# Real-Time Multi-Person Pose Tracking using Data Assimilation

Caterina Buizza Tobias Fischer Yiannis Demiris
Personal Robotics Laboratory, Dept. of Electrical and Electronic Engineering
Imperial College London
{caterina.buizza11, t.fischer, y.demiris}@imperial.ac.uk

## Abstract

*We propose a framework for the integration of data assimilation and machine learning methods in human pose estimation, with the aim of enabling any pose estimation method to be run in real-time, whilst also increasing consistency and accuracy. Data assimilation and machine learning are complementary methods: the former allows us to make use of information about the underlying dynamics of a system but lacks the flexibility of a data-based model, which we can instead obtain with the latter. Our framework presents a real-time tracking module for any single or multi-person pose estimation system. Specifically, tracking is performed by a number of Kalman filters initiated for each new person appearing in a motion sequence. This permits tracking of multiple skeletons and reduces the frequency that computationally expensive pose estimation has to be run, enabling online pose tracking. The module tracks for $N$ frames while the pose estimates are calculated for frame $N + 1$. This also results in increased consistency of person identification and reduced inaccuracies due to missing joint locations and inversion of left-and right-side joints.*

## 1. Introduction

Mathematical methods for integrating data and dynamical models have been used for a long time in several applications. This discipline is most commonly associated with weather forecasting, where it is referred to as Data Assimilation [3]. We propose that these methods can be leveraged in less traditional fields, in particular in fast-developing, data-rich machine learning applications.

Data assimilation methods are designed to be able to integrate data at irregular intervals, from multiple data sources of varying precision and accuracy. The data is used to adjust parameters of an underlying dynamical model, to best fit the data and make reliable predictions about future data points. Human motion tracking is a new, exciting field to apply these methods.

In machine learning, we often have scenarios that could benefit from information about an underlying dynamical system. This is particularly true in any instance where sequential data are treated as separate or individual cases: for example, whenever data is extracted from a sequence of images, there is necessarily an underlying time-based dynamical system. In addition to this, connected points or parts are often considered individually and then reassembled to form a body after they have been identified [5], instead of taking into account from the start that they are part of a connected body. We strongly believe that there is a wealth of very relevant data, that should be leveraged to improve accuracy, consistency and computation time.

An application where both temporal and kinematic conditions exist is in human motion tracking. The human body has several kinematic constraints. Firstly, the skeleton creates a clear joint hierarchy, and each joint has different limits and degrees of freedom. Secondly, our motion follows certain patterns (for example, our motion tends to be smooth [28]) that we can leverage to predict how the keypoints of a human body are moving in time. In fact, there are a number of sources for additional information we can include to make up a general 'model' of human motion, which can be leveraged to extract a specific motion sequence from images.

In the proposed module, we have a system where different data sources are trusted to varying degrees, and information arrives at unknown intervals. This idea can be applied in combining camera sources, linking disparate data sources, such as motion sensors and cameras, or in changing the dependence on different measurements to match changing confidence. Data assimilation methods are potentially transformative here due to their decades-long use for fitting models to data in other fields.

We present an application of the proposed technique, creating a Kalman filter-based tracking algorithm for human motion estimation methods. In the literature, these methods are often interchangeably called *pose tracking* or *pose estimation* algorithms. By *pose estimation*, we mean methods that will identify poses in images or videos, but where each image or frame is treated independently. Often methods of this kind do not have a consistent person identification

Figure 1. Here, we illustrate three common shortcomings of pose tracking methods. The figure shows images from the middle section of a motion sequence from the PoseTrack 2018 validation dataset. The top row shows the keypoint annotations given by LightTrack, the middle row shows this data once it has been filtered by our proposed method, and the bottom row shows the ground truth. We have highlighted first two quite visible people that are not tracked (yellow), and second an instance where the ankle joints of two skeletons have been swapped (red). Third, we show a point where a figure in the forefront has lost an easily identified keypoint, despite it being tracked in the previous frames (light blue). All three issues (and others) are resolved after applying our filters.

throughout a video. Instead, *pose tracking* includes any method that attempts to maintain this identification – it is not just important to identify all poses in a particular frame, but to be able to track the full sequence of motion of any particular person throughout a sequence of images.

The main contributions of this paper are the following:

1. We provide a practical example of how Data Assimilation and Machine Learning methods can be combined and used in a complementary manner.
2. We implement a Kalman filter-based tracking module that can be applied to any human motion estimation algorithm, decreasing the average run time per frame and increasing the consistency of joint position estimation.
3. We perform a noise analysis on three motion estimation systems to obtain better measurement and noise covariance matrices for the filtering module.
4. We implement an identification algorithm to improve consistency of labelling of skeletons between frames.

## 2. Related works

The computer vision community has become very successful at recovering human pose from single images [21, 26]. The focus is now moving towards multi-person pose tracking [22, 31], where a number of people are tracked consistently over several frames, possibly maintaining the identification of the same person throughout the sequence. For a relatively recent review of other methods of human motion capture, we recommend [30].

As shown in Figure 1, most pose tracking methods suffer from a number of common limitations: for example, missing joints or losing them between frames, missing background people, or those who are partially occluded, and inversion of keypoints between left and right sides or between different skeletons.

### 2.1. 2D Methods

OpenPose is a collection of work (including from [5], [25] and [29]) that provides 2D human body, hand, foot and face multi-person pose estimation, as well as 3D joint locations for single person pose estimation. The method involves comparing the location of joints with identified *part affinity fields* to construct skeletons. Though it can also perform multi-person motion estimation from video clips, individual frames are treated separately. Openpose has a beta 'tracking' module that relies on optical flow, thus linking the sequence of images, but this will only track a single person.

