+ FLK (lstm)	177.0	124.8	165.3	199.6	145.7	159.6	116.6	106.6	180.1	160.6	154.3	123.3	134.8	168.7	128.4	149.7
+ **FLK (GRU)**	168.8	101.4	**149.2**	**162.3**	**128.7**	132.4	103.6	95.8	168.5	155.1	**132.3**	107.6	**119.1**	**114.0**	**107.3**	**129.8**

CRediT authorship contribution statement

Enrico Martini: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Michele Boldo:** Writing – original draft, Validation, Software, Methodology, Investigation, Data curation. **Nicola Bombieri:** Writing – review & editing, Writing – original draft, Supervision, Project administration.

Declaration of competing interest

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

Data availability

I have shared the code in the attachments.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.sigpro.2024.109598.

References

[1] N. Kolotouros, G. Pavlakos, M.J. Black, K. Daniilidis, Learning to reconstruct 3D human pose and shape via model-fitting in the loop, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 2252–2261, http://dx.doi.org/10.48550/arXiv.1909.12828.

[2] K. Sun, B. Xiao, D. Liu, J. Wang, Deep high-resolution representation learning for human pose estimation, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2019-June, 2019, pp. 5686–5696, http://dx.doi.org/10.1109/CVPR.2019.00584, cited By 1660.

[3] J.-M. Sa, K.-S. Choi, Humanoid robot teleoperation system using a fast vision-based pose estimation and refinement method, IEIE Trans. Smart Process. Comput. 10 (1) (2021) 24–30, http://dx.doi.org/10.5573/IEIESPC.2021.10.1.024.

[4] E. Martini, M. Boldo, S. Aldegheri, N. Valè, M. Filippetti, N. Smania, M. Bertucco, A. Picelli, N. Bombieri, Enabling gait analysis in the telemedicine practice through portable and accurate 3D human pose estimation, Comput. Methods Programs Biomed. 225 (2022) 107016, http://dx.doi.org/10.1016/j.cmpb.2022.107016.

[5] A. Pereira, G. Stillfried, T. Baker, A. Schmidt, A. Maier, B. Pleintinger, Z. Chen, T. Hulin, N. Lii, Reconstructing human hand pose and configuration using a fixed-base exoskeleton, in: Proceedings - IEEE International Conference on Robotics and Automation, Vol. 2019-May, 2019, pp. 3514–3520, http://dx.doi.org/10.1109/ICRA.2019.8794059.

[6] X. Deng, J. Liu, H. Gong, H. Gong, J. Huang, A human-robot collaboration method using a pose estimation network for robot learning of assembly manipulation trajectories from demonstration videos, IEEE Trans. Ind. Inform. (2022) 1–9, http://dx.doi.org/10.1109/TII.2022.3224966.

[7] J. Niu, X. Wang, D. Wang, L. Ran, A novel method of human joint prediction in an occlusion scene by using low-cost motion capture technique, Sensors (Switzerland) 20 (4) (2020) http://dx.doi.org/10.3390/s20041119.

[8] J. Fan, P. Zheng, S. Li, Vision-based holistic scene understanding towards proactive human–robot collaboration, Robot. Comput.-Integr. Manuf. 75 (2022) 102304, http://dx.doi.org/10.1016/j.rcim.2021.102304.

[9] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME J. Basic Eng. 82 (Series D) (1960) 35–45, http://dx.doi.org/10.1115/1.3662552.

[10] J. Shu, F. Hamano, J. Angus, Application of extended Kalman filter for improving the accuracy and smoothness of Kinect skeleton-joint estimates, J. Engrg. Math. 88 (1) (2014) 161–175, http://dx.doi.org/10.1007/s10665-014-9689-2.

[11] P. Agarwal, S. Kumar, J. Ryde, J.J. Corso, V.N. Krovi, Estimating human dynamics on-the-fly using monocular video for pose estimation, Robot. Sci. Syst. 8 (July) (2013) 1–8, http://dx.doi.org/10.15607/rss.2012.viii.001.

