Long Short-Term Memory Kalman Filters:
Recurrent Neural Estimators for Pose Regularization

Huseyin Coskun[1], Felix Achilles[2], Robert DiPietro[3], Nassir Navab[1,3], Federico Tombari[1]

[1]Technische Universität München, [2]Ludwig-Maximilians-University of Munich,
[3]Johns Hopkins University

huseyin.coskun@tum.de, felix.achilles@med.lmu.de
rdipietro@gmail.com, navab@cs.tum.edu, tombari@in.tum.de

Abstract

One-shot pose estimation for tasks such as body joint localization, camera pose estimation, and object tracking are generally noisy, and temporal filters have been extensively used for regularization. One of the most widely-used methods is the Kalman filter, which is both extremely simple and general. However, Kalman filters require a motion model and measurement model to be specified a priori, which burdens the modeler and simultaneously demands that we use explicit models that are often only crude approximations of reality. For example, in the pose-estimation tasks mentioned above, it is common to use motion models that assume constant velocity or constant acceleration, and we believe that these simplified representations are severely inhibitive. In this work, we propose to instead learn rich, dynamic representations of the motion and noise models. In particular, we propose learning these models from data using long short-term memory, which allows representations that depend on all previous observations and all previous states. We evaluate our method using three of the most popular pose estimation tasks in computer vision, and in all cases we obtain state-of-the-art performance.

1. Introduction

Pose estimation from images is a recurring challenge in computer vision, for example for tasks such as camera pose estimation, body joint localization, and object tracking. Such tasks have recently benefited from learned models [16, 24, 4], but various problems persist when applying one-shot pose estimation to video data. In fact, disregarding temporal information can result in very noisy estimates and in the confusion of visually similar but spatially distinct image features, such as those that result from the left and right legs in the case of body joint localization. For this reason, temporal filters are a popular approach for improving the ac-
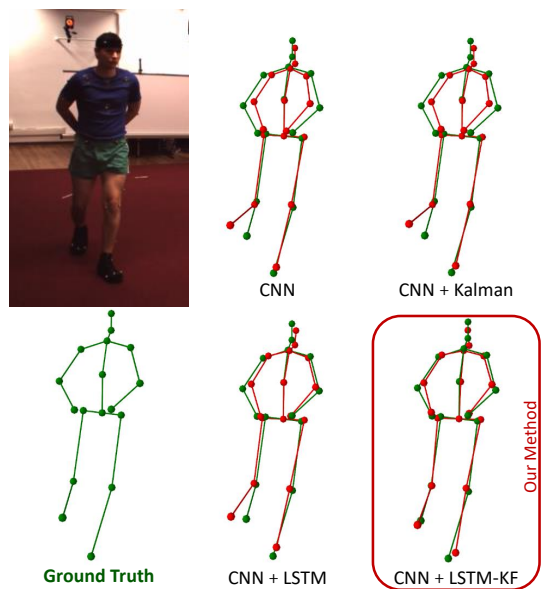

Figure 1. The proposed LSTM-KF approach builds on Kalman filters and LSTM networks to yield an improved temporal regularizer for common pose estimation tasks such as 3D body landmark localization from RGB images.

curacy of pose estimation. Among these methods, because of their simplicity and general applicability, Kalman filters (KF) [15] are an extremely widely-used choice. Moreover, the extended Kalman filter (EKF) [26] is capable of handling non linear systems for both the measurement and transition models.

However, in many tasks, these measurement and transition models cannot be specified a priori, and in these situations the application of Kalman filters is severely limited. In particular, in these in these tasks we must devise carefully tuned measurement and transition models, and even once devised they tend to be overly simplistic. For example, in the aforementioned computer vision tasks the trajectories of objects and body parts do not follow any simple

1

motion model. In such scenarios, Kalman filters are often applied under the assumptions of constant velocity or constant acceleration, which are clearly crude approximations to reality.

To overcome such limitations, attempts have been made to directly learn motion models from training data, for example with support vector machines (SVMs) [21] or with long short-term memory (LSTM) [18]. Learning motion models can alleviate the modeler from time-consuming Kalman filter selection and optimization and simultaneously enrich the underlying motion model. However, using learned motion models to enforce temporal consistency in pose estimation has to cope with the constraint that sufficient training data needs to be available in order to cover all possible motion paths of the tracked object.

In this work, we propose the LSTM Kalman filter (LSTM-KF), a new architecture which lets us learn the internals of the Kalman filter. In particular, we learn the motion model and all noise parameters of the Kalman filter, thus letting us gain the benefits of learning while letting us successfully train our models with less data. The LSTM-KF architecture is illustrated in Fig. 2. This framework can be used to temporally regularize the output of any one-shot estimation technique, which from here forward will be considered a generic black-box estimator.

Specifically, our estimation model learns to predict the uncertainty of the initial prediction as well as the uncertainty of the incoming measurement, which is crucial in order to properly perform the update step. In addition, a learned motion model is employed also for the prediction step. Importantly, the estimator is not confined to the learned motion model, as it keeps on being refined by measurements during the update step. As a result, the filter learns to implicitly regularize the pose over time without the need for a hand-crafted transition or measurement model.