A second method, which uses recurrent Spatio-Temporal Affinity Fields [24] (we will refer to this method as STAF), builds on the OpenPose algorithm to move towards consistent tracking over a motion sequence, rather than a single frame. Instead of just using the part affinity fields from OpenPose, the STAF algorithm include *spatio-temporal affinity fields*.

LightTrack [22] is the top-performing, publicly available method from the PoseTrack 2018 dataset leader board. The framework also performs top-down human pose tracking,

identifying person locations first and then using an inexpensive graphical representation of the human skeleton for pose matching. The authors discuss ID-matching in detail, including also spatial consistency metrics in their skeleton-matching module.

AlphaPose [9] is another algorithm that performs *regional* multi-person pose estimation. The framework effectively has two stages: first, a bounding box is constructed for any region that seems to contain a person, and then in the second stage the joint locations are identified.

Note that AlphaPose, LightTrack and STAF perform *pose tracking*: these methods will already track the same person over several frames and can do this for multiple people.

### 2.2. 3D Methods with Spatio-Temporal Information

We also mention the 3D pose estimation and tracking methods [2, 6, 15, 32], relevant for their inclusion of spatio-temporal extract 3D poses from 2D images.

In [32] the authors add a geometric constraint induced loss to identify the correct 3D interpretation of 2D joint predictions. Similarly, [2] include a temporal error to compensate for errors of the 2D keypoint detector at any particular frame. However, this method makes use of complete sequences to make a 'bundle adjustment' of all poses identified for a particular person. In [15] the authors present a self-supervised method that utilises epipolar geometry to combine multiple 2D views to recover 3D poses. Chang *et al*. [6] use a progressive particle filter to recover a model of the human body from images, beginning from the torso and building towards the lower extremities.

In [18], the authors use an LSTM to include temporal information in their pose tracking method to help maintain geometric consistency between frames. Another recent method [13] goes beyond pose tracking and instead focuses on motion prediction two seconds or more in the future using a spatio-temporal tensor of 3D skeleton coordinates.

### 2.3. Pose Refinement Methods

Finally, methods such as [10, 19] focus instead on *pose refinement*. This is conceptually similar to the proposed module, in that these methods all take as input only images and a given pose estimate from any method. Fieraru *et al*. [10] propose a post-processing network of this kind, using a pose refinement method to give refined heatmaps and offsets from the originally identified keypoints. In [19] the authors also perform an interesting analysis of the types of errors made by different pose tracking and estimation methods, and find that the top-performing methods have similar error distributions. The pose refinement in this method is therefore fine-tuned to the error distribution of the original pose estimation method used.

### 2.4. Kalman Filters for Motion Tracking

In human motion tracking, Kalman filters are normally associated with Inertial Measurement Unit (IMU) data fusion [14, 17, 27]. Here the noisy accelerometer, gyroscope and magnetometer data are fused to give a more accurate data point at any time instance. However, in camera-based tracking methods, they can be used to overcome the usual constraints arising from image sequences. For example, occlusion, self-occlusion and distance from the camera are all particular hurdles for recovering accurate human pose. Kalman filters have long been used for object tracking in images to overcome these issues ([7] presents a summary of Kalman filter in object tracking as far back as 2005). In joint tracking, there is often fluctuation in joint location simply because the true location corresponds to an area (rather than a point) of the image. Since Kalman filters, in general, lend themselves to de-noising and include some information about the underlying structure of a system, we argue that they can be useful in processing visual joint-tracking data.

An early (2008) application of Kalman filtering to human motion tracking was to extend tracking over hidden skeleton segments [12]. In this paper, the authors combine an extended Kalman filter (EKF) for rigid body pose with kinematic constraints for a system of connected rigid bodies (e.g. the ankle complex). In marker-based motion tracking, the impact soft-tissue artefact on skin-marker data can be significant when requiring high-precision data. Bonnet *et al*. in [4] use multi-body kinematics optimisation and EKFs to overcome soft-tissue artefact and thus estimate leg kinematics.

Kalman filters are also useful when combining multiple images. In [20], the authors combine observations from multiple cameras to obtain 3D joint positions of a 25-joint body model. Instead, in [16], views from multiple cameras are combined to track multiple people in the same environment. Multiple cameras are used to avoid inaccuracies stemming from occlusions when using only a single camera. The Kalman filters are then crucial in overcoming the data fusion problem. This is a different application area to the one in the proposed method, but more similar to how Kalman Filters are used in IMUs, as the aim is to combine sensor measurements to generate a weighted fused measurement directly.

## 3. Methodology

The proposed tracking module is designed to take steps toward combining data assimilation and machine learning methods. Kalman filters are structured to include some information about the underlying structure of the data, and leverage this to make predictions for future time steps. Without new pose information, new people coming into view will not be identified. On the other hand, when not tracking, we

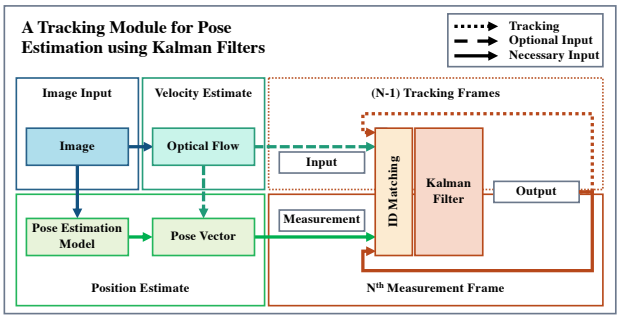


Figure 2. Structure of the proposed tracking module

may lose track of different joints or relabel them between frames.

