Smoothing Skeleton Avatar Visualizations Using Signal Processing Technology

Joela F. Gauss¹ · Christoph Brandin² · Andreas Heberle¹ · Welf Löwe³

Received: 18 January 2021 / Accepted: 7 August 2021 / Published online: 25 August 2021
© The Author(s) 2021

Abstract

Movements of a person can be recorded with a mobile camera and visualized as sequences of stick figures for assessments in health and elderly care, physio-therapy, and sports. However, since the visualizations flicker due to noisy input data, the visualizations themselves and even whole assessment applications are not trusted in general. The present paper evaluates different filters for smoothing the movement visualizations but keeping their validity for a visual physio-therapeutic assessment. It evaluates variants of moving average, high-pass, and Kalman filters with different parameters. Moreover, it presents a framework for the quantitative evaluation of smoothness and validity. As these two criteria are contradicting, the framework also allows to weight them differently and to automatically find the correspondingly best-fitting filter and its parameters. Different filters can be recommended for different weightings of smoothness and validity. The evaluation framework is applicable in more general contexts and with more filters than the three filters assessed. However, as a practical result of this work, a suitable filter for stick figure visualizations in a mobile application for assessing movement quality could be selected and used in a mobile app. The application is now more trustworthy and used by medical and sports experts, and end customers alike.

Keywords Movement visualization · Smoothing filters · Filter evaluation framework

Introduction

Automated sensor-based assessment of human movements has many applications ranging from the diagnosis and therapy of musculo-skeletal insufficiency, to elderly care, and to high performance sport. In the present work, we contribute to camera-based human movement assessment. Expensive professional motion capture systems use several cameras and markers at key-points of subjects, e.g., the joints, in order to

accurately estimate their poses and trace their movements. Key-points can be connected to stick figures and used to visualize the poses and to animate the movements.

Alternatively to or in combination with camera-based systems, other sensors, such as accelerometers, gyroscopes, and magnetometers, integrated in wearable inertial measurement units (IMU) can be used to estimate key-points. For instance, Kumar et al. [21] recorded gait data simultaneously using motion sensors and visible-light cameras and fused the data to accurately classify different types of walks. Guzov et al. [14] used IMUs attached at the body limbs and a head mounted camera looking outwards, to fuse camera based self-localization with IMU-based human body tracking. These multi-modal approaches can reduce measurement errors and noise coming from each individual channel.

We refer to a system of camera(s) and other sensors and the software tracing the key-points of moving human subjects as Skeleton Avatar Technology (SAT).

To avoid costs, commodity SAT systems only use one camera and, optionally, an infrared depth sensor. The positions of key-points are then estimated using statistics and artificial intelligence (AI)-based software. Commodity 3D

✉ Welf Löwe
Welf.Lowe@lnu.se

Joela F. Gauss
jojoelag@gmx.de

Christoph Brandin
christoph.brandin@aimo-health.com

Andreas Heberle
andreas.heberle@hs-karlsruhe.de

¹ Karlsruhe University of Applied Sciences, Moltkestraße 30, Karlsruhe 76133, Germany

² AIMO GmbH, Quellenstraße 7a, Stuttgart 70376, Germany

³ Linnaeus University, Växjö, 351 95, Sweden

SAT systems including Microsoft's Kinect,¹ Orbbec's Astra Mini,² and Intel's Real Sense³ use a video camera and an infrared depth sensor. Commodity 2D SAT systems only use a regular video camera. All commodity SAT systems use deep learning-based software for estimating key-points.

While professional SAT systems are more accurate, commodity SAT systems make it possible nowadays to assess movements in uncontrolled environments, e.g., at home or in the wild. They also make it possible to collect larger volumes of data on the health status of a patient or on the performance of an athlete and the change in this status after exercises and treatments.

The inaccuracy of commodity SAT technology leads to two major problems. The first and foremost problem is the *validity* of the analysis results. With less accurate SAT pose data it is harder to guarantee analysis validity. However, validity has been achieved and documented in a number of applications, cf. the Related Work Section "[Related work](#)" for the details. The second problem is gaining *trust* in the analysis results. Experts are probably convinced by scientific validity studies. Pose visualizations and movement animations help convincing even the non-expert users.

The main goal of this work is not to contribute to the work that validates SAT in certain application contexts, but to be able to show SAT-based animations of movements with an adequate quality to increase trust. Therefore, the animations should properly reflect the actual movement. They should be smooth and suppress the flickering due to inaccurate SAT pose data. However, the smoothing must not transform the actual movement of the subjects into an ideal one.

Therefore, we employ and compare three classic filters that cancel or remove high-frequency noise components from the actual information signal: Moving Averages (Hyndman [18]), High-pass filters using the Fourier transform and its inverse (Van Loan [34]), and Kalman filters (Kalman [19]).

It is (relatively) easy for humans to visually assess the appropriateness and the (lack of) flickering in an animation, it is not trivial to do this automatically with an assessment algorithm. The latter, however, is necessary to set the parameters of the three approaches and to select a champion. We suggest an approach based on Principle Component Analysis (PCA) to check whether the main movement signal is maintained while the noise is filtered.

In summary, the paper contributes with

- (i) a framework of alternative filter components, post-processing SAT key-points data and creating valid and smooth movement animations,
- (ii) an automated assessment of validity and smoothness of the animations, and
- (iii) an application and evaluation of (i) and (ii) in the context of a real-world human movement assessment application.

The remainder of the paper is structured as follows. "[Related work](#)" discusses related work. "[Setting the scene](#)" introduces the real-world human movement assessment application for motivating our work, illustrating the current trust problems, and showing the evaluation context. "[A Framework for Post-Processing SAT Body Data](#)" introduces the employed basic technologies, describes the filter framework and its components (i). "[Experiment and Evaluation](#)" describes the evaluation experiment including the PCA based assessment (ii) and the results of applying different filters to human movement data (iii). Finally, "[Conclusions and Future Work](#)" concludes the paper and discusses possible directions of future work.

Related work

There are quite a few SAT systems providing human pose data. Accurate but rather expensive key-point-based systems use more than one camera and additional sensors such as, e.g., inertial measurement units [31, 33] and pressure sensors [7], to estimate poses. It is not difficult to create smooth stick figure animations based on movement data coming from these systems.

On the budget solution end, SAT uses commodity hardware and estimates the body key-points using computer vision (CV) solutions. Such solutions can estimate (multi-person) body poses [29, 30].

The CV solutions for pose estimation use large data sets annotated with pose information to train neural networks. It is a hot research topic and new methods have been proposed in recent years and advanced the state-of-the-art significantly. For instance, Residual Convolutional Networks [17]—ResNetX for different numbers X of layers—have successfully been applied to key-point detection. Xiao et al. [38] used a ResNet50 model and Papandreou et al. [29, 30] a ResNet101 model. The common objects in context (COCO) data sets⁴ initially introduced by Lin et al. [23] define benchmarks for competitions in key-point detection and other image processing tasks. The top-ranked models of the COCO key-point detection challenge set the state-of-the-art in this field. These challenges have been conducted

¹ <https://en.wikipedia.org/wiki/Kinect>.