We believe that our approach is advantageous with respect to learning-based Kalman filter techniques such as those in [21, 18]. On one hand, in contrast to SVR [21], LSTM is able to estimate filter parameters using a model that depends on all previously observed inputs. On the other hand, by explicitly incorporating the prediction of LSTM with measurements in a Kalman update fashion, we relax the requirement on the LSTM to implicitly learn to fuse measurements with the state prediction for all possible motion paths, as attempted in [18]. Indeed, our model splits up the task of learning temporal regularization onto three distinct LSTMs that each have a defined objective: predicting the new state, estimating the prediction noise, and estimating the measurement noise. Due to this split of objectives in a Kalman filter fashion, each individual LSTM learns a simpler task and our model will automatically start to rely on the measurements in case of low accuracy predictions. We evaluate the LSTM-KF using three relevant pose esti-

mation tasks: body landmark localization, object tracking, and camera pose estimation, using real data from benchmark datasets. LSTM-KF outperforms both Kalman filters with different transition models and LSTM.

In the next section, we discuss related work. Next, we review Kalman filtering and long short-term memory in detail. In Section 4, we introduce the LSTM Kalman filter (LSTM-KF), including the underlying model, the modified prediction and update steps, and the full architecture which joins three LSTM modules with the Kalman filter. Next we move on to results, where we see LSTM-KF outperform other temporal regularization techniques, including standalone Kalman filters and standalone LSTM. Finally, we conclude and discuss future work.

2. Related Work

In recent literature, temporal regularization for pose estimation has been extensively studied. We will first focus on those works that use an implicit regularization scheme and in the second part discuss those that explicitly use a learning-based Kalman filter architecture to infer temporal coherence.

For 3D human pose estimation, Du *et al.* [6] trained an overcomplete dictionary of body joint positions as well as joint velocities. They use a Levenberg-Marquardt optimizer to find the dictionary basis coefficients that minimize the 2D backprojection error on the RGB input frame. This way, joint velocities are used to regularize the joint position estimates. In the experiments section we show that our approach yields superior results on the Human3.6M dataset.

Temporal regularization for 6 DOF object pose estimation was introduced by Krull *et al.* [19], who are using pose estimations from a random forest as input to a particle filter method. The particle filter propagates a posterior distribution of the objects pose though time, using a predefined constant velocity motion model. Choi *et al.* extend the particle filter approach by introducing improved 3D features and a GPU implementation[5].

Two main lines of work can be identified that combine machine learning and Kalman filter models for temporal regularization. We divide the approaches into those that learn static parameters of the Kalman filter and those that actively regress the parameters during filtering. *Static optimization* of noise covariance matrices was performed by Abbeel et al. [2], who seek to replace manual fine-tuning of noise parameters in robotic navigation tasks. The authors employ a coordinate ascent algorithm and optimize each individual element of the measurement and prediction noise covariance matrices. However, this approach is only valid for noisy but time-invariant systems. As opposed to our dynamic model, a change in measurement noise, for example due to partial occlusion of the tracked object, cannot be taken into account by their method and will therefore pro-
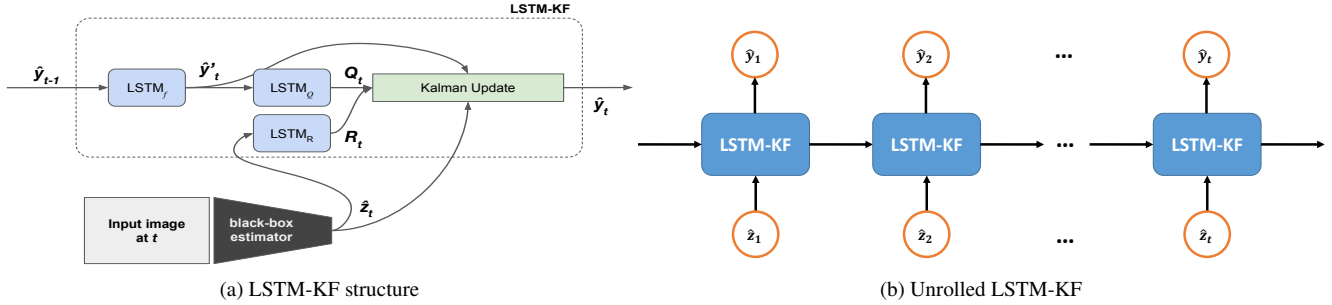
(a) LSTM-KF structure

(b) Unrolled LSTM-KF

Figure 2. **Overview of the LSTM-KF.** (a) A high-level depiction of the architecture which uses three LSTM modules to predict the internals of the Kalman filter. (b) The LSTM-KF unrolled over time, which can be trained end to end with backpropagation through time.

duce inaccurate state estimates.

Another approach is chosen by Krishnan et al. [18], who focus on learning the underlying state transition function that controls the dynamics of a hidden process state. However, only the state space equations of the Kalman filter are used, not the prediction and update scheme that performs optimally under the condition of linear state transitions and additive Gaussian noise [26]. Instead, the authors train neural network models that jointly learn to propagate the state, incorporate measurement updates and react to control inputs. Covariances were assumed to be constant throughout the estimation. In our experiments section, we show that this approach produces inferior state estimations than a distinct prediction and update model, especially in the absence of large-scale training data.

Dynamic regression of Kalman filter parameters was approached by Salti and Di Stefano [21]. In their work, support vector regression (SVR) is used to estimate a linear state transition function at each prediction step. The prediction noise covariance matrix is estimated jointly with the transition function. Their SVR based system is therefore able to deal with time-variant systems and outperforms manually tuned Kalman models on tracking tasks. As opposed to our model, measurement noise covariances are kept constant. The transition function is modeled as a matrix multiplication and can therefore only estimate linear motion models, while by design our model is able to estimate non-linear transition functions based on all previous state observations.