In this section, we outline the methodology of the proposed tracking module (Figure 2). The module does not affect the (underlying) human motion estimation algorithm, but runs alongside it. From the perspective of the Kalman filtering, the pose estimation algorithm acts as a sensor providing regular data points. Thus we treat the tracked pose as a noisy measurement, obtained from the algorithm every $N$ frames. When the algorithm is run, we both predict the next state of each skeleton using the Kalman filters, and correct these predictions using the pose measurement. The proposed module also makes use of an Optical Flow algorithm, and merges predictions made from this data into the predictions made at every time step.

### 3.1. Design of New Tracking Module

To implement the new tracking module, there are three key contributions: first, the design of the system structure to include all the available information and to allow for flexibility in choice of dataset, Kalman filter and optical flow algorithm; second, the inclusion of appropriate Kalman filters; and third, the addition of covariance matrices calculated using each of three pose estimation methods on the PoseTrack 2018 dataset [1].

We will make use of different sources of data: we know that our data follows a time series, and thus successive frames will give us information about the motion time series (though this will not be true when the camera view changes). We need to leverage this for the individual keypoints (joint locations), rather than the whole body, as the pose can change between time steps. We can also make some additional assumptions about how the identified keypoints will move in time, for example by using a constant acceleration or velocity model (in fact, in [28], [11] the authors suggest that human motion tends to follow profiles that minimise jerk, the derivative of acceleration). In terms of kinematic information, the structure of the skeleton, including joint hierarchy, depends on the choice of dataset or pose estimation algorithm. However, this does not affect the inner workings of the module, other than the requirement of the model to know the number of joints it needs to track.

We identified three open-source methods to work with that performed relatively consistently on the PoseTrack 2018 dataset [1], namely OpenPose [5], Spatio-Temporal Affinity Fields (STAF, [24]) and LightTrack [22]. In addition to these pose estimation functions, we make use of the iterative Lucas-Kanade optical flow method implemented in OpenCV (though this could be substituted with any other similar method). We also make use of an identification module (Section 3.5) to compare pose IDs obtained every time the pose tracking algorithm is run with the IDs already identified.

### 3.2. The Kalman Filter

The general time update equations for a discrete Kalman filter are

$$\hat{x}^-_{k+1} = \mathbf{A}_k \hat{x}_k + \mathbf{B}_k u_k + w_k \tag{1}$$

$$\mathbf{P}^-_{k+1} = \mathbf{A}_k \mathbf{P}_k \mathbf{A}^T_k + \mathbf{Q}_k \tag{2}$$

where the superscripts $x^-$ and $\hat{x}$ denote an *a priori* (as opposed to *a posteriori*) value and an estimated value respectively, $\mathbf{A}_k$ is the $(n \times n)$ state transition matrix, $\mathbf{B}$ is the $(n \times l)$ input matrix, $w_k$ is the process noise, $\mathbf{P}_{k+1}$ is the $(n \times n)$ estimate error covariance, and $\mathbf{Q}_k$ is the $(n \times n)$ process covariance matrix. The values $n$ and $l$ correspond to the state and input dimensions respectively. To these equations, we add the filter measurement update equations:

$$\mathbf{K}_k = \mathbf{P}^-_k \mathbf{H}^T_k (\mathbf{H}_k \mathbf{P}^-_k \mathbf{H}^T_k + \mathbf{R}_k)^{-1} \tag{3}$$

$$\hat{x}_k = \hat{x}^-_k + \mathbf{K}(z_k - \mathbf{H}_k \hat{x}^-_k) \tag{4}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}^-_k \tag{5}$$

where $\mathbf{K}_k$ is the Kalman gain, $\mathbf{H}_k$ is the observation model, $\mathbf{R}_k$ is the observation noise (see Section 3.4 and $z_k$ is a measurement subject to (Gaussian) noise.

The Kalman filtering works so that at each timestep, a prediction of the next state is calculated, as well as the error covariance ahead. These are Equations (1) and (2). If a new measurement is received, we perform the correction step, using equations (3) to (5).

In this instance, we have (noisy) measurements every $N$ frames from the human pose estimator, and (noisy) measurements every frame from the LK optical flow algorithm. The state vector $x$ is a vector of $M$ $x$- and $y$- joint positions. Depending on how we choose to model the motion kinematics, we can include an input $u$. The input $u$ can be thought of as an unfiltered measurement – there is no tracking over time of this variable, we give an input and this is considered correct at this time instance but unrelated to other time steps.

For example, we could design a simple system where the state $x$ is a measurement of position only, and at each frame, we input a velocity estimated from the optical flow as our input $u$. Alternatively, we could have a state vector of position and velocity (of the form $\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]$), with no input $u$. This latter model would give a system operating under a *constant velocity assumption* (plus noise). The different models are discussed in more detail in the next section on System Kinematics.