² <https://orbbec3d.com/astra-mini>.

³ <https://realsense.intel.com>.

⁴ <https://cocodataset.org>.

annually since 2016; the rankings of the best-performing models^{5,6} changes annually as well. While ResNet-based models are still among the top-entries (regarding the average precision on the validation set) they are nowadays outperformed by other models, e.g., those defined by McNally et al. [26], by Cai et al. [6], and by Zhang et al. [42].

CV solutions exist even for 3D pose estimations. For instance, Xu and Takano [39] proposed a graph convolutional network architecture for 2D-to-3D human pose estimation tasks setting the current state of the art. Choi et al. [10] proposed MobileHumanPose for real-time 3D human pose estimation from a single RGB image suitable even for real-time pose estimations on mobile devices. Lin and Lee [22] introduced improvements to multi-view multi-person 3D pose estimation. Further accuracy improvements can be expected in the near future, also because of recent fundamental technology improvements: Gong et al. [13] introduced PoseAug, an auto-augmentation framework that enables the generation of 2D–3D pose pairs for training 3D pose estimation models. Ma et al. [24] improved context modeling that, in turn, reduces the ambiguity of 3D joint positions with the same 2D projection. Last but not least, Muller et al. [27] developed new datasets and methods that significantly improve human pose estimation with self-contact, i.e., when humans cross their arms and legs, put their hands on their hips, etc.

The aforementioned key-point 2D and 3D detection models are trained on static images of one or several humans in the wild, annotated howsoever with the ground truth of key-points. These models can trivially be used to animate movement sequences by concatenating the sequence of key-points estimated from a sequence of frame images of a human movement video. This approach has been implemented in PoseNet⁷ for ResNet models. PoseNet constitutes the SAT front-end of the tools and experiments in the present paper. However, more powerful pose estimation models are employed and made available in OpenPose,⁸ AlphaPose,⁹ and HRNet.¹⁰

The approach achieves fairly good key-point animations, even for several humans moving in the wild. Better performance could be expected in the easier single-human in-door scenario addressed in this paper. However, as the key-points of each frame (image) of a video sequence are estimated in isolation, the models minimize the estimation error only for the current frame and disregard previous and subsequent

frames. This leads to jumpy key-points in subsequent frames and overall to flickering animations.

There are two principle approaches to mitigate the problem, a supervised and an unsupervised approach.

The supervised approach is also referred to as *pose tracking*. It needs ground truth data, i.e., videos with annotated key-points for each frame. For instance, Chiang et al. [9] use supervised learning with a ground truth from a motion capture SAT system to remove noise in commodity SAT data based animations of physiotherapy exercises. Therefore, a Gaussian process regression model is trained by recording movements simultaneously with a commodity Kinect sensor and a high-end motion capture SAT system. To avoid that different body sizes affect the regression model, all transaction data are standardized. Cherian et al. [8] first estimate sets of key-point candidates frame-by-frame and optimize the animation by selecting the best-fitting candidates for each frame that minimize the overall flickering. Similar two-stage detect-and-track approaches are suggested by Girdhar et al. [12] and by Xiao et al. [38]. Wang et al. [36] uses a three-step approach consisting of simultaneous key-point detection and tracking in short video sub-sequences, a concatenation of the resulting animation sub-sequences, and a final optimization step for smoothing the animations. For the improvement of 3D pose tracking, Yuan et al. [41] introduced a simulation-based approach that integrates image-based kinematic inference with physics-based dynamics modeling.

Similar to the COCO challenge, there exist a PoseTrack¹¹ benchmark and pose tracking challenges Andriluka et al. [2]. There are, however, fewer data sets and contenders in the pose tracking than in the pose estimation, arguably also due to the difficulties in providing video data annotated with ground truth key-points.

The alternative approach is based on unsupervised learning and works without access to a ground truth in video data. It can simply be added as post-processing to pose estimation, but also to supervised pose tracking. To remove flickering without additional hardware equipment or ground truth video data annotations, we suggest signal processing filters that can be applied on the key-point animations to smooth the noisy signals. The principle idea is not new: a moving average computation, individually applied to each key-point coordinate time series, can be seen as such a filter. This is a common post-processing step used, e.g., in PoseNet.

This approach works for the improvement of 2D and 3D alike. It has successfully been applied in other CV solutions, e.g., for estimating the positions of robots where the signals from hardware sensors are filtered [40]. Filters are also used in other work to improve the accuracy of position and

⁵ <https://paperswithcode.com/sota/keypoint-detection-on-coco>.

⁶ <https://paperswithcode.com/sota/pose-estimation-on-coco-test-dev>.

⁷ <https://github.com/tensorflow/tfjs-models/tree/master/posenet>.

⁸ <https://github.com/CMU-Perceptual-Computing-Lab/openpose>.

⁹ <https://github.com/MVIG-SJTU/AlphaPose>.

¹⁰ <https://github.com/leoxiaobin/deep-high-resolution-net.pytorch>.

¹¹ <https://posetrack.net>.

Fig. 1 AIMO fitness app production with Skeleton Avatar Technology (SAT) based replay: before (left) and during (middle) a deep squat movement. Video replay of the same movement (right)

motion data [16, 35]. We take a different approach and optimize not only for improved accuracy but also for a smoother animation. To the best of our knowledge, no such solutions have been applied and evaluated systematically in the context of SAT-based animations.

On top of pose estimation and tracing, there is a large body of research that validates commodity SAT systems. We distinguish studies that *indirectly* validate SAT by validating their *usage* in an application context from studies that *directly* compare SAT body key-point with some ground truth (validations) or with high-accuracy SAT systems (agreement). For an overview of these studies, we refer to Hagelbäck et al. [15]. In summary, commodity SAT systems perform well enough to pass medical validation studies. Also, trust-building animations based on the key-point data they produce are acceptable on certain body parts, e.g., the head. Their performance on other body parts, such as the lower arms and legs, remains poor, which makes the animations flicker and hampers the trust in the underlying technology and the motion analysis results.

Finally, we acknowledge the work on estimating volume data—as opposed to key-point-based stick figures—of human poses and movements from images and videos, such as suggested by Kocabas et al. [20]. We also acknowledge the work on creating free animations—as opposed to scientific visualizations—let it be volume animations [43] or stick figure animations [1]. All these somewhat related approaches are outside the scope of this paper.

Setting the scene

This section discusses a concrete application context to illustrate the problem and to motivate the requirements of validity and smoothness. It gives insights why flickering stick figure animations are a principle problem and how it emerges.

A Real-World Human Movement Assessment Application

The movement scan technology AIMO (Artificial Intelligence in Motion)¹² allows users to perform a mobile movement scan of an overhead deep squat exercise for assessing the overall functional quality of the movement and identifying the weakest links in the kinetic chain of the body. The assessment reveals how well a person can control their movement, their average range of motion, stability, and coordination. Compensations identified during this movement provide expert sufficient information for creating personalised preventive training programs to improve the individual movement patterns.¹³ Based on the scan and the assessment, personalised preventive exercises for improving the lacking mobility and strength are proposed to prevent future pain and injuries.