Haarnoja et al. [11] focus on the integration of a one-shot estimation as measurement into a Kalman framework, but require the estimator to provide a prediction of the noise covariance together with the measurement. The authors demonstrate a superior performance of their Kalman model by comparing to simple one-shot estimation and to a recurrent model that disregards measurement noise covariance. In contrast, our model is designed to regard the estimator that provides measurement updates as a black-box system and automatically estimates the measurement noise covariance based on past observations, which enables us to com-

bine it with existing one-shot estimators.

3. Background

In this section, we describe Kalman filters and long short-term memory (LSTM) and highlight the aspects of both methods which are most relevant to our LSTM Kalman filter, which we will describe in Section 4.

3.1. Kalman Filters

Kalman Filters (KFs) are optimal state estimators under the assumptions of linearity and Gaussian noise. More precisely, if we represent our state as $\mathbf{y}_t$ and our measurement as $\mathbf{z}_t$, and we assume the model

$$\mathbf{y}_t = \mathbf{A}\mathbf{y}_{t-1} + \mathbf{w}, \quad \mathbf{w} \sim N(\mathbf{0}, \mathbf{Q}) \tag{1}$$
$$\mathbf{z}_t = \mathbf{H}\mathbf{y}_t + \mathbf{v}, \quad \mathbf{v} \sim N(\mathbf{0}, \mathbf{R}) \tag{2}$$

where the matrices $\mathbf{A}$, $\mathbf{Q}$, $\mathbf{H}$, and $\mathbf{R}$ are known, then the Kalman filter yields the best estimate $\hat{\mathbf{y}}_t$ in terms of sum-of-squares error.

The Kalman filter achieves optimality through an iterative feedback loop with two update steps, the prediction step and the update step. In the prediction step, we estimate the mean and covariance of our current state, independent of the current measurement:

$$\hat{\mathbf{y}}'_t = \mathbf{A}\hat{\mathbf{y}}_{t-1} \tag{3}$$
$$\hat{\mathbf{P}}'_t = \mathbf{A}\hat{\mathbf{P}}_{t-1}\mathbf{A}^T + \mathbf{Q} \tag{4}$$

In the update step, we compute the optimal Kalman gain $\mathbf{K}_t$ and use this along with our *observed* measurement $\hat{\mathbf{z}}_t$ to estimate the mean and covariance of $\mathbf{y}_t$:

$$\mathbf{K}_t = \hat{\mathbf{P}}'_t\mathbf{H}^T(\mathbf{H}\hat{\mathbf{P}}'_t\mathbf{H}^T + \mathbf{R})^{-1} \tag{5}$$
$$\hat{\mathbf{y}}_t = \hat{\mathbf{y}}'_t + \mathbf{K}_t(\hat{\mathbf{z}}_t - \mathbf{H}\hat{\mathbf{y}}'_t) \tag{6}$$
$$\hat{\mathbf{P}}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\hat{\mathbf{P}}'_t \tag{7}$$

3.2. Long Short-Term Memory

Recurrent neural networks (RNNs), unlike their feedforward counterparts, are naturally suited to modeling sequential data. However, early variants such as simple RNNs
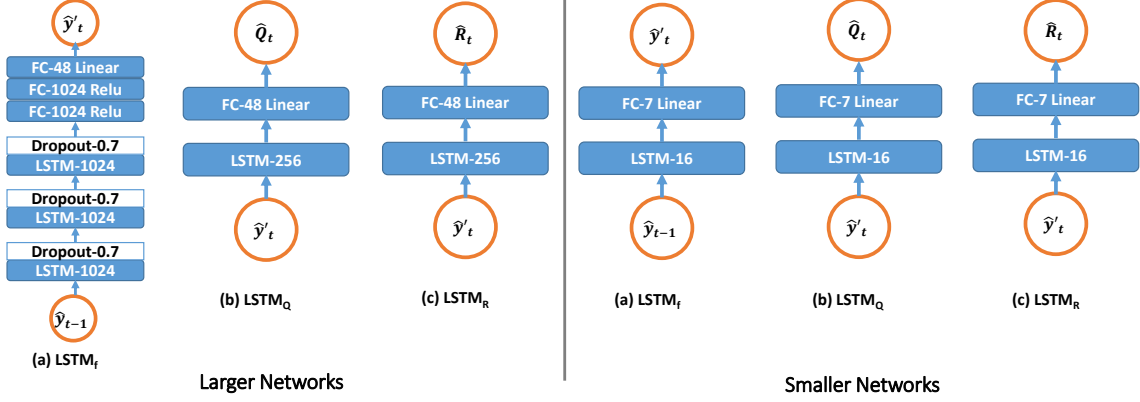
Figure 3. **LSTM-KF architectures.** As detailed in Section 5, the larger networks are used for the *Human 3.6M* dataset, and the smaller networks for all other (smaller) datasets.

[7] were extremely difficult to train because of what is now known as the *vanishing gradient problem* [12, 3].