### 3.3. System Kinematics

We track the position (in $x$- and $y$-pixel positions) and velocity (using optical flow in the $x$- and $y$-directions) of each joint, so that each filter tracks $2M$ measurements (since we are tracking $M$ joint skeletons). To model the system kinematics, using the matrices $\mathbf{A}_k$ and $\mathbf{B}_k$, we assume that each joint is following a motion path governed by the usual equations of motion

$$x_t = x_{t-1} + dt\, u_{t-1} + \frac{1}{2} dt^2 a_{t-1} \tag{6}$$

If we define $x = [x_0, y_0, ...]^T$, so that we have a vector of the $x$- and $y$-positions for the $M$ joints, and consider the first model described earlier with velocity as input, we have a matrix equation that looks like

$$\hat{x}_{k+1}^- = I^{n\times n}\hat{x}_k + \Delta T^{m\times m} u_k + w_k \tag{7}$$

where $dt$ is the timestep between frames and $\Delta T^{m\times m} = dtI^{m\times m}$ is a matrix of timesteps. Note that the dimensions of the state vector, $n$, and the dimensions of the input vector, $m$, are not necessarily equal. This depends on how many values we are measuring. The pixel tracking for the joints gives us an estimate for joint velocity. Both the position measurements and pixel tracking processes are subject to error noise.

In the second case, with both position and velocity forming the state vector $[x, u]$, our equation (dropping dimension superscripts for clarity) becomes

$$\hat{x}_{k+1}^- = \begin{bmatrix} I & \Delta T \\ \mathbf{0} & I \end{bmatrix} \hat{x}_k + w_k \tag{8}$$

and our state vector has dimensions $[2n \times 1]$.

The equations internal to the Kalman filter define how the filters perform for the different keypoints. Firstly, we note in Table 2 that the simple linear model does not capture the motion well. This is because the equations imply too static a model to describe motion well, and there is a lag when motion changes direction.

In general, in human motion analysis, the keypoints at the wrists and ankles will move more and faster, and change direction more frequently, than the keypoints on the main body or head. Here, we see this reflected in the performance shown for the extremity keypoints in the constant velocity model relative to the model with velocity as input, and that with an acceleration component. When velocity is given as an input, there is no lag when changing direction, but the measurement is also subject to unfiltered error from the additional measurement source.

In Table 2, we see that though we are running the tracking algorithm every frame, the Kalman filter does introduce over-filtering in the more simple models. This is particularly evident in the wrists and ankles, which are the body joints that tend to move more. Including the acceleration covariance means the model captures most of the movement again ($11\%$ improvement for LightTrack). Including velocity as input performs better than tracking velocity in the state vector, particularly for STAF ($19\%$ improvement). This is likely because a constant velocity model does not capture changes in movement particularly well.

### 3.4. Noise Matrices

Three sources of error and noise are addressed in any Kalman filter; the first two can be used to change the behaviour of the filter, the third gives us information about the prediction made by the filter:

1. Measurement Noise: (with covariance $\mathbf{R}$) respectively for position and velocity, the noise of the joint positions obtained from the pose estimation algorithm, and the noise of velocities obtained from the pixel tracking algorithm (detailed in Section 3.4.1).
2. Process Noise: (with covariance $\mathbf{Q}$) a measure of how we expect the human motion we are tracking to deviate from our model – in effect, how much we expect to deviate from our equations (detailed in Section 3.4.2).
3. Error Covariance: ($\mathbf{P}$) a measure of the estimated accuracy of the state estimate, as in Equation (5).

#### 3.4.1 Measurement Noise

To address the measurement noise matrix, we run each pose estimation algorithm on the PoseTrack 2018 [1] train dataset to compare how the measurements differ from the ground truth labels. The poses we obtain for each frame are matched with the poses given in the ground truth keypoint labels, and the differences in keypoint measurements are calculated. The covariance of these differences is used as the measurement noise matrix. When also considering velocity in the state vector, we add the covariance similarly calculated from the values returned by the optical flow algorithm.

#### 3.4.2 Process Noise

If we consider the two systems mentioned in Section 3.3, neither a constant velocity or constant acceleration assumption totally describes the motion we might expect from a joint moving through space. To capture the changing acceleration,

we can make use of the process noise matrix. If we define the state vector to be as in the second case (see Equation (8)) so that $\left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]$, and our state equation to be:

$$\hat{x}_{k+1}^{-} = \begin{bmatrix} I & \Delta T \\ \mathbf{0} & I \end{bmatrix} \hat{x}_k + \begin{bmatrix} \frac{1}{2}\Delta T^2 \\ \Delta T \end{bmatrix} \hat{a}_k \tag{9}$$

where if we let

$$\mathbf{G} = \begin{bmatrix} \frac{1}{2}\Delta T^2 \\ \Delta T \end{bmatrix} \tag{10}$$

we can define the noise term $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q})$ to have a structure that takes into account the acceleration of the system, with

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{4}\Delta T^4 & \frac{1}{2}\Delta T^3 \\ \frac{1}{2}\Delta T^3 & \Delta T^2 \end{bmatrix}. \tag{11}$$

In this way, we can leverage the process covariance matrix to include the acceleration into the system kinematics.

### 3.5. ID Matching

We initialise a new Kalman filter for each skeleton that is tracked in the motion clip. This means that the system is reliant on the consistent identification (or labelling) of the skeletons. In several pose tracking algorithms, each frame is considered individually, meaning the skeleton IDs frequently change between frames. Over the $N$ tracked frames, there is no issue of changing IDs as the tracking is performed for each skeleton individually, but when the tracking algorithm is run, a comparison must be made to ensure the IDs are consistent as often as possible.