Traditional movement assessments without SAT systems rely completely on the judgement of human experts and,

¹² <https://apps.apple.com/us/app/aimo/id1503692093>.

¹³ <https://blog.nasm.org/a-guide-to-movement-assessments#overhead-squat-assessment>.

usually, the movement has to be repeated several times. The use of SAT systems helps experts to assess clients faster and objectively. Furthermore, clients can assess themselves, e.g., at home, unsupervised and with a higher frequency.

While the validity of the automated assessment is a prerequisite for offering the app, trust in the technology is a key driver for successful adoption. Movement replay using stick figures can be used to visualize individual movement inefficiencies, which creates confidence and trust in the tracking quality and the validity of the motion analysis results, see Fig. 1.¹⁴ However, in customer surveys, AIMO found that flickering during movement replays immediately diminishes the feeling of trust in the results and the entire underlying technology, in general. Consequently, SAT-body-data-based stick figure replay was disabled in the first public versions of the app.

To overcome this obstacle, the movement data needs to be smoothed and flickering should be suppressed. However, when smoothing the SAT body data, it is essential that it still properly reflects the entire movement including all movement compensations identified during the motion analysis.

The Reason for Noisy SAT Body Data

The deep squat application uses PoseNet¹⁵ that is based on the TensorFlow Framework¹⁶ PoseNet is a convolutional neural network (CNN) trained to estimate human poses in images. In the AIMO app and also in the experiments in this paper, PoseNet is used to estimate motion sequences in videos, i.e., image sequences. The example demo of TensorFlow shows that PoseNet can be used for posture estimation of one or more people in real-time but, with quite some flickering [28].

PoseNet takes the video frames of a movement as input. For each frame, it returns the coordinates of key-points, more precisely, the key-points of each pose, a confidence score for the whole pose, and confidence scores for each key-point. The confidence scores “determine the confidence that an estimated key-point position is accurate. [...] It can be used to hide key-points that are not deemed strong enough.” [28]. The key-points are returned as 2D coordinates of the original frame. They can be used to create a stick figure of the pose. A sequence of such stick figures of poses corresponding to a sequence of video frames gives an animation of the movement in the video.

Fig. 2 PoseNet estimates the key-point positions differently, e.g., the knee position shown in the circle, even for one and the same pose image (right). The knee positions shown in the three circles (left) were estimated on the same image with artificially changed brightness, exposure, and contrast

The problem of this CV solution for pose estimation is that every frame of a video is estimated separately. Even if a neural network was trained very well for a special kind of movement, there would still be some uncertainty where the key-points actually are (therefore the confidence scores). This even holds for human experts: estimating, for example, a knee key-point in a frame of a video is difficult and to mark it exactly is nearly impossible when the subject wears clothes. So, even the ground truth data contains statistical errors. Training with image instead of video data and inaccurately annotated training data are only two reasons for the observed inaccuracy of key-points suggested by SAT systems. Hence, even with more training data, there will always be noise in the estimated key-points, which leads to flickering stick figure animations later on.

To illustrate this inaccuracy, Fig. 2 shows a photo with key-points estimated by PoseNet. If only the brightness, exposure, or contrast of the same photo is slightly changed—done here with an editing program—the points are estimated differently. This can be seen in the picture with the knee

¹⁴ App Store <https://apps.apple.com/lu/app/aimo-movement-scan/id1503692093>, Google Play <https://play.google.com/store/apps/details?id=com.aimo.fit.production>.

¹⁵ <https://github.com/tensorflow/tfjs-models/tree/master/posenet>.

¹⁶ <https://www.tensorflow.org/>.

Fig. 3 The (experimental) framework for post-processing Skeleton Avatar Technology (SAT) body data: processing pipeline with its components and variation point (2) for changing the filter

key-point as an example. The estimated key-points are not completely off, but would create a flickering when (re-) playing them in subsequent frames one after another in an animation. For a smoother stick figure animation, that noise needs to be removed from the estimation data.

A Framework for Post-Processing SAT Body Data

This section first introduces our framework for smoothing stick figure animations. It then suggests three noise cancelling filters. Finally, it formalizes the assessment of the filter quality, i.e., its validity and smoothness, two contradicting criteria.

The Framework and its Components

Smoothing of stick figure animations can be done with moving average methods that position each key-point at the coordinates that is the average of the coordinates of this key-point in a fixed size window of adjoined frames. Noise removal is well known in signal processing and several filter approaches have been used including high-pass and Kalman filters. It is, however, an open question if these filters also lead to smoother still valid stick figure animations that outperform the moving average methods. The suggested framework aims at answering this question experimentally in a systematic way.

The experimental setup (and the framework) is structured as shown in Fig. 3. It processes the recorded video sequences as follows:

- For each video, the coordinates of the key-points are estimated frame by frame with PoseNet.
- A filter is selected and the selected filter is applied to the key-point data.
- Optionally, the filtered and unfiltered movements can then be displayed.
- Based on the key-point data of many test videos, a *Validity Score* and a *Smoothness Score* are calculated. How the scores are composed is explained in section “[Validity Assessment](#)” and “[Smoothness Assessment](#)”.
- As a result, a weighted overall score is calculated for each filter which makes it possible to compare the filters with each other.

The production system comprises of the components for pose estimation (1), the presumably best filter (2), and the stick figure animation replay (3).

Noise Cancelling Filter Components

Below we introduce three filters—Moving Average, High-pass, and Kalman filters—that we assessed in the present paper.

Moving Average

Used, e.g., for smoothing financial series, the Moving Average is a simple filter to smooth out noisy data. As its input, it takes a series of real values. The result is also a series of real values in which the individual value is calculated as the average of a fixed number of adjoined values of the input series.

In our context, the input for the Moving Average filter is a time series of the x - and y -coordinates of each key-point. The input series $x(t)$, $y(t)$ contains n coordinates per key-point; $2k + t$ defines how many x - and y -values are averaged for one output key-point $x'(t)$, $y'(t)$.

$$z'_t = \frac{1}{2k+1} \sum_{t'=t-k}^{t+k} z_{t'} \quad (1)$$

$$t = k+1, k+2, \dots, n-k.$$

for $z \in \{x, y\}$ and $z' \in \{x', y'\}$. The output of this filter are smoother coordinate sequences x'_t, y'_t , one for each key-point. In the beginning of the time series and in the end, there are not enough points to calculate an average. Therefore, the resulting time series is $2k$ data points shorter than the original series. Alternatively, key-points averaging over fewer raw data values could be used at the beginning and at the end of the series.

It is important to note that the Moving Average filter is parameterized with parameter k controlling the window size $2k+1$.

Fig. 4 The Kalman filter process: after an initialization for time step $t = 0$, iteratively for each time step t , the measured values z_t are corrected to $\hat{z}_{t,t}$ based on a measurement uncertainty R_t , and based on predicted (extrapolated) measurement value $\hat{z}_{t+1,t}$ and measurement uncertainty $P_{t+1,t}$

High-Pass Filter with the Fourier Transform

High-pass filters are used to remove high-frequency signal components, and the flickering can be understood as a high-frequency signal on top of the actual movement information. They are widely used in electronics, e.g., to remove noise from sound signals. The idea behind high-pass filters is that high frequencies are the result of random processes during recording and do not belong to the original sound signal. Consequently, these frequencies are removed to get a signal without noise.