[12] E.A. Wan, R. Van Der Merwe, The unscented Kalman filter for nonlinear estimation, in: Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373), IEEE, 2000, pp. 153–158, http://dx.doi.org/10.1109/ASSPCC.2000.882463.

[13] K. Loumponias, N. Vretos, G. Tsaklidis, P. Daras, An improved tobit Kalman filter with adaptive censoring limits, Circuits Syst. Signal Process. 39 (11) (2020) 5588–5617, http://dx.doi.org/10.1007/s00034-020-01422-w, arXiv:1911.06190.

[14] S.R. Tripathy, K. Chakravarty, A. Sinha, D. Chatterjee, S.K. Saha, Constrained Kalman filter for improving kinect based measurements, in: 2017 IEEE Int. Symp. Circuits Syst., Vol. 2018-Septe, IEEE, 2017, pp. 1–4, http://dx.doi.org/10.1109/ISCAS.2017.8050664.

[15] Y.R. Musunuri, O.S. Kwon, State estimation using a randomized unscented kalman filter for 3d skeleton posture, Electronics 10 (8) (2021) http://dx.doi.org/10.3390/electronics10080971.

[16] F. Ahmed, A.S.M. Hossain Bari, B. Sieu, J. Sadeghi, J. Scholten, M.L. Gavrilova, Kalman filter-based noise reduction framework for posture estimation using depth sensor, in: 2019 IEEE 18th International Conference on Cognitive Informatics & Cognitive Computing, ICCI*CC, 2019, pp. 150–158, http://dx.doi.org/10.1109/ICCICC46617.2019.9146069.

[17] X. Liu, Y.M. Cheung, S.J. Peng, Z. Cui, B. Zhong, J.X. Du, Automatic motion capture data denoising via filtered subspace clustering and low rank matrix approximation, Signal Process. 105 (2014) 350–362, http://dx.doi.org/10.1016/j.sigpro.2014.06.009.

[18] J. Xiao, Y. Feng, M. Ji, X. Yang, J.J. Zhang, Y. Zhuang, Sparse motion bases selection for human motion denoising, Signal Process. 110 (2015) 108–122, http://dx.doi.org/10.1016/j.sigpro.2014.08.017.

[19] D. Holden, J. Saito, T. Komura, T. Joyce, Learning motion manifolds with convolutional autoencoders, in: SIGGRAPH Asia 2015 Tech. Briefs, SA 2015, 2015, pp. 1–4, http://dx.doi.org/10.1145/2820903.2820918.

[20] U. Mall, G.R. Lal, S. Chaudhuri, P. Chaudhuri, A deep recurrent framework for cleaning motion capture data, 2017, http://dx.doi.org/10.48550/arXiv.1712.03380, arXiv preprint arXiv:1712.03380.

[21] S.J. Li, H.S. Zhu, L.P. Zheng, L. Li, A perceptual-based noise-agnostic 3D skeleton motion data refinement network, IEEE Access 8 (2020) 52927–52940, http://dx.doi.org/10.1109/ACCESS.2020.2980316.

[22] C. Nakatsuka, S. Komorita, Denoising 3D human poses from low-resolution video using variational autoencoder, IEEE Int. Conf. Intell. Robot. Syst. (2021) 4625–4630, http://dx.doi.org/10.1109/IROS51168.2021.9636144.

[23] A. Zeng, L. Yang, X. Ju, J. Li, J. Wang, Q. Xu, SmoothNet: A plug-and-play network for refining human poses in videos, in: European Conference on Computer Vision, Springer, 2022, http://dx.doi.org/10.48550/arXiv.2112.13715.

[24] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, M.J. Black, SMPL: A skinned multi-person linear model, ACM Trans. Graph. (Proc. SIGGRAPH Asia) 34 (6) (2015) 248:1–248:16, http://dx.doi.org/10.1145/2816795.2818013.

[25] E. Martini, A. Calanca, N. Bombieri, Denoising and completion filters for human motion software: a survey with code, 2023, http://dx.doi.org/10.36227/techrxiv.22956482.v1.