To tackle this issue, we integrate an identification module. This module works by comparing the number of inlier / outlier joints (within a tolerance) of a skeleton relative to the skeletons in the previous frame. Any joints that are within this threshold contribute to a score for the skeleton pair. There is also a score bonus if the two poses have similar scales and are close to each other. In the case of a mismatch, the IDs are reassigned in the way that maximises the total score. A new ID is assigned to any new skeletons, though as far as possible we aim to maintain the IDs of established Kalman filters. Specifically, as far as possible, we avoid defining new pose IDs, unless a new pose is very far from any previously recognised skeleton.[1]

## 4. Experiments and Results

### 4.1. The Datasets

To test the new tracking module, we require datasets with ground truth joint annotations in real-world images in a range of lighting conditions, with a variety of different situations. We particularly wanted to test the algorithm in sequences with several people, as this is the only way to test the identification module, and where treating frames individually might normally lead to joint positions being inconsistently tracked over several frames. To the best of our knowledge, there are very few annotated human motion datasets, but Posetrack 2017 and Posetrack 2018 [1] fulfilled these requirements. In addition, [8] provide some annotated video sequences taken with a mobile phone.

[1]Note that though this was a necessary addition, it was not the main focus of this work.

| Model | MOTA Head | MOTA Shou | MOTA Elb | MOTA Wri | MOTA Hip | MOTA Knee | MOTA Ankl | MOTA Total |
|---|---|---|---|---|---|---|---|---|
| Acceleration Model – Our ID matching | | | | | | | | |
| STAF | 35.5 | 59.4 | 53.7 | 39.8 | 57.3 | 43.1 | 46.6 | 47.8 |
| LT | 63.2 | 64.7 | 63.4 | 57.3 | 54.6 | 57.9 | 59.1 | 60.0 |
| GT | 86.8 | 89.3 | 80.8 | 88.6 | 88.5 | 85.1 | 82.8 | 84.7 |
| Acceleration Model – Model's own ID matching / Ground Truth IDs | | | | | | | | |
| STAF | 32.2 | 55.6 | 50.3 | 36.6 | 54.4 | 40.9 | 42.8 | 44.6 |
| LT | 61.3 | 64.5 | 62.5 | 58.2 | 54.4 | 57.2 | 59.6 | 59.7 |
| GT | 92.7 | 93.3 | 94.0 | 93.7 | 94.0 | 94.0 | 93.5 | 93.5 |

Table 1. PoseTrack 2018 MOT metrics (tracking 0 frames) to show performance of our ID matching method. First, we compare our ID matching method to the methods used by the original models, and show that performance is comparable. Second, we compare performance when using ground truth keypoint annotations with our ID matching compared to with the ground truth person IDs. OpenPose does not have its own ID matching. (LT = LightTrack, STAF = Spatio-Temporal Affinity Fields, GT = Ground Truth)

### 4.2. Benchmarking

We use the multi-person multi-frame benchmark for Posetrack 2018. The evaluation is based on a number of parameters, namely 1) The correct identification of the number of people in the scene, 2) The consistent identification of different people in the scene, and 3) The identification of joint locations for all 25 joints for each person, compared with the ground truth labels of the images.

We consider two benchmarks given for the PoseTrack dataset, both of which are given by the 'poseval' evaluation module: The first is the Average Precision (AP) metric for per-frame multi-person pose estimation. The measure is proposed in [23] and evaluates the precision of joint detections in each frame. The pose detections are greedily assigned to the ground truth (GT) annotations, and duplicate or unassigned poses are counted as false positives. The metric is given for each body part and as a total over the skeleton.

The second is a Multiple Object Tracking (MOT) metric for the evaluation of pose tracking over several frames or a video sequence. This considers predicted poses and their tracking IDs and evaluates the distances between predicted and GT joint locations both on a global basis (*i.e.* not considering tracking IDs), and taking the tracking IDs into account. Multiple Object Tracker Accuracy (MOTA), Multiple Object Tracker Precision (MOTP), Precision, and Recall metrics are computed across all joints, and MOTA is given for each