High-pass filters can be implemented using the well-known Fourier transform. With a Fourier transform it is possible to transform signals from time space to frequency space. As for the Moving Average, we transform each key-point and each of its coordinates x and y separately. Each such time series is transformed to the frequency space, high frequency components of the transformed series are removed, and the filtered series is transformed back using the inverse of the Fourier transform.

For transforming the key-point coordinates, the discrete Fourier transform is used:

$$f_k^z = \sum_{t=0}^{n-1} z_t e^{-\frac{i2\pi}{n} kt} \quad (2)$$

$$k = 0, 1, \dots, n-1.$$

with f_k^z the strength of the k -th frequency and, as before, n the length of the data series and $z \in \{x, y\}$.

In frequency space, all distinct frequencies above a maximum K are set to zero. This is the actual high-pass filtering. Then the signal is transformed back without the “disturbing” frequencies and the signal is the same as before, but with

less noise. Therefore the inverse discrete Fourier transform is used:

$$z'_t = \frac{1}{n} \sum_{k=0}^{n-1} f_k^z e^{\frac{i2\pi}{n} kt} \quad (3)$$

$$t = 0, 1, \dots, n-1.$$

We note again that the High-pass filter is parameterized with the cutoff frequency K .

Kalman Filter

The Kalman filters are used to remove the noise of sensor data. In the present case, the neural network PoseNet that estimates the key-points of a pose is the “sensor”.

More precisely, Kalman filters use a measured time series data containing statistical noise and other inaccuracies to produce estimates of the underlying actual signal by estimating and updating (joint) probability distributions over the signal variable(s) for each time frame. The signal variable(s) for each time step are estimated based on the estimations of the previous step, the new measured data, and the expected change of the variables between the steps.

For each time step t , filtering is usually separated into two phases: correction and prediction, cf. Fig. 4.¹⁷ In the correction phase, the actually observed value z_t is corrected to the output value $\hat{z}_{t,t}$ using the expected value $\hat{z}_{t,t-1}$ predicted in the previous step¹⁸. Also, the estimation uncertainty $P_{t,t}$ is

¹⁷ Figure inspired by <https://www.kalmanfilter.net/kalman1d.html>.

¹⁸ In the formalization of Kalman filters, we use subscripts $x_{t',t''}$ for different variables X . Here t' refers to an estimate of X for time step t' , while t'' refers to an estimation done in time step t'' .

updated using the predicted estimation uncertainty of the previous step $P_{t,t-1}$. For both updates, the so-called Kalman gain is computed first, based on $P_{t,t-1}$ and a measurement uncertainty R_t . In the prediction step, the next state $\hat{z}_{t+1,t}$, i.e., the next expected value(s) of the signal variable(s), is estimated along with a corresponding next expected estimation uncertainty $P_{t+1,t}$. Then the filter advances to the next step $t \leftarrow t + 1$. The filter process starts with the predict phase initialized with guesses of the initial state $\hat{x}_{0,0}$ and the initial estimation uncertainty $P_{0,0}$. For details, we refer to Welch et al. [37].

All updates in step t are based on the previous step $t - 1$ alone. Therefore, it is not necessary to keep all past estimations and observations in memory. Hence, the filter process is computationally very efficient [11].

We make the below simplifying assumptions to set up the Kalman filter, which allow for an easy adoption of the filter regardless of the actual movements and SAT systems. We decided so to assess the baseline performance of this type of filter. Any possible improvement, also discussed below, is matter of future work.

First, the horizontal and vertical movements of human joints are naturally dependent. Although it is possible to use the Kalman filter with multi-variable signals, we again filter the x - and the y -coordinates of each key-point separately, i.e., $z \in \{x, y\}$ in the formulas below. Improvements could observe the joint distribution of x and y coordinates—for a particular exercise in question or for human movements in general—and exploit this in a multi-variable setup of the Kalman filter.

Second, we do not exploit any knowledge about the dynamics of the underlying movement and estimate the next state to be as the previous one. Improvements could consider the physics of movements and extrapolate the next position based on movement trajectory, speed, and acceleration. The dynamics of the system could be model for human movements in general or with a particular exercise in mind.

Finally, the measurement uncertainty of the underlying SAT system changes from movement to movement, e.g., depending on environment conditions such as light and contrast, but also for different frames and key-points of one movement, e.g., due to partial hiding of limbs and joints. Although PoseNet provides uncertainty scores per frame and even per key-point, we assume a constant uncertainty for all exercises, all frames, and all key-points.

With these assumptions, the (baseline) Kalman filter is defined as follows.

For the prediction phase:

$$\hat{z}_{t+1,t} = z_{t,t} \quad (4)$$

$$P_{t+1,t} = P_t \quad (5)$$

which reflects the simplified constant dynamics.

For the correction phase:

$$K_t = \frac{P_{t,t-1}}{P_{t,t-1} + R} \quad (6)$$

$$\hat{z}_{t,t} = \hat{z}_{t,t-1} + K_t(z_t - \hat{z}_{t,t-1}) \quad (7)$$

$$P_{t,t} = (1 - K_t)P_{t,t-1} \quad (8)$$

We take the first measured signal value as the initial corrected signal value, i.e., $\hat{z}_{0,0} = z_0$. The corrected signal value output is the filter result in each time step, i.e., $z'_t = \hat{z}_{t,t}$. Note that the remaining parameters for our instance of the Kalman filter are the initial estimation uncertainty $P = P_{0,0}$ and the (constant) measurement uncertainty R . In our one-dimensional case, P and R are two scalar values. For multi-dimensional Kalman filters P and R would be matrices.

Automated Assessment and Scoring of Validity and Smoothness

The *validity* of a SAT movement animation measures the degree to which visualized key-points match the actual positions of key-points in the ground truth. However, this ground truth is not available.

The *smoothness* of a SAT movement animation measures the absence of jumps between consecutive frames of an animation.

An animation that is perfectly valid is (almost) perfectly smooth. However, smoothness can obviously be optimized by reducing validity: an animation that just shows one static picture is perfectly smooth but is certainly not valid for a dynamic movement. Hence, we need to assess both properties of the animations proposed after filtering and trade them off against each other. In the following two subsections, we discuss how we measure validity and smoothness, resp., in isolation, before we finally suggest a trade-off in a common assessment score. We apply the latter in the subsequent experimental assessments of the filters.

Validity Assessment

As we do not have any ground truth validity, any validity score can only be an approximation of the actual property. We suggest such a validity score based on principle component analysis (PCA). For this experiment the library `ml-pca` provided via `npm` was used for this. The idea behind this is that random noise added to a signal only marginally changes the principle components of that signal. Therefore, if we compare the principle components of a noisy input signal per key-point with the results of a validly filtered signal for that key-point, we expect only minor changes.