Long short-term memory (LSTM) [13] was introduced specifically to address this problem, and has since become one of the most widely-used RNN architectures. In this work, we use the common variant with forget gates [8], which are known to be crucial to achieving good performance [10]. This LSTM variant is described by

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fx}\mathbf{x}_t + \mathbf{b}_f) \tag{8}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ix}\mathbf{x}_t + \mathbf{b}_i) \tag{9}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{ox}\mathbf{x}_t + \mathbf{b}_o) \tag{10}$$
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{W}_{cx}\mathbf{x}_t + \mathbf{b}_c) \tag{11}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{12}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{13}$$

where $\sigma(\cdot)$ denotes the element-wise sigmoid function and $\odot$ denotes element-wise multiplication. Focusing on Equations 12 and 13, we can see that LSTM can be interpreted as resetting memory according to the forget gate $\mathbf{f}_t$, writing to memory according to the input gate $\mathbf{i}_t$, and reading from memory according to the output gate $\mathbf{o}_t$, finally forming the output or *hidden state*, $\mathbf{h}_t$, at time step t. The intermediate memory cell $\tilde{\mathbf{c}}_t$ and all gates depend on $\mathbf{x}_t$, the input at the current time step, and on all $\mathbf{W}$ and $\mathbf{b}$, which collectively form the parameters to be learned.

This architecture also easily extends to multiple-layer LSTM, where the hidden state $\mathbf{h}_t$ from the first layer is simply treated as the input $\mathbf{x}_t$ to the second layer, or from the second to third layer, and so on.

4. LSTM Kalman Filters

In this section, we present the long short-term memory Kalman filter (LSTM-KF), a model for the temporal regularization of pose estimators. The main idea is to leverage Kalman filters *without* the need to specify a linear transition

function $\mathbf{A}$ or fixed process and measurement covariance matrices $\mathbf{Q}$ and $\mathbf{R}$. Instead, we will model a nonlinear transition function f along with $\mathbf{Q}$, and $\mathbf{R}$ using three different long short-term memory (LSTM) networks, thus providing our model with the ability to learn rich, dynamic Kalman components from data.

4.1. Model

We always assume that incoming measurements are noisy estimates of the underlying state, and thus $\mathbf{H} = \mathbf{I}$ in Equation 2. Equations 1 and 2 then take on the modified form

$$\mathbf{y}_t = f(\mathbf{y}_{t-1}) + \mathbf{w}_t, \quad \mathbf{w}_t \sim N(\mathbf{0}, \mathbf{Q}_t) \tag{14}$$
$$\mathbf{z}_t = \mathbf{y}_t + \mathbf{v}_t, \quad \mathbf{v}_t \sim N(\mathbf{0}, \mathbf{R}_t) \tag{15}$$

which specifies the underlying model of the LSTM-KF.

4.2. Prediction and Update Steps

Our prediction step is then defined by

$$\hat{\mathbf{y}}'_t = f(\hat{\mathbf{y}}_{t-1}) \tag{16}$$
$$\hat{\mathbf{P}}'_t = \mathbf{F}\hat{\mathbf{P}}_{t-1}\mathbf{F}^T + \hat{\mathbf{Q}}_t \tag{17}$$

where f is modeled by one LSTM module, $\mathbf{F}$ is the Jacobian of f with respect to $\hat{\mathbf{y}}_{t-1}$, and $\hat{\mathbf{Q}}_t$ is the output of a second LSTM module. Finally, our update step is

$$\mathbf{K}_t = \hat{\mathbf{P}}'_t(\hat{\mathbf{P}}'_t + \hat{\mathbf{R}}_t)^{-1} \tag{18}$$
$$\hat{\mathbf{y}}_t = \hat{\mathbf{y}}'_t + \mathbf{K}_t(\hat{\mathbf{z}}_t - \hat{\mathbf{y}}'_t) \tag{19}$$
$$\hat{\mathbf{P}}_t = (\mathbf{I} - \mathbf{K}_t)\hat{\mathbf{P}}'_t \tag{20}$$

where $\hat{\mathbf{R}}_t$ is the output of a third LSTM module and where $\hat{\mathbf{z}}_t$ is our observed measurement at time t. Next we describe these LSTM modules in detail.

4.3. Architecture

We denote the three LSTM modules for f, $\hat{\mathbf{Q}}_t$, and $\hat{\mathbf{R}}_t$ by LSTM_f, LSTM_Q, and LSTM_R; each is depicted in Fig. 3, and an overview of the system is depicted in Fig. 2.

At each time step t, LSTM_f takes in the previous prediction $\hat{\mathbf{y}}_{t-1}$ as input and produces the intermediate state $\hat{\mathbf{y}}'_t$ (which does not depend on the current measurement). LSTM_Q then takes $\hat{\mathbf{y}}'_t$ as input and produces an estimate of the process covariance, $\hat{\mathbf{Q}}_t$, as output. Meanwhile, the observation $\mathbf{z}_t$ serves as input to LSTM_R, which only produces an estimate of the measurement covariance, $\hat{\mathbf{R}}_t$, as output. Finally, $\hat{\mathbf{y}}'_t$ and $\mathbf{z}_t$, along with our covariance estimates, are fed to a standard Kalman filter, as described by Equations 17 through 20, finally producing the new prediction $\hat{\mathbf{y}}_t$.

We remark that in this work $\mathbf{Q}$ and $\mathbf{R}$ are restricted to be diagonal, and they are restricted to be positive definite by exponentiating the outputs of the LSTM_Q and LSTM_R modules.