| Model | MOTA Head | MOTA Shou | MOTA Elb | MOTA Wri | MOTA Hip | MOTA Knee | MOTA Ankl | MOTA Total | MOTP Total | Prec Total | Rec Total | AP Head | AP Shou | AP Elb | AP Wri | AP Hip | AP Knee | AP Ankl | AP Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Original Algorithm | | | | | | | | | | | | | | | | | | | |
| OpenPose | -77.9 | 6.2 | -26.4 | -53.8 | -10.1 | -34.3 | -62.3 | -39.7 | 59.5 | 50.8 | 66.0 | 23.4 | 71.2 | 58.1 | 45.9 | 57.2 | 54.1 | 45.3 | 48.9 |
| STAF | 39.5 | 67.7 | 54.4 | 44.4 | 55.9 | 50.0 | 42.3 | 49.8 | 60.3 | 76.0 | 67.8 | 44.6 | 75.2 | 68.9 | 57.6 | 65.9 | 63.1 | 56.4 | 60.5 |
| LightTrack | 67.7 | 72.6 | 67.3 | 57.8 | 63.5 | 63.8 | 57.7 | 64.6 | 85.3 | 88.0 | 76.0 | 75.1 | 80.1 | 75.1 | 66.4 | 71.5 | 71.2 | 66.1 | 72.4 |
| Simple Linear Model | | | | | | | | | | | | | | | | | | | |
| OpenPose | -7.0 | -8.9 | -9.8 | -8.9 | -9.2 | -9.0 | -8.5 | -8.7 | 56.2 | 44.0 | 36.9 | 15.4 | 24.4 | 18.9 | 13.0 | 19.4 | 16.6 | 13.5 | 17.2 |
| STAF | 28.2 | 50.7 | 37.3 | 27.2 | 37.8 | 33.0 | 27.0 | 34.0 | 56.3 | 62.3 | 50.3 | 33.0 | 56.4 | 49.8 | 39.5 | 47.2 | 44.9 | 39.6 | 43.6 |
| LightTrack | 57.3 | 59.6 | 50.2 | 51.9 | 52.5 | 52.1 | 55.6 | 54.2 | 57.3 | 68.2 | 65.3 | 63.2 | 65.8 | 61.0 | 58.9 | 59.7 | 60.1 | 58.3 | 61.0 |
| Velocity Input Model | | | | | | | | | | | | | | | | | | | |
| OpenPose | 16.3 | 21.8 | 11.3 | 8.1 | 25.2 | 13.3 | 7.3 | 14.7 | 57.1 | 60.6 | 58.1 | 34.6 | 74.1 | 54.3 | 36.2 | 65.7 | 46.1 | 38.0 | 48.8 |
| STAF | 33.7 | 57.2 | 55.9 | 35.8 | 61.3 | 40.1 | 43.6 | 46.8 | 62.5 | 71.3 | 63.2 | 43.5 | 69.1 | 63.8 | 45.2 | 57.6 | 53.2 | 44.1 | 53.8 |
| LightTrack | 62.6 | 63.4 | 58.3 | 52.8 | 54.3 | 53.1 | 54.8 | 57.0 | 63.2 | 69.2 | 72.4 | 66.7 | 69.3 | 67.2 | 61.5 | 65.9 | 66.8 | 60.3 | 65.4 |
| Constant Velocity Model | | | | | | | | | | | | | | | | | | | |
| OpenPose | 14.6 | 16.3 | 7.3 | 3.5 | 16.2 | 15.3 | 1.0 | 10.6 | 46.3 | 58.5 | 55.3 | 30.7 | 64.9 | 43.6 | 27.7 | 57.2 | 38.5 | 30.9 | 41.2 |
| STAF | 30.1 | 53.2 | 44.0 | 29.3 | 57.9 | 33.7 | 27.4 | 39.4 | 57.5 | 70.1 | 63.7 | 37.6 | 60.2 | 52.7 | 33.1 | 52.2 | 40.3 | 38.4 | 44.9 |
| LightTrack | 59.2 | 62.1 | 53.8 | 54.3 | 55.7 | 53.2 | 57.4 | 56.5 | 62.6 | 68.4 | 73.8 | 65.3 | 69.7 | 65.4 | 59.8 | 67.6 | 64.7 | 59.1 | 64.5 |
| Acceleration Model | | | | | | | | | | | | | | | | | | | |
| OpenPose | 24.1 | 21.2 | 19.4 | 13.7 | 28.0 | 23.2 | 17.4 | 21.0 | 66.3 | 52.5 | 65.9 | 30.7 | 63.8 | 44.0 | 29.1 | 57.4 | 38.1 | 29.2 | 41.0 |
| STAF | 35.5 | 59.4 | 53.7 | 39.8 | 57.3 | 43.1 | 46.6 | 47.8 | 62.5 | 73.3 | 67.1 | 44.7 | 67.3 | 59.8 | 47.9 | 53.1 | 49.3 | 46.9 | 52.7 |
| LightTrack | 63.2 | 64.7 | 63.4 | 57.3 | 54.6 | 57.9 | 59.1 | 60.0 | 72.3 | 73.4 | 78.2 | 70.2 | 76.8 | 69.4 | 60.7 | 68.7 | 68.1 | 60.5 | 67.8 |
| Particle Filter | | | | | | | | | | | | | | | | | | | |
| OpenPose | 17.2 | 25.3 | 10.5 | 7.5 | 23.4 | 15.7 | 6.2 | 15.1 | 56.5 | 62.0 | 57.3 | 19.8 | 31.8 | 26.9 | 19.8 | 26.5 | 24.0 | 20.4 | 23.9 |
| STAF | 36.8 | 64.3 | 50.3 | 39.2 | 52.0 | 46.3 | 40.1 | 46.3 | 60.2 | 76.4 | 65.3 | 42.9 | 72.9 | 66.3 | 54.3 | 63.3 | 60.7 | 54.5 | 58.2 |
| LightTrack | 33.3 | 36.5 | 32.1 | 26.5 | 31.5 | 30.5 | 27.5 | 31.3 | 70.4 | 86.3 | 41.9 | 40.0 | 43.5 | 39.4 | 34.6 | 38.8 | 37.6 | 34.7 | 38.5 |

Table 2. PoseTrack 2018 MOT and AP metrics (tracking 0 frames) for four different Kalman filter models, compared to the performance of the original algorithms. We notice some over-filtering, particularly in the wrists and ankles, but also significant improvements in OpenPose, which is not designed to track poses over several frames. The measurements are better in the velocity input and acceleration models.

body joint class as well.

## 4.3. Results

We compare the methods OpenPose [5], LightTrack [22] and STAF [24]. Note that LightTrack is currently the best performing publicly available system on the PoseTrack 2018 leaderboard. For each of the methods, we test four Kalman filters that describe different kinematic models:

1. Simple Linear Model: State vector contains position only $[x]$, no input.
2. Velocity Input Model: State vector contains position only $[x]$, velocity $[v]$ as input.
3. Constant Velocity Model: State vector contains position and velocity $[x, v]$, no input.
4. Acceleration Model: State vector contains position and velocity $[x, v]$, no input, process covariance matrix $\mathbf{Q}$ has the form of Equation (11).

We also test how the filters perform as we increase the number of frames $N$ over which we track the poses. For each of these, we report both AP and MOT metrics, noting however that the Kalman filters will provide a benefit primarily in the MOT metric.