To illustrate this effect, we generate a sinoid key-point signal, a fake ground truth that is unknown in reality. Then

Fig. 5 Positions x, y of an artificially generated key-point signal (true signal, left) and the same signal with Gaussian random errors added (noisy signal, right)

Fig. 6 After principle component analysis (PCA), bi-plots of the principle components of the true key-point signal (left) and the noisy key-point signal (right) from Fig. 5. The principle components are (almost) identical

we add random noise to this signal and compare the principle components of both the true and the noisy signal.

In this illustration, we generate 5 seconds of data of a cyclically moving key-point, e.g., the left knee in a deep squat. It is sampled with 100 data points per second. The movement is repeated with a frequency of $f = 1$ Hz. We set

$$y_t = 5 \cos(2\pi f t) \quad (9)$$

$$x_t = |\sin(2\pi f t)| \quad (10)$$

To create a realistic noisy signal, we add Gaussian ($\mu = 0, \sigma = 1$) random errors to the (x, y) coordinates of the ground truth signal, multiplied with 10% of the amplitudes in each dimension, i.e., (0.1, 0.5). The data points of the true and the noisy signals are shown in Fig. 5.

Next, we compute the principle components of both signals with Matlab [25]. The two eigenvectors of the true signal are $e_1 = (-0.0846, 0.9964)$ and $e_2 = (0.9964, 0.0846)$ explaining 99.96% and 0.04%, resp.,

of the variability. The two eigenvectors of the noisy signal are $e_1 = (-0.0813, 0.9967)$ and $e_2 = (0.9967, 0.0813)$ and explain 99.87% and 0.13%, resp., of the variability. The principle components of both signals are depicted in the bi-plots of Fig. 6. They are almost identical.

In generalization of this illustration, an appropriate measure of validity can be based on PCA by checking the similarity of the principle components of signals before and after filtering. If the filter only removes random noise, we expect similar principle components before and after filtering. If instead the filter systematically modifies the signal, we expect them to be different.

We define such a measure as follows.

$$valid = \|e_1^{raw\ data} - e_1^{filter}\|_2 + \|e_2^{raw\ data} - e_2^{filter}\|_2 \quad (11)$$

with e_1, e_2 the two eigenvectors of the raw and the filtered data, respectively, and $\|\cdot\|_2$ the vector length (l_2 norm) of the difference vectors. The smaller the sum of differences between the eigenvectors of the raw data and the filtered data, the higher the validity.

Obviously, if the data would not be filtered at all, identity filter, the validity would be best, but the flickering would still be there. Therefore, a measurement of the flicker reduction is needed, as well.

It is interesting to note that PCA could possibly be used as yet another filter. Babu et al. [3] used PCA for (static) image denoising by transforming a pixelated image into the PCA domain and keeping only the most significant components before they transform it back. Similar to the high-pass filter, an analog PCA-based approach to time series denoising would transform a the key-point sequence into the PCA domain and keep the most significant components before transforming it back. Although such a PCA filter would be different from our assessment method, we did not include it to keep a logic distance between the object of assessment and the validity assessment criterion.

Smoothness Assessment

To measure the quality of the flicker reduction, we propose to measure the distances of the coordinates of one key-point to the coordinates of the same key-point in the next frame. The reason for this is that the flickering results from the distance between the key points in consecutive frames. For a stationary image the smoothness would be zero (best). The distance between the points in 2D-space is calculated with the l_2 norm again. The overall smoothness is the average of all distances between key-points of consecutive frames in a sequence:

$$smooth = \frac{\sum_{t=1}^n ||p_{t-1} - p_t||_2}{n} \quad (12)$$

with n the number of frames in a sequence.

Normalizing and Aggregating Validity and Smoothness

The metrics *valid* and *smooth* have different value ranges and need to be normalized to make them comparable before aggregating them to a common score. Therefore, we use the tail distribution of the values, i.e., the complementary cumulative distribution function (CCDF). CCDF calculates for each value v of *valid* (*smooth*) the probability of finding a larger (worse) or equal value in the whole population *valid* (*smooth*). Note that this probability is close (but not equal) to 0 for the worst possible *valid* (*smooth*) values and 1 for the best. Formally, we define the scoring functions for values $v \in valid$ ($v \in smooth$) as:

$$valid_s : valid \rightarrow (0, 1] \quad (13)$$

$$valid_s(v) : P(valid \geq v) \quad (14)$$

$$smooth_s : smooth \rightarrow (0, 1] \quad (15)$$

$$smooth_s(v) : P(smooth \geq v) \quad (16)$$

We calculate the probabilities empirically based on the measured data.

Finally, for each filter, the corresponding scores are aggregated using the weighted harmonic mean.

$$WHM = \frac{w_v + w_s}{\frac{w_v}{valid_s} + \frac{w_s}{smooth_s}} \quad (17)$$

with the weights w_v and w_s for the scores. If both are considered equally important ($w_v = 1, w_s = 1$), the aggregated score WHM is the harmonic mean of the two scores. We will vary w_v and w_s in the experiments. Regardless of the weights, the best possible WHM score is 1 and the worst possible is almost (but never equal to) 0.

Experiment and Evaluation

This section describes our experimental evaluation. We first explain the setup of two experiments in “[Experimental Setup](#)” before we present their results in “[Experimental Results](#)” and discuss the findings and implications in “[Discussion](#)”.

Experimental Setup

This subsection introduces two experimental settings that have been studied. Before it describes the subjects and the collected data, and the details of the implementation, which are common to both settings.

Subjects and Collected Data

We recorded and assess ca 500 deep squat movements—some squats were only recorded during in the downward movement—of ca 25 adult female and male individuals recorded at ca 5 different locations with different lighting conditions and diverse environmental conditions. We did not choose the individuals and the locations randomly to sample a specific population in a fair way. They were rather selected based on availability.

Implementation Details

For this project, TensorFlow PoseNet was used to estimate the poses [32]. This neural network creates 2D pose data that can be used for real-time estimation. However, to be able to

Fig. 7 Visualization of a sequence of pose estimations (of a half deep squat movement) based on raw key-points from PoseNet

use all filters, the pose data were collected and stored after recording an entire movement.

All code was written in Angular TypeScript, but it uses third party components for the Kalman filter as detailed below.

Experiment 1: Fixed Parameter Setting

The initial parameter setting of the filters was determined after visual pre-analysis together with an expert from AIMO. They are defined below.

Moving Average

The presumably best results were received when the average is calculated with five points, i.e., the parameter settings for the Moving Average described in Eq. (1) is $k = 2$.

An only theoretic problem occurs if the estimated video contains movements with five frames or less. Then the filter produces a static visualization, not an animation. We drop these too short sequences. However, this problem did actually never occur in the test data.

Fourier transform and implementation of the high-pass filter

As we do not know what frequencies are to be expected in advance, the high-pass filter set the highest κ percent of the observed frequencies to zero, i.e., the high-pass filter parameter $K = (1 - \kappa)n$ with n the number of frames in a sequence. In other words, in the frequency space $f_k, k \in [0, n]$, frequencies $k \in [(1 - \kappa)n, n]$ were set to 0 before transforming the signal back to the time space. The presumably best results were received with $\kappa = 80\%$.