[26] H. Coskun, F. Achilles, R. Dipietro, N. Navab, F. Tombari, Long short-term memory Kalman filters: Recurrent neural estimators for pose regularization, in: Proc. IEEE Int. Conf. Comput. Vis., Vol. 2017-Octob, 2017, pp. 5525–5533, http://dx.doi.org/10.1109/ICCV.2017.589, arXiv:1708.01885.

[27] S. Butterworth, et al., On the theory of filter amplifiers, Wireless Eng. 7 (6) (1930) 536–541.

[28] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, İ. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental algorithms for scientific computing in python, Nat. Methods 17 (2020) 261–272, http://dx.doi.org/10.1038/s41592-019-0686-2.

[29] C. Ionescu, D. Papava, V. Olaru, C. Sminchisescu, Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments, IEEE Trans. Pattern Anal. Mach. Intell. 36 (7) (2014) 1325–1339, http://dx.doi.org/10.1109/TPAMI.2013.248.

[30] S. Tsuchida, S. Fukayama, M. Hamasaki, M. Goto, AIST dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing, in: Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, Netherlands, 2019, pp. 501–510.

[31] T. von Marcard, R. Henschel, M. Black, B. Rosenhahn, G. Pons-Moll, Recovering accurate 3D human pose in the wild using IMUs and a moving camera, in: European Conference on Computer Vision, ECCV, 2018, http://dx.doi.org/10.1007/978-3-030-01249-6_37.

[32] M. Kocabas, N. Athanasiou, M.J. Black, Vibe: Video inference for human body pose and shape estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 5253–5263, http://dx.doi.org/10.48550/arXiv.1912.05656.

[33] H. Choi, G. Moon, J.Y. Chang, K.M. Lee, Beyond static features for temporally consistent 3d human pose and shape from a video, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 1964–1973, http://dx.doi.org/10.48550/arXiv.2011.08627.

[34] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, http://dx.doi.org/10.48550/ARXIV.1412.6980.

[35] J. Luo, K. Ying, J. Bai, Savitzky–Golay smoothing and differentiation filter for even number data, Signal Process. 85 (7) (2005) 1429–1434, http://dx.doi.org/10.1016/j.sigpro.2005.02.002.

[36] J. Martinez, R. Hossain, J. Romero, J.J. Little, A simple yet effective baseline for 3d human pose estimation, in: ICCV, 2017, http://dx.doi.org/10.48550/arXiv.1705.03098.

[37] Z. Cao, T. Simon, S.-E. Wei, Y. Sheikh, Realtime multi-person 2d pose estimation using part affinity fields, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7291–7299, http://dx.doi.org/10.48550/arXiv.1611.08050.

[38] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, J. Sun, Cascaded pyramid network for multi-person pose estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7103–7112, http://dx.doi.org/10.48550/arXiv.1711.07319.

[39] Y. Zhan, F. Li, R. Weng, W. Choi, Ray3D: ray-based 3D human pose estimation for monocular absolute 3D localization, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022, pp. 13106–13115, http://dx.doi.org/10.1109/CVPR52688.2022.01277.

[40] M. Kocabas, C.-H.P. Huang, O. Hilliges, M.J. Black, PARE: Part attention regressor for 3D human body estimation, in: Proc. International Conference on Computer Vision, ICCV, 2021, pp. 11127–11137, http://dx.doi.org/10.48550/arXiv.2104.08527.

[41] H. Joo, N. Neverova, A. Vedaldi, Exemplar fine-tuning for 3D human model fitting towards in-the-wild 3D human pose estimation, 2020, http://dx.doi.org/10.48550/ARXIV.2004.03686.

[42] H. Choi, G. Moon, J.Y. Chang, K.M. Lee, Beyond static features for temporally consistent 3D human pose and shape from a video, in: Conference on Computer Vision and Pattern Recognition, CVPR, 2021, http://dx.doi.org/10.48550/arXiv.2011.08627.

[43] R. Jozefowicz, W. Zaremba, I. Sutskever, An empirical exploration of recurrent network architectures, in: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML '15, JMLR.org, 2015, pp. 2342–2350, http://dx.doi.org/10.5555/3045118.3045367.