### 4.3.1 Original Algorithm Results

In Table 2 we report the results for the benchmarks mentioned above for the three chosen pose estimation methods on the PoseTrack 2018 validation dataset. OpenPose performs relatively well on single frame pose estimation, but in our experience suffers in the tracking metrics because the tracking ID assigned to each pose is not consistent frame to frame. We note that with AlphaPose we had difficulties in replicating the results reported by the authors, and thus have included these results in the supplementary material.

### 4.3.2 Length of Tracking Sequence

We also investigate how the value of $N$, the number of frames tracked, affects the performance of the model. In Figure 3, we see MOTA total scores and computation speed when all three methods are run alone, and when they are run with tracking over $N = [0, 2, 5, 10]$ frames. For the tracking here we use the Acceleration model, as this performed best overall for both AP and MOTA metrics. The complete results are shown in the Supplementary Material.

We can see that all three methods reach real-time tracking speed (30 FPS), at $N = 2, 5$ and 10 for OpenPose, STAF

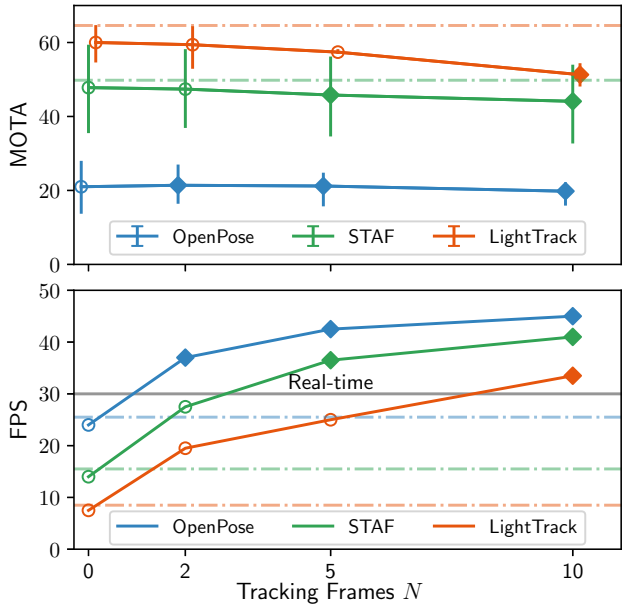


Figure 3. MOTA Totals and computational speed (FPS) with $N = [0, 2, 5, 10]$ for each method. The diamond markers show where each method achieves a real-time speed of 30FPS or higher. The horizontal lines show performance for the original methods (we have excluded the original MOTA performance of OpenPose as it is a pose estimation method).

and LightTrack respectively. For all three methods, there is a slight loss in MOTA performance with increasing $N$ ($8\%$ and $14.5\%$ for STAF and LightTrack respectively). This means that our module allows the use of some of these powerful pose tracking methods in real time, without sacrificing too much accuracy. We also transform a pose estimation method, OpenPose, into a pose tracking method.

For pose estimation methods that frequently change tracking ID of the different skeletons, or that see joints frequently disappear from the image, the primary benefit of any tracking is the added consistency. However, the filter is also not able to capture all of the intricacies of motion when tracking over a large number of frames.

## 5. Discussion

Kalman filters are powerful tools for dealing with measurement uncertainties and data inconsistencies. Here, we have presented a system that is to an extent reliant on the data points obtained from the pose tracking algorithm, but manages to perform close to the original algorithm, despite not running any pose inference on the $N - 1$ tracked images. The system includes equations of motion to make sense of the measurements obtained and because of the filtering, finds a balance between 'trusting' the data from the algorithm (*i.e.* the best possible measurement we can obtain), and maintaining the consistency that the filtering methods provide.

To an extent, the balance between these differs depending on the number of frames, $N$, that we track between measurements, the original pose estimation method, and the system kinematics we have tried to describe with the Kalman filter equations. All the existing methods tried were sped up to over 30 FPS, which is an acceptable speed for real-time pose tracking.

## 6. Conclusions

We present a module that leverages both Data Assimilation and Machine Learning methods for human motion tracking. We can transform methods that perform pose estimation to *pose tracking* methods, or speed up existing *pose tracking* methods to real-time speeds. The different Kalman filters show how simple but considered changes to the internal structure of the system can tweak performance in different ways. In addition, these filters are computationally inexpensive, and can easily be run alongside a more expensive network to increase the speed at which a motion sequence can be analysed.

Data Assimilation and Machine Learning methods are often similar enough in structure to complement each other. The method proposed here does not rely on any particularity of human motion estimation to be successful; thus, there is no reason why it could not be applied to other areas. Several of the inaccuracies we encounter in human motion tracking are actually common in computer vision applications and are simply due to inconsistencies in the interpretation of images. Data Assimilation is a field founded on resolving and working with inconsistencies and uncertainties in datasets and therefore is an area that can make a valuable contribution to the computer vision field.

In future works, it would be worth considering how a neural network based model can be incorporated into an Extended Kalman Filter. In this work, we have used our filters to complement the pose estimation methods, but a more fundamental integration would bring to light how a neural network compares to historical methods. Of particular interest is whether a data-based approach can capture nonlinearities better than existing methods.

Another avenue to consider would be to use the Kalman filters presented in this method as an ensemble, to leverage the strengths of the different models. In particular, moving weights depending on the velocity of the keypoints would allow more or less stable models to be used to accurately describe the motion.

## Acknowledgements

This research was supported in part by Doctoral award number 1859268 of the Engineering and Physical Sciences Research Council (EPSRC, UKRI) to Caterina Buizza.