Kalman filter

For this experiment, the library `kalmanjs` was used. It implements a filter for one-dimensional data. It is publicly available via the standard dependency management system `npm`. We followed the advices on the homepage and in a blog post. It also contains details on how the filter was implemented [4, 5].

Recall, to use the Kalman filter, parameters P and R have to be defined; P is the dynamic model specific estimation uncertainty and R is the sensor specific measurement uncertainty.

The visual inspection suggested that for higher P -values the results are smoother. For higher R -values, the filter reacts more dynamically to changes in the data. The following settings provided the presumably best results: $P = 3$ and $R = 0.05$

Experiment 2: Parameter Identification

Different parameter settings can be chosen for each filter to achieve results that might even better fit the individual expectations. Therefore, we run a second experiment with the same filters and their implementations. However, in this second experiment, we approximate a solution of the inverse problem of finding the optimal parameters for each filter that maximize the weighted harmonic mean for each ratio of smoothness and validity weighting.

We use a simple grid analysis. For each filter and weighting ratio, different parameters were tested, the test recordings are filtered with these parameter settings, and the scores were calculated again. For each filter and weighting ratio, we choose the maximum aggregated score.

Moving Average The following parameter settings were used for the Moving Average: $k = 1, 1.5, 2, 2.5, 3$. For the parameter setting $k = 1.5$ and $k = 2.5$, the outermost points of the original time series were weighted 50%.

Fourier transform and implementation of the high-pass filter For the high-pass filter, the parameter κ was set to 0.9, 0.85, 0.8, 0.75, 0.7 leading to frequency cut-off points $K = (1 - \kappa)n$ with n the number of frame in a sequence.

Kalman filter The Kalman filter has two parameters that affect the filter results. First we set $P = 3$ and varied $R = 0.0, 0.05, 0.1, 0.15, 0.2$, then we set $R = 0.05$ and varied $P = 1, 2, 3, 4, 5$ (Fig. 7).

Experimental Results

An example (frame-by-frame) visualization of a deep squat movement smoothed with a moving average calculated over five data points is shown in Fig. 8. The visualization of the same movement instance after applying a high-pass filter with $K = 80\%$ is shown in Fig. 9. Figure 10 shows the visualization after applying the described instance of the Kalman filter for smoothing this movement instance. In all sequences

Fig. 8 The same sequence of pose estimations as in Fig. 7 filtered with the Moving Average filter using a 5-point smoothing (therefore fewer frames)

Fig. 9 The same sequence of pose estimations as in Fig. 7 filtered with the High-pass filter

Fig. 10 The same sequence of pose estimations as in Fig. 7 filtered with the Kalman filter

Table 1 Mean and standard deviation of smoothing and validity scores of recorded videos

Filter	Smoothing Score mean	Standard deviation	Validity Score mean	Standard deviation
Moving Average:	0.55258	0.01132	0.54631	0.01310
High-pass filter:	0.26111	0.00900	0.50821	0.01231
Kalman filter:	0.68731	0.01005	0.44649	0.01292

the stick figures get apparently smaller. This is a visual effect caused by some joints—wrist, elbows, shoulders, knees—moving closer to the camera.

The visual inspection by the AIMO expert *subjectively* revealed the Kalman filter as the overall best approach. This is hardly a scientific result, but supports the idea that there are perceivable differences in the animations when using different filter technologies.

The motion visualization shown in Fig. 8 is missing two frames at the beginning. Due to the application of the Moving Average with a 5-point smoothing, no smoothed values can be calculated for the first two frames and they are therefore missing.

Table 2 Mean of aggregated scores of recorded videos

Filter	Aggregated Score mean
Moving average	0.54943
High-pass filter	0.34498
Kalman filter	0.54132

Experiment 1: Fixed Parameter Setting Results

Besides the visual feedback, the filters were tested according to the methods explained in the “[Validity Assessment](#)” and “[Smoothness Assessment](#)”. For the validity assessment,

Fig. 11 Aggregated scores of smoothness and validity (y axis) and the weightings of the smoothness and validity scores in this aggregation (x axis) for the three assessed filters. The figure shows how the weighting of smoothness and validity affects the aggregated score, i.e., the presumed overall filter quality

Fig. 12 Similar to Fig. 11, aggregated scores of smoothness and validity (y axis) and the weightings of the smoothness and validity scores in this aggregation (x axis) for the three filters, each optimized using grid search within the range of parameters given in “Experiment 2: Parameter Identification” for each weighting

a PCA was performed for each of the videos and the key points contained and, for each filter, the difference between the principal component of the raw data and the filtered data was calculated. These values were then normalized using the complementary cumulative distribution function (CCDF) and averaged to a movement validity score per video.

For the smoothing assessment, the average distance to the following point for each video and the key points contained was calculated, then normalized using the CCDF, and averaged to a smoothness score per video.

The descriptive statistics (mean and standard deviation) of the validity and smoothness scores are shown in the Table 1. All values are rounded to 5 decimals.

We aggregated these scores with an equally weighted harmonic mean. The resulting average aggregated score results

for the three filters are shown in the Table 2. Then we aggregated the two scores with 11 different weightings ranging from no weight on the smoothing and the whole weight on the validity score, over a 50:50 weighting already reported in Table 2, to the other extreme of putting the whole weight on the smoothing and no weight on the validity score. The results are shown in Fig. 11.

Experiment 2: Parameter Identification Results

For each filter and the corresponding parameter setting a smoothing score and validity score were calculated and aggregated using the weighted harmonic mean with 11 different weightings, as described above. The diagram in

Fig. 13 The same sequence of pose estimations as in Fig. 7 filtered again with the High-pass filter. Here, only the half squat movement was filtered and it shows differences to the sequence displayed in Fig. 9 where the High-pass filter was applied on the whole squat movement

Fig. 12 shows the respectively highest aggregated score results for each filter and weighting.

Discussion

As the diagram in Fig. 11 shows, the Moving Average filter performs best in the fixed parameter setting when the Smoothing Score is weighted less or equal to the Validity Score. With shifts in the weights toward validity, the Kalman filter outperforms the Moving Average filter. The smoothness to validity break-even point is expected to be between 50:50 and 60:40. In none of the weightings the high-pass filter performs best.

If the scores are calculated with different parameterizations, as shown in the diagram in Fig. 12, the picture is slightly different. With equal weighting the Kalman filter reaches the highest aggregated score. The more weight is put on validity, the better the Moving Average filter performs. The smoothness to validity break-even point is expected to be close to 40:60 in favor of validity. Again, in none of the weightings the High-pass filter performs best.

For each application in which movement data need to be smoothed for an appropriate animation, it must be decided individually what the quality goals are and how they impact the weighting. Our experimental results suggest, that Moving Average or Kalmar filters should be used depending on the quality goals. Moreover, they also suggest, that a high-pass filter is not appropriate in any setting.