4.4. Loss

In preliminary experiments, we used standard Euclidean loss summed over all time steps, but in this case we found that the LSTM_f module would fail to learn any reasonable mapping. Because of this, we added a term to our loss to enhance gradient flow to the LSTM_f block, resulting in the loss

$$L(\boldsymbol{\theta}) = \frac{1}{T}\sum_{t=1}^{T}\|\mathbf{y}_t - \hat{\mathbf{y}}_t(\boldsymbol{\theta})\|^2 + \lambda\|\mathbf{y}_t - \hat{\mathbf{y}}'_t(\boldsymbol{\theta})\|^2 \quad (21)$$

We set the hyperparameter λ to 0.8 using the Human3.6M dataset and kept it fixed for all other experiments, as we found that performance was relatively insensitive around this value.

4.5. Optimization

Our objective is to optimize all parameters $\boldsymbol{\theta}$ to minimize the loss given by Equation 21 with respect to all free parameters in our model, which are a concatenation of all weight matrices and biases from all three LSTM modules. (Note that these modules are combinations of LSTM layers and linear layers, as depicted by figure 3.)

Our model can be trained end to end, with gradients obtained using the backpropagation through time algorithm[27], which we implement using the TensorFlow framework [1]. We use gradient updates according to the Adam [17] optimizer.

5. Experiments

In this section we compare the pose estimation performance of our LSTM-KF architecture to a range of temporal
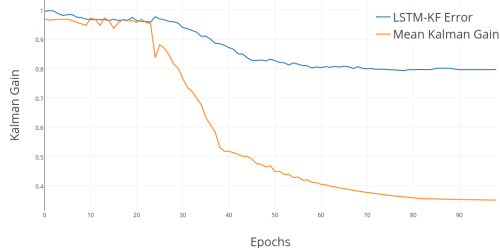

Figure 4. **LSTM-KF error and mean Kalman gain during training.** At the beginning of training, the Kalman gain (as well as error) is high, indicating that the model is relying almost entirely on measurements. As training progresses, the Kalman gain drops considerably, indicating that the Kalman filter relies significantly on both on the measurements and the LSTM_f module's output.

regularization methods, including two standard Kalman filters that assume either a constant velocity or constant acceleration motion (respectively *Kalman Vel, Kalman Acc*), to an exponential moving average filter (*EMA*), and to a standard LSTM module (*Std. LSTM*). Specifically, this LSTM model that we compare to is a representative of the class of models proposed in [18], and it is characterized by implicitly learning the prediction step as well as the measurement update step in an end-to-end fashion.

We evaluate these models on four different datasets, one for 3D human pose estimation, two for camera pose estimation, and one for object pose estimation, all of them using RGB images as input modality [14, 16, 22].

5.1. Implementation Details

We initialize all LSTM state-to-state weight matrices as random orthogonal matrices, all other LSTM weight matrices using a uniform distribution over $[-0.01, 0.01]$, and all linear-layer weight matrices using Xavier initialiation [9]. All biases are initialized with zeros except for LSTM forget-gate bias; following best practices, we set these biases to 1.0 [8, 10].

Noise covariance matrices of the Kalman filter methods (*Kalman Vel, Kalman Acc*) as well as the window size of the exponential moving average method (*EMA*) were optimized via grid search.

5.2. Human Pose Estimation

The *Human3.6M* dataset of Ionescu *et al.* [14], consists of 3.6 million RGB video frames from video sequences that were recorded in a controlled indoor motion capture setting. In each of these sequences, one out of seven actors performs 15 activities with varying levels of movement complexity. Each of the activities is between 3,000 and 5,000 frames long. In our experiments, we follow the same data partition scheme as [4, 28] for training and test set: training has 5 subjects (*S1, S5, S6, S7, S8*) and test data 2 subjects (*S9, S11*). Similar to [4] we compute the model performance

	Directions	Discussion	Eating	Greeting	Phoning	Photo	Posing	Purchases
Li *et al.* [20]	-	136.88	96.94	124.74	-	168.68	-	-
Tekin *et al.* [25]	102.39	158.52	87.95	126.83	118.37	185.02	114.69	107.61
Zhou *et al.* [28]	87.36	109.31	87.05	103.16	116.18	143.32	106.88	99.78
SMPLify [4]	62.0	**60.2**	67.8	76.5	92.1	**77.0**	73.0	75.3
Inception	67.18	74.79	71.80	73.85	81.04	88.73	72.58	73.12
+ Kalman Vel.	67.70	74.01	71.73	73.32	80.74	88.03	72.22	73.45
+ Kalman Acc.	67.08	74.75	71.21	73.23	80.74	88.01	72.11	73.31
+ EMA	67.01	74.78	71.81	73.81	81.04	88.70	72.50	72.02
+ Std. LSTM	62.70	70.11	63.53	67.24	75.42	85.37	67.42	67.07
+ LSTM-KF (**ours**)	**61.41**	69.98	**62.12**	**65.93**	**71.93**	83.92	**63.0**	**65.87**
	Sitting	SitDown	Smoking	Waiting	WalkDog	Walk	WalkTogether	**Mean**
Li *et al.* [20]	-	-	-	-	132.17	69.97	-	-
Tekin *et al.* [25]	136.15	205.65	118.21	146.66	128.11	65.86	77.21	125.28
Zhou *et al.* [28]	124.52	199.23	107.42	118.09	114.23	79.39	97.70	113.01
SMPLify [4]	100.3	137.3	83.4	83.4	79.7	86.8	81.7	82.3
Inception	91.36	111.19	79.25	71.67	88.04	71.95	74.01	79.8
+ Kalman Vel.	91.04	111.1	79.01	71.90	87.99	87.99	74.35	79.20
+ Kalman Acc.	90.88	111.11	79.13	71.51	87.62	87.62	74.10	79.07
+ EMA	91.31	111.11	79.21	71.70	88.04	71.91	73.97	79.26
+ Std. LSTM	85.15	104.16	72.69	72.68	80.77	59.23	61.36	73.22
+ LSTM-KF (**ours**)	**84.81**	**98.85**	**69.79**	**65.88**	**79.44**	**55.32**	**60.29**	**70.98**

Table 1. Average 3D joint error on *Human 3.6M* for test subjects 9 and 11. The error is given in [mm].

in terms of average Euclidean distance between estimated and ground-truth 3D joint positions. Furthermore, following previous works for this dataset, we express all joint positions relative to a root joint, which is the pelvis joint in our case. In order to get initial 3D human pose estimations on the RGB videos, we refine a Inception-v4 CNN model that was pre-trained on ImageNet [23]. For this fine tuning, we use a batch size of 30 and set the initial learning rate to 0.01 and reduce it about a decay factor of 10 at each epoch, and train for a total of only 3 epochs. To prevent overfitting, we augment the RGB data by randomly cropping 300×300 patches from the 350×350 input images and randomly distort the brightness, hue, saturation and contrast of each input image. Besides data augmentation, we apply dropout in the last layer, retaining values with a probability of 0.8. Retraining the network for the pose estimation task on a Tesla K40 GPU took 10 days. We then use the Inception-v4 estimation values as measurement inputs to train the LSTM-KF and standard LSTM model.

In particular, given the abundance of training samples for this dataset, we employ the bigger network architectures presented in Fig. 3. Specifically, LSTM_f consists of 3 stacked layers with 1024 hidden units each, followed by three fully connected (FC) layers with 1024, 1024 and 48 hidden units. The standard LSTM is constructed in the same way as LSTM_f. We apply the ReLU non-linearity to all FC layer activations except for the last layer, and each LSTM layer is followed by a dropout layer with a keep probability

Figure 5. **Measurement noise covariance during occlusion.** Here we include the Euclidean norm of covariance coefficients for the left hand (normalized between 0 and 1) along with the corresponding images from a *Walking* test sequence. The model has learned to assign high measurement uncertainty to those frames in which the left hand is occluded.

of 0.7. LSTM_Q and LSTM_R follow a single layer architecture with 256 hidden units, followed by an FC layer with 48 hidden units. LSTM-KF and the standard LSTM are trained with a learning rate of 1e-5, with a decay of 0.95 starting from the second epoch. For this training we use truncated backpropagation through time, propagating gradients for 100 time steps. Qualitative pose estimation results are shown in Figs. 1 and 6 and quantitative pose estimation errors in Table 1 together with those of four recently published state-of-the-art approaches. We furthermore show how the estimated measurement noise covariance develops over the course of a test sequence in Fig. 5.

The results show that the LSTM-KF significantly improves on the raw measurements and outperforms standard LSTM across all actions, achieving on average 14% improvement over the best state-of-the-art approach. Furthermore, as expected, temporal information consistently

	Chess		Fire		Heads		Office		Pumpkin		R. Kitchen		Stairs		**Mean**	
	tran.	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*
PoseNet [16]	0.38	7.51°	0.47	16.61°	0.32	13.6°	0.48	7.79°	0.54	11.17°	0.59	9.14°	0.55	15.65°	0.50	11.47°
+ Kalman Vel.	0.38	8.35°	0.47	16.66°	0.32	14.73°	0.48	8.64°	0.54	12.06°	0.59	9.94°	0.54	16.58°	0.50	12.40°
+ Kalman Acc.	0.37	8.34°	0.47	16.67°	0.32	14.71°	0.48	8.62°	0.54	12.09°	0.59	9.95°	0.54	16.58°	0.49	12.39°
+ EMA	0.37	7.31°	0.47	16.46°	0.32	13.53°	0.47	**7.48°**	0.54	11.01°	0.53	8.85°	0.55	15.56°	0.49	11.29°
+ Std. LSTM	0.41	8.4°	0.5	17°	0.35	15.05°	0.48	9.99°	0.53	**10.38°**	**0.51**	9.71°	0.65	**13.62°**	0.51	11.75°
+ LSTM-KF (ours)	**0.33**	**6.9°**	**0.41**	**15.7°**	**0.28**	**13.01°**	**0.43**	7.65°	**0.49**	10.63°	0.57	**8.53°**	**0.46**	14.56°	**0.44**	**10.83°**

Table 2. Comparison of temporal regularisation methods on camera pose estimations provided by PoseNet on the *7 Scenes* dataset. As in [16], values are given as median errors in translation [m] and rotation [degrees].

	Street		K. College		S. Facade		St. M. Church		Old Hospital		**Mean**	
	tran.	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*
PoseNet [16]	3.35	6.12°	1.97	5.38°	1.65	8.49°	2.88	9.04°	2.60	5.32°	2.49	6.87
+ Kalman Vel.	3.16	5.93°	**1.85**	5.29°	**1.48**	8.20°	2.94	9.29°	2.53	5.07°	2.39	6.75°
+ Kalman Acc.	3.14	5.92°	1.88	5.29°	1.49	8.33°	2.95	9.33°	2.45	5.07°	2.38	6.79°
+ EMA	3.33	5.63 °	1.95	**5.28°**	1.62	8.35°	2.82	8.99°	2.68	5.10°	2.48	6.67°
+ Std. LSTM	9.56	11.2°	4.24	7.95°	1.87	7.04°	3.34	11.52°	4.03	6.46°	4.61	8.83°
+ LSTM-KF (ours)	**3.05**	**5.62°**	2.01	5.35°	1.63	**6.89°**	**2.61**	**8.94°**	**2.35**	**5.05°**	**2.33**	**6.37°**

Table 3. Comparison of temporal regularisation methods on camera pose estimations provided by PoseNet on the *Cambridge Landmarks* dataset. As in [16], values are given as median errors in translation [m] and rotation [degrees].
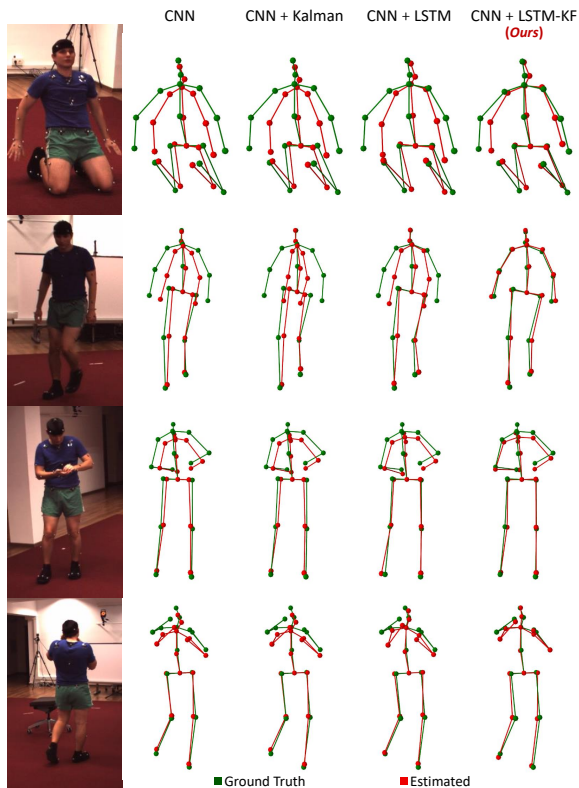

Figure 6. Qualitative results on the *Human3.6M* dataset. Ground truth pose in green and estimation in red. Based on the initial CNN estimation, we compare temporal regularization output of Kalman, standard LSTM and our LSTM-KF method. Especially for arm and leg joints, our model improves over the other methods.

improves over the raw one-shot estimations from the Inception-v4 model. It is also relevant to note that the use of the inception architecture alone outperforms previous work.

5.3. Camera Tracking

To demonstrate the wide applicability of our method, we selected camera pose estimation as another application domain and evaluate on the *Cambridge Landmarks*[16] and *7 Scenes*[22] datasets. The *Cambridge Landmarks* dataset contains 5 different large outdoor scenes of landmarks in the city of Cambridge. The *7 Scenes* dataset contains 7 image series captured in typical everyday indoor scenes. Both datasets come with a predefined training and test split that we follow. In order to generate one-shot camera pose estimates on which we compare the temporal regularisation methods, we retrain the publically avaliable PoseNet CNN architecture [16] on the respective training partition of each dataset.

Since these datasets are much smaller than the previously used Human3.6M dataset, we employ the smaller network architectures presented in Fig. 3 so to prevent overfitting. Specifically, for $LSTM_f$, $LSTM_Q$, and $LSTM_R$ we use a single layer architecture with 16 hidden units, where each LSTM layer is followed by a fully connected layer without non-linearity. The standard LSTM follows the $LSTM_f$ architecture. We use batch size of 2, set the learning rate to 5e-4, and train for 10 epochs. Here, we use truncated backpropagation through time, propagating gradients for 10 time steps.

Table 3 for *Cambridge Landmarks* and Table 2 for *7 Scenes* show the quantitative results on those datasets. Our

	Kinect Box		Tide		Orange Juice		Milk		**Mean**	
	tran.	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*	*tran.*	*rot.*
Tan *et al*. [24]	1.70	0.30°	1.17	0.44°	1.29	0.35°	1.27	0.41°	1.36	0.37°
+ Kalman Vel. al.	1.69	0.29°	1.84	**0.38°**	1.27	0.35°	1.27	**0.35°**	1.52	0.34°
+ Kalman Acc.	1.69	**0.28°**	1.84	**0.38°**	1.28	0.31°	1.79	0.42°	1.65	0.35°
+ EMA	1.71	**0.28°**	1.17	0.39°	1.50	**0.28°**	1.49	0.37°	1.47	**0.33°**
+ Std. LSTM	41.03	6.30°	32.23	8.31°	30.16	7.42°	18.3	7.95°	30.43	7.49°
+ LSTM-KF (**ours**)	**0.86**	0.35°	**0.77**	0.49°	**0.59**	0.37°	**0.66**	0.43°	**0.72**	0.41°

Table 4. We show the effect of temporal regularisation on object tracking estimations of Tan *et al*. We denoting the errors in translation as [mm] and rotation in [degrees]

approach consistently improves estimations on the *7 Scenes* dataset. The same is true for the *Cambridge Landmarks* dataset, except for the *King's College* and *S. Facade* sequence. In the *King's College* sequence, learning the motion model might be a disadvantage, as the camera trajectory in the training set moves in curves, while in the test set it resembles a straight line. The *S. Facade* sequence poses a different challenge for the LSTM-KF, as its training set only consists of 231 frames, which is most likely too short for the $LSTM_f$ to learn a valid motion model (average training sequence length: 1370 frames). Since the datasets are quite limited in size, the standard LSTM was not able to improve the results, and even decreases the accuracy. Our LSTM-KF model achieves an improvement of up to 6.23% for translation and 7.53% for rotation on average over the *Cambridge Landmarks* dataset, while *Kalman Vel* and *Kalman Acc* improve 4.1% and 4.43% for translation and 1.66% and 1.17% for rotation, respectively. For the *7 Scenes* dataset, LSTM-KF improves the PoseNet estimations about 10.13% for translation and 7.53% for rotation. *Kalman Acc*, *Kalman Vel* and standard LSTM algorithms were not able to improve over the original PoseNet estimation.

5.4. Object Tracking

As third experiment, we evaluated our method on the public *MIT RGB-D Object Pose Tracking Dataset* [5]. As in Tan *et al*. [24], we used four synthetically generated object tracking sequences from the dataset, for which 6-DOF ground truth poses were available. The sequences consist of 1,000 RGB-D frames in which the tracked object (*Kinect Box, Milk, Orange Juice, Tide*) was rendered in front of a virtual kitchen scene.

Our model parameters were set up equal to experiment 5.3, specifically using single layer LSTMs with 16 hidden units, a batch size of 2 and a learning rate of 5e-4. We trained for 120 epochs, again using truncated back-propagation through time, propagating gradients for 10 time steps. The same holds true for the standard LSTM method that we evaluated against. As no separate training set was provided, we performed 2-fold cross validation by training on the *Kinect Box* and *Milk* sequence to test on *Orange*

Juice, Tide and vice versa. As input to all methods, we use the raw object pose estimations of [24], which were provided by the authors. This tracking algorithm exploits successive frame pairs to estimate the 3D pose of a 3D CAD model being tracked through a sequence of depth frames. Hence, the task for all methods compared in this experiment is to gain additional improvements over an existing object tracking method. Results for this scenario are reported in Table 4. The methods that did not learn the motion model on training data, i.e. *Kalman Vel, Kalman Acc* and *EMA*, were not able to meaningfully improve on the translation estimation, while rotation was slightly improved. For the object position, LSTM-KF achieves the best results at 0.72 mm average error, improving 47.05 % over the original estimation. The standard LSTM approach yields a high error in both position and rotation estimation. It does not follow the measurement and starts to deviate from the correct trajectory rather quickly. We assume that the task of implicit fusion of past state and measurement update is too difficult for the standard LSTM to learn, given the available training data.

6. Conclusions

In this work, we introduced the long short-term memory Kalman filter (LSTM-KF). This model alleviates the modeler from specifying motion and noise models a priori and simultaneously allows the learning of rich models from data which are extremely difficult to write down explicitly. In an extensive set of experiments, we found that the LSTM-KF outperforms both the standalone Kalman filter and standalone LSTM for temporal regularization. In addition, we achieved state-of-the-art performance on three diverse tasks, for example reducing the joint error in the Human 3.6M dataset by 13.8%, from 82.3 mm to 71.0 mm.

7. Acknowledgments

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[2] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun. Discriminative Training of Kalman Filters. *Proceedings of Robotics: Science and Systems I*, pages 289–296, 2005.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[4] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it smpl : Automatic estimation of 3d human pose and shape from a single image. In *Proc. European Conf. on Computer Vision (ECCV)*, pages 34–36, 2016.

[5] C. Choi and H. I. Christensen. RGB-d object tracking: A particle filter approach on GPU. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1084–1091, 2013.

[6] Y. Du, Y. Wong, Y. Liu, F. Han, Y. Gui, Z. Wang, M. Kankanhalli, and W. Geng. Marker-less 3d human motion capture with monocular image sequence and height-maps. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[7] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[8] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471, 2000.

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[10] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[11] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel. Backprop KF: Learning Discriminative Deterministic State Estimators. In *Int. Conf. on Neural Information Processing Systems (NIPS)*, 2016.

[12] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, page 91, 1991.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2014.

[15] R. E. Kalman et al. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[16] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *Proc. Int. Conf. on Computer Vision (ICCV)*, 2015.

[17] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. In *NIPS Workshop on Advances in Approximate Bayesian Inference and Black Box Inference*, 2015.

[19] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother. 6-dof model based tracking via object coordinate regression. In *Proc. Asian Conference on Computer Vision (ACCV)*, 2014.

[20] S. Li, W. Zhang, and A. B. Chan. Maximum-margin structured learning with deep networks for 3d human pose estimation. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 2848–2856, 2015.

[21] S. Salti and L. Di Stefano. Online support vector regression of the transition model for the kalman filter. *Image and Vision Computing*, 2012.

[22] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013.

[23] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proc. 31st Conference on Artificial Intelligence (AAAI)*, pages 4278–4284, 2017.

[24] D. J. Tan, F. Tombari, S. Ilic, and N. Navab. A Versatile Learning-Based 3D Temporal Tracker: Scalable, Robust, Online. In *Proc. Int. Conf. on Computer Vision (ICCV)*, 2015.

[25] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, and P. Fua. Structured prediction of 3d human pose with deep neural networks. In *Proc. British Conf. on Computer Vision (BMVC)*, 2016.

[26] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical Report 1, University of North Carolina at Chapel Hill, 2006.

[27] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[28] X. Zhou, M. Zhu, S. Leonardos, K. G. Derpanis, and K. Daniilidis. Sparseness meets deepness: 3d human pose estimation from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4966–4975, 2016.