## References


[1] M. Andriluka, U. Iqbal, E. Ensafutdinov, L. Pishchulin, A. Milan, J. Gall, and S. B. PoseTrack: A benchmark for human pose estimation and tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5167–5176, 2018.

[2] A. Arnab, C. Doersch, and A. Zisserman. Exploiting temporal context for 3d human pose estimation in the wild. In *Computer Vision & Pattern Recognition*, pages 3395–3404, 2019.

[3] M. Bocquet. Introduction to the principles and methods of data assimilation in geosciences. Technical report, Ecole des Ponts ParisTech, 2014.

[4] V. Bonnet, V. Richard, V. Camomilla, G. Venture, A. Cappozzo, and R. Dumas. Joint kinematics estimation using a multi-body kinematics optimisation and an extended Kalman filter, and embedding a soft tissue artefact model. *Journal of Biomechanics*, 62:148–155, 2017.

[5] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[6] I. C. Chang and S. Y. Lin. 3D human motion tracking based on a progressive particle filter. *Pattern Recognition*, 43(10):3621–3635, 2010.

[7] E. Cuevas, D. Zaldivar, and R. Rojas. Kalman filter for vision tracking. Technical report, Freie Universität Berlin, 2005.

[8] A. Ess, B. Leibe, K. Schindler, and L. van Gool. A mobile vision system for robust multi-person tracking. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2008.

[9] H. S. Fang, S. Xie, Y. W. Tai, and C. Lu. RMPE: Regional Multi-person Pose Estimation. *IEEE International Conference on Computer Vision*, pages 2353–2362, 2017.

[10] M. Fieraru, A. Khoreva, L. Pishchulin, and B. Schiele. Learning to refine human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 318–327, 2018.

[11] T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *The Journal of Neuroscience*, 5(7):1688–1703, 1985.

[12] K. Halvorsen, C. Johnston, W. Back, V. Stokes, and H. Lanshammar. Tracking the Motion of Hidden Segments Using Kinematic Constraints and Kalman Filtering. *Journal of Biomechanical Engineering*, 130(1):011012, 2008.

[13] A. Hernandez Ruiz, J. Gall, and F. Moreno-Noguer. Human motion prediction via spatio-temporal inpainting. In *IEEE International Conference on Computer Vision*, pages 7134–7143, 2019.

[14] V. Joukov, V. Bonnet, M. Karg, G. Venture, and D. Kulić. Rhythmic EKF for pose estimation during gait. *IEEE-RAS International Conference on Humanoid Robots*, 2015-Decem:1167–1172, 2015.

[15] M. Kocabas, S. Karagoz, and E. Akbas. Self-Supervised Learning of 3D Human Pose using Multi-view Geometry. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1077–1086, 2019.

[16] G. Lee, Y. Lim, and J. Choi. Kalman Filter Based Multiple Human Tracking Method in Auto-aligned Multi-Camera Environment. In *IEEE International Conference on Ubiquitous Robots*, pages 19–23, 2018.

[17] G. Ligorio and A. M. Sabatini. A novel kalman filter for human motion tracking with an inertial-based dynamic inclinometer. *IEEE Transactions on Biomedical Engineering*, 62(8):2033–2043, 2015.

[18] Y. Luo, J. Ren, Z. Wang, W. Sun, J. Pan, J. Liu, J. Pang, and L. Lin. LSTM Pose Machines. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 5207–5215, 2018.

[19] G. Moon, J. Y. Chang, and K. M. Lee. PoseFix: Model-agnostic General Human Pose Refinement Network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7773–7781, 2019.

[20] S. Moon, Y. Park, D. W. Ko, and I. H. Suh. Multiple kinect sensor fusion for human skeleton tracking using Kalman filtering. *International Journal of Advanced Robotic Systems*, 13(2):1–10, 2016.

[21] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis. Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nature Protocols*, 14:2152–2176, 2019.

[22] G. Ning and H. Huang. LightTrack: A Generic Framework for Online Top-Down Human Pose Tracking. *arXiv preprint arXiv:1905.02822*, 2019.

[23] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4929–4937, 2016.

[24] Y. Raaj, H. Idrees, G. Hidalgo, and Y. Sheikh. Efficient Online Multi-Person 2D Pose Tracking with Recurrent Spatio-Temporal Affinity Fields. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 4620–4628, 2019.

[25] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1145–1153, 2017.

[26] K. Sun, B. Xiao, D. Liu, and J. Wang. Deep High-Resolution Representation Learning for Human Pose Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019.

[27] A. Szczęsna and P. Pruszowski. Model-based extended quaternion Kalman filter to inertial orientation tracking of arbitrary kinematic chains. *SpringerPlus*, 5(1), 2016.

[28] P. Viviani and T. Flash. Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning. *Journal of Experimental Psychology: Human Perception and Performance*, 21(1):32–53, 1995.

[29] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[30] S. Xia, L. Gao, Y.-K. Lai, M.-Z. Yuan, and J. Chai. A Survey on Human Performance Capture and Animation. *Journal of Computer Science and Technology*, 32(3):536–554, 2017.

[31] X. Yun and E. R. Bachmann. Design, Implementation, and Experimental Results of a Quaternion-Based Kalman Filter for Human Body Motion Tracking. *IEEE Transactions on Robotics*, 22(6):1216–1227, 2006.

[32] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei. Towards 3D Human Pose Estimation in the Wild: A Weakly-Supervised Approach. In *IEEE International Conference on Computer Vision*, pages 398–407, 2017.