However, before concluding the inappropriateness of the high-pass filter too soon, we need to identify potential reasons for the low performance as they might lay in our experimental setup, not in the technology. The high-pass filter does not affect the individual data points of a data series, but transforms the entire series. Therefore, a whole squat and a half a squat movement produce different results. For example, in Fig. 9 the whole movement was filtered but only the downward movement frames were displayed. In Fig. 13 the same video was cut after the downward movement, then filtered and displayed. The high-pass filter might get better results if the movements that are to be smoothed are periodic. Remember, that we also have half squat movement

videos in our data set. Anyway, this needs further systematic studies.

Threats to Validity

Theoretical validity: To evaluate the overall quality of the filters, two scores are defined that determine the opposing objectives. In detail, these are the movement fidelity, which is measured with the validity score (4.3.1) and the flicker reduction, which is expressed in the smoothing score (4.3.2). The scores are produced by an *empirical CCDF* and, therefore, are only valid in the context of this comparison.

According to the definition of the success criteria, both flickering reduction and fidelity of movement are important in an appropriate solution. Their definitions follow sound and well-known mathematical concepts for measurement, normalization, and aggregation. However, the corresponding scores *validity* and *smoothness* are approximations of the actual success criteria.

Internal validity

All filters were applied on exact the same raw data. While testing, the filters were not changed. So, one test did not affect the results of the following test. For each filter the exactly same input data were used. So internal validity is given.

Construct validity The central constructs in this study are the assessed filters. The assessments are made relative to each other. Therefore, the conclusions are limited to the compared filters and does not exclude others to even outperform any of these three.

External validity

It is important to use representative data to obtain valid test results. The test subjects recorded videos in different realistic environments showing them doing a squat. The setting for the videos varied, the subjects wore different clothes and filmed themselves in front of different backgrounds with different lighting. However, we did not select subjects nor locations as a random sample of a full population.

Also, we only selected one movement, the deep squat, which is a rather one-sided movement. The results of the Kalman Filter and Moving Average are not dependent on the direction of movement. Therefore, we consider the test

set with 500 squats as sufficient. The results of these two filters could (most likely) be transferred to other movements.

With the high-pass filter, the whole movement sequence is manipulated in the frequency space. As the Fourier transform searches for sinoid components in the data series, it can be expected that for more irregular movements—just as the half squat in Fig. 13—the results are worse. Therefore, the movement selection is relevant and, when testing with other movements than just squats, the results for the high-pass filter will be different.

Reliability: The type of experimental set-up and the application of the evaluation methods are described very precisely to be able to reproduce the experiment. When building the same experiment with the collection of other test data, very similar results should be obtained.

Limitations: As mentioned before, the scores are determined based on the comparison of the three specific filters, so these scores cannot be used to compare with scores from other filters in other experiments. Also, adding or removing a filter would change the scores as a result of the *CCDF*. What will by-and-large not change is the champion filter for a certain weighting of smoothness and validity. Another limitation is the generalization of the score for the high-pass filter, which could be much better for periodic movements and worse for very non-periodic movements. This has to be tested in future work.

Conclusions and Future Work

The paper introduced a framework of assessing alternative filter components that smooth Skeleton Avatar Technology (SAT)-based movement animations (i). It consists of a filter plug-in and automated assessment and scoring of smoothness and validity of the animation (ii). The paper finally compares three different filters, Moving Average, high-pass and Kalman filters, in a real-world application, a fitness app around deep squat movements (iii). In this context, the animations need to reflect the actual movement faithfully and avoid flickering that is a result of noisy input raw data. The results indicate that different filters should be used under different weightings of expected smoothness and validity. The Moving Average tends to be better if validity is more important than smoothness; the Kalman filter in the opposite scenarios. The high-pass filter could not convince in any of the tested scenarios. Finally, the paper additionally solved the inverse problem of finding the best performing filter parameters for a given weighting of smoothness and validity.

To the best of our knowledge, it is the first study that applies filters from signal processing, high-pass and Kalman filters, to the problem of smoothing flickering animations and assessing the results automatically and objectively. This brings together two rather disjoint fields:

scientific visualization and signal processing. However, there is plenty of future work.

The employed inverse problem solver is a simple grid approach. Other derivative-free optimizations, such as, e.g., randomized optimizations or adaptive grid approaches, are well known to perform better and more efficient. They should be assessed. Beyond that, instead of experts determining the weighting of smoothness and validity and an inverse problem optimization finding the filter parameters, user could rate the overall acceptance of the animations under different (randomized) filter parameter settings to find the best one.

In addition to the two evaluated success criteria smoothness and validity, it could also be interesting to consider others. The Kalman filter, for example, is the only one that offers the possibility to smooth the data in real-time. Also, the calculation time and memory requirements could be included in an extended evaluation.

Since the signal processing filters are unaware of specific actions in the temporal sequences, it is impossible for them to know whether or not the difference between key-point locations in adjacent frames comes from the inaccurate detection or from the correct motions of a specific action. This might result in scale drifting. Future work will systematically assess whether or not this is an actual problem. If so, we need to analyze if the validity criterion penalizes scale drifting (or if another criterion should be added) and how it can be compensated (as part of the filters or in yet another post-processing step).

It would also be valuable to investigate other variants of the assessed filters or completely new filters for the same problem. To get better results with the Kalman filter, e.g., it is possible to use a 2D variant, to exploit the measurement uncertainty scores, to better approximate the system dynamics based on a statistical model of all observed movement instances, and to better guess the starting points based on such a model.

Moreover, extended and more systematically selected sets of exercises, recording environments, and subjects, respectively, would allow to generalize our results. Adding a real ground truth to the movements would strengthen the theoretical validity. We therefore plan to apply the filters to the PoseTrack data set [2] for further evaluation of the validity of smoothed skeletons, since the movement videos are not limited to one exercise and contains ground truth annotations for the video frames.

Finally, we will extend the experiments by assessing alternatives to the PoseNet SAT front-end, e.g., those based on the pose estimation and tracking models discussed in “[Related work](#)”, and alternative noise cancelling filters, e.g., filters based on Wavelet and PCA transforms.

Funding Open access funding provided by Linnaeus University.

Declarations

Funding Open access funding provided by Linnaeus University. Seed funding from the Linnaeus Center for Data Intensive Science and Applications (<http://lnu.se/disa>).

Conflict of interest None.

Availability of data and material (data transparency) yes, on request

Code availability (software application or custom code) yes, on request

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Aberman K, Li P, Lischinski D, Sorkine-Hornung O, Cohen-Or D, Chen B. Skeleton-aware networks for deep motion retargeting. *ACM Trans. Graph. (TOG)* 2020;39(4), 62.
2. Andriluka M, Iqbal U, Milan A, Insafutdinov E, Pishchulin L, Gall J, Schiele B. Posetrack: A benchmark for human pose estimation and tracking. 2017; <http://arxiv.org/abs/1710.10000>.
3. Babu Y, Subramanyam M, Prasad MG. Pca based image denoising. *Signal Image Process Int J (SIPIJ)* 1998;3(2), 236–244. <https://www.aircconline.com/sipij/V3N2/3212sipij18.pdf>.
4. Bulten W. Lightweight Javascript library for Noise filtering using Kalman filters. 2015; <https://www.wouterbulten.nl/blog/tech/lightweight-javascript-library-for-noise-filtering>. Accessed 20 June 2020.
5. Bulten W, Kalman JS. 2018; <https://github.com/wouterbulten/kalmanjs>. Accessed: 13 June 2020.
6. Cai Y, Wang Z, Luo Z, Yin B, Du A, Wang H, Zhang X, Zhou X, Zhou E, Sun J. Learning delicate local representations for multi-person pose estimation. 2020.
7. Casas L, Mürwald C, Achilles F, Mateus D, Huber D, Navab N, Demirci S. Human pose estimation from pressure sensor data. In: *Bildverarbeitung für die Medizin 2018*, Springer, New York, pp 285–290.
8. Cherian A, Mairal J, Alahari K, Schmid C. Mixing body-part sequences for human pose estimation. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
9. Chiang AT, Chen Q, Li S, Wang Y, Fu M. Denoising of joint tracking data by kinect sensors using clustered gaussian process regression. In: *Proceedings of the 2nd International Workshop on Multimedia for Personal Health and Health Care*, Association for Computing Machinery, New York, NY, USA, MMHealth '17, 2017; pp 19–25. <https://doi.org/10.1145/3132635.3132642>.
10. Choi S, Choi S, Kim C. Mobilehumanpose: Toward real-time 3d human pose estimation in mobile devices. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021; pp. 2328–2338.
11. Erik Cuevas DZ, Rojas R. Kalman filter for vision tracking. 2018; https://refubium.fu-berlin.de/bitstream/handle/fub188/19186/2005_12.pdf?sequence=1&isAllowed=y.
12. Girdhar R, Gkioxari G, Torresani L, Paluri M, Tran D. Detect-and-track: Efficient pose estimation in videos. 2017; <http://arxiv.org/abs/1712.09184>.
13. Gong K, Zhang J, Feng J. Poseaug: A differentiable pose augmentation framework for 3d human pose estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021; pp. 8575–8584.
14. Guzov V, Mir A, Sattler T, Pons-Moll G. Human poseitoning system (hps): 3d human pose estimation and self-localization in large scenes from body-mounted sensors. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4318–4329.
15. Hagelbäck J, Lincke A, Löwe W, Rall E. On the agreement of commodity 3d cameras. In: *ICIP* 2019.
16. Halvorsen K, Söderström T, Stokes V, Lanshammar H. Using an extended Kalman filter for rigid body pose estimation. *J Biomech Eng.* 2004;127(3), 475–483. <https://doi.org/10.1115/1.1894371>.
17. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. <http://arxiv.org/abs/1512.03385>
18. Hyndman R. Moving averages. In: Lovric M (ed) *International Encyclopedia of Statistical Science*, Springer, Berlin, 2010; pp 866–869. https://doi.org/10.1007/978-3-642-04898-2_380
19. Kalman RE. A new approach to linear filtering and prediction problems. *ASME J Basic Eng* 1960.
20. Kocabas M, Athanasiou N, Black MJ. VIBE: video inference for human body pose and shape estimation. <http://arxiv.org/abs/1912.05656>.
21. Kumar P, Mukherjee S, Saini R, Kaushik P, Roy PP, Dogra DP. Multimodal gait recognition with inertial sensor data and video using evolutionary algorithm. *IEEE Trans Fuzzy Syst.* 2019;27(5):956–65. <https://doi.org/10.1109/TFUZZ.2018.2870590>.
22. Lin J, Lee GH. Multi-view multi-person 3d pose estimation with plane sweep stereo. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021; pp 11886–11895
23. Lin T, Maire M, Belongie SJ, Bourdev LD, Girshick RB, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL. Microsoft COCO: common objects in context. 2014; <http://arxiv.org/abs/1405.0312>
24. Ma X, Su J, Wang C, Ci H, Wang Y. Context modeling in 3d human pose estimation: A unified perspective. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021; pp. 6238–6247.
25. MathWorks. MATLAB version 9.8.0.1323502 (R2020a). The Mathworks, Inc., Natick, Massachusetts. 2020.
26. McNally W, Vats K, Wong A, McPhee J. Evopose2d: Pushing the boundaries of 2d human pose estimation using neuroevolution. 2020.
27. Muller L, Osman AAA, Tang S, Huang CHP, Black MJ. On self-contact and human pose. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021; pp. 9990–9999.
28. Oved D, Alvarado I, Gallo A. Real-time human pose estimation in the browser with tensorflow.js. *TensorFlow Medium*. 2018. <https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5>.
29. Papandreou G, Zhu T, Kanazawa N, Toshev A, Tompson J, Bregler C, Murphy K. Towards accurate multi-person pose estimation in the wild. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017; pp. 4903–4911.

30. Papandreou G, Zhu T, Chen LC, Gidaris S, Tompson J, Murphy K. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In: Proceedings of the European Conference on Computer Vision (ECCV), 2018;pp. 269–286.
31. Pons-Moll G. Human pose estimation from video and inertial sensors. PhD thesis, Max-Planck-Institute for Intelligent Systems. 2014.
32. TensorFlow. Pose Detection in the Browser: Posenet Model. 2020. <https://github.com/tensorflow/tfjs-models/tree/master/posenet>
33. Trumble M, Gilbert A, Malleson C, Hilton A, Collomosse J. Total capture: 3d human pose estimation fusing video and inertial sensors. In: Tae-Kyun Kim GB Stefanos Zafeiriou, Mikolajczyk K (eds) Proceedings of the British Machine Vision Conference (BMVC), BMVA Press, 2017;pp. 14.1–14.13. <https://doi.org/10.5244/C.31.14>.
34. Van Loan C. Computational Frameworks for the Fast Fourier Transform. Society for Industrial and Applied Mathematics, 1992.
35. Wang J, Wilson W. 3d relative position and orientation estimation using kalman filter for robot control. In: Proceedings 1992 IEEE International Conference on Robotics and Automation, IEEE Computer Society, Los Alamitos, CA, USA, 1992;pp. 2638–2645. <https://doi.org/10.1109/ROBOT.1992.220044>.
36. Wang M, Tighe J, Modolo D. Combining detection and tracking for human pose estimation in videos. 2020.
37. Welch G, Bishop G, et al. An introduction to the kalman filter. 1995.
38. Xiao B, Wu H, Wei Y. Simple baselines for human pose estimation and tracking. 2018. <http://arxiv.org/abs/1804.06208>
39. Xu T, Takano W. Graph stacked hourglass networks for 3d human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021;pp. 16105–16114.
40. Xue Z. A comparison of nonlinear filters on mobile robot pose estimation. PhD thesis, Carleton University; 2013.
41. Yuan Y, Wei SE, Simon T, Kitani K, Saragih J. Simpose: Simulated character control for 3d human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021;pp. 7159–7169.
42. Zhang F, Zhu X, Dai H, Ye M, Zhu C. Distribution-aware coordinate representation for human pose estimation. 2019. <http://arxiv.org/abs/1910.06278>.
43. Zhang H, Starke S, Komura T, Saito J. Mode-adaptive neural networks for quadruped motion control. ACM Trans Graph 2018;37(4). <https://doi.org/10.1145/3197517.3201366>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations