Dynamic Facial Analysis: From Bayesian Filtering to Recurrent Neural Network

Jinwei Gu, Xiaodong Yang, Shalini De Mello, Jan Kautz
Nvidia
jinweig@nvidia.com

Abstract

Facial analysis in videos, including head pose estimation and facial landmark localization, is key for many applications such as facial animation capture, human activity recognition, and human-computer interaction. In this paper, we propose to use a recurrent neural network (RNN) for joint estimation and tracking of facial features in videos. We are inspired by the fact that the computation performed in an RNN bears resemblance to Bayesian filters, which have been used for tracking in many previous methods for facial analysis from videos. Bayesian filters used in these methods, however, require complicated, problem-specific design and tuning. In contrast, our proposed RNN-based method avoids such tracker-engineering by learning from training data, similar to how a convolutional neural network (CNN) avoids feature-engineering for image classification. As an end-to-end network, the proposed RNN-based method provides a generic and holistic solution for joint estimation and tracking of various types of facial features from consecutive video frames. Extensive experimental results on head pose estimation and facial landmark localization from videos demonstrate that the proposed RNN-based method outperforms frame-wise models and Bayesian filtering. In addition, we create a large-scale synthetic dataset for head pose estimation, with which we achieve state-of-the-art performance on a benchmark dataset.

1. Introduction

Analyzing facial features, including estimating head pose [33] and locating facial landmarks [7], from consecutive video frames is of great importance for many applications, such as facial animation capture, activity recognition, and human-computer interaction. Videos provide temporal links among neighboring frames, which have been shown to be useful for accurate and robust estimation [50].

Prior work on dynamic facial analysis [32, 7] primarily employ Bayesian filters, *e.g.*, Kalman filters (KF) or particle filters (PF), to exploit temporal connections. However, for facial tracking, these Bayesian filters require complicated, problem-specific design and tuning. For example, tracking can be performed at different levels for different tasks—tracking of a face's bounding box [49], of the rigid transformation parameters of heads [31, 55], or of facial features [38]. To deal with drifting, many tracking methods require a failure detection and reinitialization method as a backup [31]. For complex tasks (*e.g.*, non-rigid face, hand, or body tracking), implementing a Bayesian filter can be challenging. In short, the need for such *tracker-engineering* makes these methods cumbersome and less generic for dynamic facial analysis.

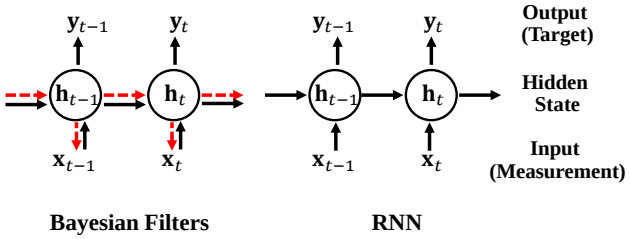

Figure 1: Connection between Bayesian filters (left) and RNN (right). Bayesian filters model the dynamics between the hidden state $\mathbf{h}_t$ and the measurement $\mathbf{x}_t$ as a stochastic Markov process (shown as dash red arrows). Given a sequence of noisy measurement $\mathbf{x}_t$, the goal of Bayesian filtering is to estimate the optimal states $\mathbf{h}_t$ and optionally the target output $\mathbf{y}_t$ that is a function of the state $\mathbf{h}_t$ (shown as solid black arrows). Similarly, RNN learns the prediction from a sequential input $\mathbf{x}_t$ to the sequential output $\mathbf{y}_t$ by passing information over time via the hidden state $\mathbf{h}_t$.

In this paper, we propose to use RNN for joint estimation and tracking of facial features in videos. We are inspired by the fact that the computation performed in RNN bears resemblance to Bayesian filters, as shown in Fig. 1. As a generic and learning-based approach for time series prediction, RNN avoids *tracker-engineering* for tasks performed on videos, much like CNN avoids *feature-engineering* for tasks performed on images. The proposed method provides a generalized and integrated solution for joint estimation and tracking of various facial features in dynamic facial analysis. Our main contributions are three-fold:

- We systematically study the connections between Bayesian filtering and RNN. We show that Bayesian filtering (in particular Kalman filtering) is a special type of RNN with adaptive weights. For complex facial tracking problems, RNN may serve as an effective alternative approach, given sufficient training data.

- We propose an end-to-end RNN-based approach for general facial analysis tasks from videos including head pose estimation and facial landmark estimation. Prior work using deep learning methods for facial analysis are either for still images [40, 30, 52] or specifically designed for face alignment only [37]. Experimental results show that the proposed RNN-based method outperforms frame-wise estimation and Bayesian filtering in both tasks.

- To cope with the need for large training data with accurate annotations, we create a large-scale synthetic dataset, called SynHead, for head pose estimation. SynHead contains 10 head models, 70 motion tracks and 510,960 frames. With this dataset, we achieve state-of-the-art performance for head pose estimation on benchmark datasets. We will release the SynHead dataset upon publication, as we believe it will be invaluable for the field of dynamic facial analysis.

2. Related Work

Head Pose Estimation Many techniques have been proposed for head pose estimation using RGB [33, 40], depth [28, 13, 36], and a combination of both modalities [30, 41, 2, 36]. They use either rigid/deformable model fitting [28, 35, 27, 33, 2, 41] or train regression functions to map input images to the head pose manifolds [30, 40, 47, 13]. Another widely used approach is to locate the facial landmarks and use them to estimate head pose with the POSIT [10] algorithm.

The vast majority of these techniques estimate the head's pose independently for each frame, with the exception of the method developed by Murphy-Chutorian et al. [33], which employs a particle filter coupled with a face image renderer to track the head's pose in videos. The particle filter in [33] is a complicated and heavily hand-tuned system, that is specifically designed for use in cars.

Facial Landmark Localization There are two main categories of methods for facial landmark localization: the discriminative regression-based methods [54, 5, 22, 44], and the generative model-based methods [8, 9, 48]. A few recent studies achieve good performance via a divide-and-conquer strategy, by localizing semantically meaningful facial parts via multi-stage regression [53], or using different estimators for different head pose angles [39] or head shapes [51]. The HyperFace method was recently proposed in [40] to construct a multi-tasking network for landmark localization from still images. RNN was also recently used for the refinement of facial landmarks [52]. Refer to [7] for a latest and comprehensive survey on facial landmark localization.

In order to utilize the temporal information from videos, most prior work focuses on tracking the detected bounding box of a face [49, 7, 20]. This global rigid motion tracking, however, does not benefit the tracking of local non-rigid facial deformations such as facial expression. A recent method in [55] tracks the pose and size of the 3D head, and uses the multi-frame cascade shape regression and re-initialization to achieve top performance on the 300-VW challenge [20]. Prabhu *et al*. [38] exploit Kalman filter for facial landmark localization in videos, where they propose to track the position, orientation, and size of the the facial shape, as well as its top four PCA coefficients separately. Peng *et al*. [37] recently designed a spatial-temporal RNNs for sequential face alignment (see Sec 5 for comparison).

All the tracking methods for dynamic facial analysis require complex, problem-specific design and tuning (except for the model-free tracking of the face's bounding box, which however has limited use). In contrast, the proposed RNN-based approach is a generic, end-to-end approach that (1) directly learns optimal feature extraction via CNN and tracking via RNN from training data, and (2) can be easily applied to different sub-tasks of facial analysis from videos.

RNN and Bayesian Filtering RNN is a generic and learning-based approach for time series prediction, with successful applications in speech recognition [15], natural language processing [46], activity recognition [11, 34], and hand gesture detection and classification [29]. Variants of RNN include long short-term memory (LSTM) [17] and gated recurrent unit (GRU) [6], which are able to adaptively discover temporal dependencies at different time scales. Recently, there have been a few studies that discuss the relationship between a Kalman filter and an RNN. Haarnoja *et al*. [16] propose to train a Kalman filter as a type of RNN with back propagation. Krishnan *et al*. [24] introduce to use an RNN as a component in a Kalman filter. While these two methods focus on integrating an RNN with a Kalman filter, in this paper, we focus on finding the connections between Bayesian filters and RNN and advocate RNN as a generic, alternative approach for facial analysis in videos.

3. From Bayesian Filtering to RNN

Figure 1 illustrates the connections between Bayesian filters and RNN. Bayesian filters model the dynamics between the state $\mathbf{h}_t$ and the measurement $\mathbf{x}_t$ as a stochastic Markov process with two conditional probability distributions $p(\mathbf{h}_t|\mathbf{h}_{t-1})$ and $p(\mathbf{x}_t|\mathbf{h}_t)$. Consider a basic form of a Bayesian filter, *i.e*., the linear Kalman filter, which assumes Gaussian distributions for $p(\mathbf{h}_t|\mathbf{h}_{t-1})$ and $p(\mathbf{x}_t|\mathbf{h}_t)$

and linear models for state transition and measurement:

$$p(\mathbf{h}_t|\mathbf{h}_{t-1}) = \mathcal{N}(\mathbf{W}\mathbf{h}_{t-1}, \mathbf{\Sigma}_p)$$
$$p(\mathbf{x}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{V}\mathbf{h}_t, \mathbf{\Sigma}_m),$$

where $\mathbf{h}_t$ is the state and $\mathbf{x}_t$ is the measurement at time t, $\mathbf{\Sigma}_p$ and $\mathbf{\Sigma}_m$ are covariance matrices of the process noise and measurement noise, and $\mathbf{W}$ and $\mathbf{V}$ are the matrices for the state transition and measurement models, corresponding to the red dash arrows shown in the left of Figure 1.

Given a sequence of noisy measurements $\mathbf{x}_t$, the goal of Bayesian filtering is to estimate the states $\mathbf{h}_t$ (and optionally the target output $\mathbf{y}_t$ that is a function of the state $\mathbf{h}_t$), as shown as the solid black arrows in the left of Figure 1. For the linear Kalman filter [21], the optimal estimator is

$$\begin{aligned}\mathbf{h}_t &= \mathbf{W}\mathbf{h}_{t-1} + \mathbf{K}_t(\mathbf{x}_t - \mathbf{V}\mathbf{h}_{t-1}) \\ &= (\mathbf{W} - \mathbf{K}_t\mathbf{V})\mathbf{h}_{t-1} + \mathbf{K}_t\mathbf{x}_t \\ &= \mathbf{W}^t_{bh}\mathbf{h}_{t-1} + \mathbf{W}^t_{bi}\mathbf{x}_t, \end{aligned} \tag{1}$$

where $\mathbf{K}_t$ is the Kalman gain matrix that updates over time, $\mathbf{W}^t_{bh} = \mathbf{W} - \mathbf{K}_t\mathbf{V}$ and $\mathbf{W}^t_{bi} = \mathbf{K}_t$ are the two weight matrices that relate $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ to $\mathbf{h}_t$. With the estimated state $\mathbf{h}_t$, we can estimate the target output as $\mathbf{y}_t = \mathbf{V}\mathbf{h}_t$.

An extension of the Kalman filter is the particle filter [43], which is widely used in computer vision for tracking non-rigid objects in videos. In a particle filter, $p(\mathbf{h}_t|\mathbf{h}_{t-1})$ and $p(\mathbf{x}_t|\mathbf{h}_t)$can be non-parametric probability distributions (*e.g.*, histograms of sampled particles). The optimal estimate of the state $\mathbf{h}_t$ is the maximum likelihood estimation given the current measurement $\mathbf{x}_t$ and a set of randomly initialized particles.

The computation in Equation (1) resembles to that of RNN, which is a sequence-based model to capture temporal evolution. It keeps a recurrent hidden state $\mathbf{h}_t$ whose activation depends on that of the previous time step $\mathbf{h}_{t-1}$,

$$\mathbf{h}_t = \mathcal{H}(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{ih}\mathbf{x}_t + \mathbf{b}_h), \tag{2}$$

where $\mathcal{H}$ is an activation function, $\mathbf{W}_{hh}$ is the hidden-to-hidden matrix, $\mathbf{h}_{t-1}$ is the hidden state from previous time step, $\mathbf{W}_{ih}$ is the input-to-hidden matrix, $\mathbf{x}_t$ is the input to this layer, and $\mathbf{b}_h$ is the bias. The target output $\mathbf{y}_t$ is given by $\mathbf{y}_t = \mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o$. Consider a linear activation function $\mathcal{H}(\mathbf{x}) = \mathbf{x}$ and subsume the bias term $\mathbf{b}_h$ into the hidden state $\mathbf{h}$, Equation (2) can be simplified to

$$\mathbf{h}_t = \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{ih}\mathbf{x}_t. \tag{3}$$

Note the similarity between Equations (1) and (3): the optimal estimate of the state $\mathbf{h}_t$ is a weighted linear combination of the estimate of the previous state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$. The two weight matrices are $\mathbf{W}^t_{bh}$ and $\mathbf{W}^t_{bi}$ for the Kalman filter, and $\mathbf{W}_{hh}$ and $\mathbf{W}_{ih}$ for RNN.

Table 1: Connections between Bayesian filters and RNN

	Bayesian filters	RNN
Input	Measurements $\{\mathbf{x}_t\}$ & Models $p(\mathbf{h}_t\|\mathbf{h}_{t-1}), p(\mathbf{x}_t\|\mathbf{h}_t)$	Measurements $\{\mathbf{x}_t\}$ & Training Data $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{\text{train}}$
Output	Estimates of $\{(\mathbf{h}_t, \mathbf{y}_t)\}$	Estimates of $\{(\mathbf{h}_t, \mathbf{y}_t)\}$ & $\mathbf{W}_{hh}$, $\mathbf{W}_{ih}$
Training	No	Yes
Applicability	Challenging	Easy

This relationship between the input $\mathbf{x}_t$ and the estimate of the state $\mathbf{h}_t$ also applies to other variants of Bayesian filters and RNNs. One obvious difference is that for the Kalman filter (and other Bayesian filters), the two weight matrices change over time, indicating that it is an adaptive estimator. While for RNN, after the training stage, the two learned weight matrices $\mathbf{W}_{hh}$ and $\mathbf{W}_{ih}$ are usually fixed.

In practice, there are two other important differences. Firstly, for Bayesian filters, most effort goes into designing the state transition and measurement models $p(\mathbf{h}_t|\mathbf{h}_{t-1})$ and $p(\mathbf{x}_t|\mathbf{h}_t)$ and tuning the parameters (*e.g.*, $\mathbf{\Sigma}_p$ and $\mathbf{\Sigma}_m$), which is usually challenging for complex tracking tasks (*e.g.*, non-rigid tracking of faces). RNN is more generally applicable to almost any tracking task, since the optimal parameters, $\mathbf{W}_{hh}$ and $\mathbf{W}_{ih}$, can be learned from the training data. Secondly, integrating Bayesian filters with the static estimators for generic vision tasks is also challenging, as discussed previously. Instead, RNN can be easily concatenated with a CNN that performs frame-wise feature extraction, and forms an end-to-end network for joint estimation and tracking. We show the effectiveness of this end-to-end network in the experiments on head pose estimation and facial landmark localization in videos. Table 1 summarizes the connections between Bayesian filters and RNN.

3.1. A Toy Problem

We first use an expository example to illustrate the similarity and difference between Bayesian filtering and RNN. Assume that a cursor moves in a sinusoidal pattern in one dimension. The measurement $\mathbf{x}_t$ is the observed position of the cursor with added Gaussian noise. The goal is to estimate the true position $\mathbf{y}_t$. We use a Kalman filter as an example of Bayesian filtering for this task. The state $\mathbf{h}_t$ is defined to be the position, velocity, and acceleration of the cursor. The matrices $\mathbf{W}$ and $\mathbf{V}$ are designed according to the kinematic equations, and $\mathbf{\Sigma}_p$ and $\mathbf{\Sigma}_m$ are estimated as in [25]. Figure 2 shows two examples, where the black circles are the inputs $\mathbf{x}_t$ and the blue curves are the targets $\mathbf{y}_t$. The output of the Kalman filter, shown as the green curves, smooths the noisy input as expected. We also use RNN to

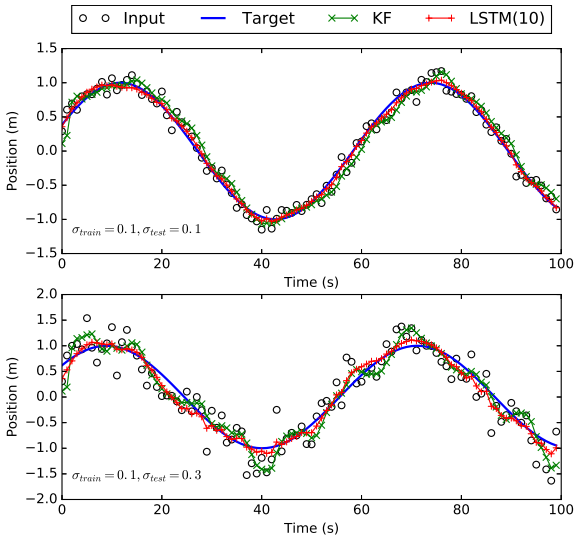

Figure 2: Results on a toy problem (tracking a moving cursor in 1D) for Kalman filter and RNN. The RNN (LSTM with 10 neurons) results in more accurate regression than the Kalman filter, even when the testing sequence has a different noise level from the training sequences.

solve this problem. We generate 1000 sequences with random additive Gaussian noise with $\sigma_{train} = 0.1$ as the training data. The input to the RNN is $\mathbf{x}_t$ and its output is the estimate of $\mathbf{y}_t$. We train an LSTM with one layer of 10 neurons and an ℓ_2 loss. As shown in Figure 2, the regression results of RNN are more accurate than those of the Kalman filter, even when the testing sequence has a different noise level versus the training sequence.

3.2. RNN For Dynamic Facial Analysis

We now apply RNN for facial analysis in videos. As shown in Figure 3, we study four variants of the proposed method for head pose estimation and facial landmark localization in videos. Figure 3(a) is the baseline method where per-frame estimation is conducted by CNN and the temporal connection is ignored, similar to the recently proposed deep learning-based methods for head pose estimation [30] and facial landmark localization [40]. For temporally filtering the per-frame results, we either use Bayesian filters (KF and PF) for post-processing in Figure 3(b), or perform Post-RNN in Figure 3(c) by using RNN to map from the sequence of per-frame estimates to the sequence of known ground truth. Finally, we learn an end-to-end network, which includes both the CNN and RNN for joint estimation and tracking in Figure 3(d).

RNN, in most vision tasks, is built upon a CNN pre-trained from large-scale datasets [19]. In all of our experiments the CNN is modified from VGG16 [45] by adding one additional `fc` layer. However, in traditional RNN, as in Equation (2), both $\mathbf{W}_{ih}$ and $\mathbf{W}_{hh}$ are randomly initialized. So a recurrent layer is required to be trained from scratch even if a pre-trained CNN is used for feature extraction. FC-RNN was recently proposed in [56] to leverage the important generalization property of a pre-trained CNN. FC-RNN transforms the `fc` layers of a pre-trained CNN into recurrent layers with the intention of preserving the structure of a pre-trained network as much as possible. Suppose that a pre-trained `fc` layer at timestamp t has the structure:

$$\mathbf{f}_t = \mathcal{H}(\mathbf{W}_{io}\mathbf{x}_t + \mathbf{b}_f), \tag{4}$$

where $\mathbf{W}_{io}$ is the pre-trained input-to-output matrix, $\mathbf{x}_t$ is the output of the previous feed-forward layer and $\mathbf{b}_f$ is the bias. FC-RNN transforms it into a recurrent layer through:

$$\mathbf{f}_t = \mathcal{H}(\mathbf{W}_{io}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{f}_{t-1} + \mathbf{b}_f). \tag{5}$$

This recurrent structure, initialized with the `fc` layer of the CNN, only introduces a single hidden-to-hidden weight matrix $\mathbf{W}_{hh}$, which needs to be trained from scratch while the other weight matrices are pre-trained and can be merely fine-tuned.

We adopt FC-RNN in our experiments for dynamic facial analysis because of its simple structural design and superior or on a par performance to other more complex variants of recurrent networks. We apply a set of regularization techniques to train the FC-RNN. We first use variational dropout [14] to repeat the same dropout mask with 0.25 dropout rate at each time step for both the feed-forward and recurrent connections. This is in contrast to other dropout schemes where different dropout masks are sampled at each time step for feed-forward connections only and no dropout is used with recurrent connections. We also apply soft gradient clipping to prevent gradients from exploding in the

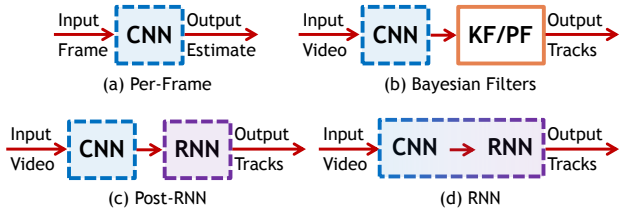

Figure 3: Definitions of the methods used in this paper. (a) We use a CNN for per-frame estimation as our baseline. (b) a Kalman filter (KF) or a particle filter (PF) is built upon the per-frame estimation results for temporal filtering. (c) For Post-RNN, the inputs to the RNN are the per-frame estimation results, and thus the RNN is used in the place of the KF/PF for temporal filtering. (d) The proposed RNN is an end-to-end network containing a CNN and an RNN for joint estimation and tracking. In (a)-(c), the CNN is trained separately, whereas in (d) it is trained end-to-end together with the RNN.

recurrent layers, *i.e.*, if the ℓ_2-norm of gradients $\|g\|$ is larger than a threshold $\tau = 10$, we rescale the gradients to $g \leftarrow g\tau/\|g\|$.

4. Head Pose Estimation in Videos

Our first application for dynamic facial analysis is head pose (*i.e.*, pitch, yaw, and roll angles) estimation in videos. We conduct experiments on two datasets to evaluate the different methods. First, we create a large-scale synthetic head pose dataset, SynHead, which contains 10 subjects, 70 motion tracks, and 510,960 frames in total. Second, we experiment with the BIWI dataset [13], which includes 20 subjects, 24 videos, and 15,678 frames in total. Finally, we fine-tune the model trained on the SynHead dataset, with the training data from the BIWI dataset and obtain significant improvement in performance. We modified our network from VGG16 by adding one more `fc` layer with 1024 neurons and by changing the output layer to 3 neurons corresponding to the pitch, yaw and roll angles. We applied FC-RNN to model the temporal evolution of the sequence of measurements and used the ℓ_2 loss function for training.

4.1. SynHead Dataset and Results

There are two motivations for creating the synthetic head pose dataset. (1) While quite a few datasets are available for head pose estimation from still images such as AFLW [23] and LFW [18], there are very limited video-based datasets. The Oxford dataset [3] has low spatial resolution, and the ETH dataset [4] has only depth images. Only the BIWI dataset [12] is suitable for our task. (2) Due to various difficulties in ground truth collection, head pose datasets usually have errors and noises in the ground truth annotations. For example, the BIWI [12] dataset has, on an average, 1 degree of error. A synthetic dataset with accurate ground truth annotation is therefore desirable for algorithm evaluation.

Figure 4 demonstrates the pipeline for creating this large-scale synthetic head pose dataset. The 10 (5 female and 5 male) 3D head models are high resolution 3D scans from [1]. To simulate realistic head motions, we gather head motion tracks, 24 from the BIWI and 26 from the ETH dataset. Additionally, we recorded 20 depth video sequences performed by 13 (11 male and 2 female) subjects with the Kinect and SoftKinetic sensors. We compute the raw head pose angles for these sequences with [28] and discard any failure cases by manual inspection. Finally, we temporally smoothen these head motion tracks with a Gaussian filter. In all, there are 70 motion tracks with 51,096 different head poses. The distributions of head poses in yaw, pitch, and roll angles of this dataset are shown in Figure 4.

For each head pose, we render the 10 head models and compose the rendered images with a randomly selected background image per motion track. We find that adding the random background is an efficient way to augment the dataset and helpful for good performance. Examples of the composed images are shown in Figure 4. An example of a synthesized track is in the supplementary material.

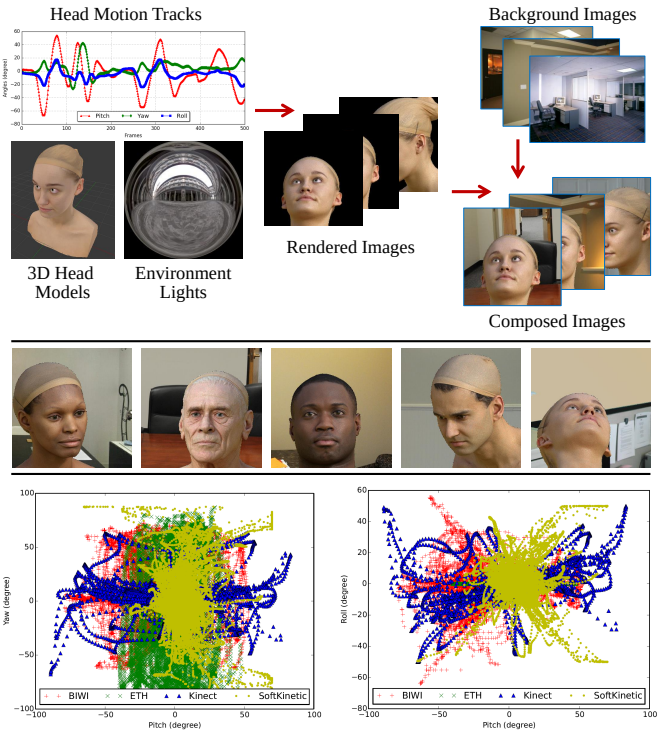

Figure 4: SynHead dataset. Top: The pipeline for rendering the SynHead dataset with given motion tracks, 3D head models and background images. Middle: Examples of the rendered images. Bottom: Distributions of the pitch-yaw and pitch-roll angles.

Table 2: The errors for the pitch, yaw, and roll angles (°) on the SynHead dataset: average (top three rows) and standard deviation (bottom three rows).

Err/Std	Per-Frame	KF	PF	Post-RNN	RNN
pitch	1.94	1.92	2.16	1.84	**1.55**
yaw	2.63	2.62	2.80	2.15	**1.78**
roll	2.15	2.14	2.35	2.11	**1.66**
pitch	1.94	1.88	2.43	1.89	**1.51**
yaw	2.72	2.66	2.82	2.70	**2.32**
roll	3.10	3.16	3.30	3.08	**2.37**

In our experiments, we randomly select 8 subjects and 46 motion tracks from the ETH, Kinect, and SoftKinectic datasets for training, as well as 2 subjects and 24 motion tracks from BIWI dataset for testing. This ensures that no overlap of the 3D head models or motion tracks exists between training and test sets. Table 2 summarizes the average head pose estimation error and its standard derivation for the pitch, yaw, and roll angles. An example sequence of the estimated head poses, by different methods, is given

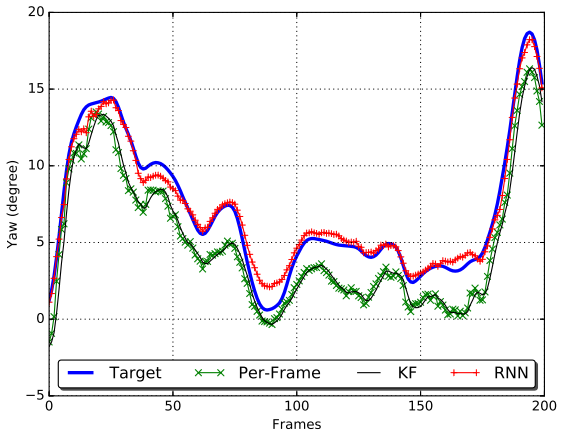

Figure 5: An example sequence from the SynHead dataset with its estimated head pose (yaw). The proposed end-to-end RNN algorithm produces more accurate results than the per-frame estimates (with or without Kalman filtering).

in Figure 5. It demonstrates that the end-to-end RNN approach not only reduces the estimation error, but also generates a smoother track over time, indicating that it learns the temporal variation of head poses in videos. In contrast, Kalman filtering (and similarly particle filtering) can only reduce the variability/noise in the per-frame estimates over time, but cannot reduce their estimation errors.

4.2. Results on BIWI Dataset

We also evaluate our methods on real data from the BIWI dataset which contains 24 videos in all. We follow the experimental protocol proposed previously in [30] and split the dataset into 70% for training (16 videos) and 30% for testing (8 videos). We have three such splits and we report the measurement errors averaged across them. As shown in Table 3, we have the same conclusions as those from the SynHead dataset. Among all the five algorithms, the end-to-end RNN approach performs the best in not only the average error but also the standard deviation of the estimated error, which shows that the estimation with the end-to-end RNN method is more stable over time.

Table 4 shows the comparison between our algorithm and the state-of-the-art methods on the BIWI dataset. Our approach with only RGB images achieve superior performance over the two learning-based methods [13, 30] which rely on both RGB and depth images.[1] Moreover, after fine-tuning the model trained on the SynHead dataset (excluding the BIWI motion tracks), on the BIWI dataset, we further reduce the average angular error to around 1.5 degrees. This validates the effectiveness of the proposed method for head pose estimation with large training data obtained by image synthesis. Figure 6 shows several examples of head

[1]The 3DModel approach [28] requires no training and is evaluated on the entire BIWI dataset.

Figure 6: Examples of head pose estimation on the BIWI dataset with the RNN and per-frame algorithms. RNN outperforms the per-frame estimation for various head poses.

Table 3: The errors for the pitch, yaw, and roll angles (°) on the BIWI dataset: average (top three rows) and standard deviation (bottom three rows).

Err/Std	Per-Frame	KF	PF	Post-RNN	RNN
pitch	4.03	4.12	4.32	3.90	**3.48**
yaw	3.91	4.15	4.22	3.78	**3.14**
roll	3.03	3.09	3.19	2.98	**2.60**
pitch	3.61	3.70	3.99	3.53	**2.89**
yaw	3.82	3.93	4.11	3.56	**3.12**
roll	3.05	3.11	3.34	3.05	**2.76**

pose estimation on the BIWI dataset. RNN outperforms the per-frame estimation for various head poses. A comparison video is in the supplementary material.

5. Facial Landmark Localization in Videos

Our second application for dynamic facial analysis is facial landmark localization in videos. We experiment with the recently released 300-VW [20] benchmark dataset. It contains 114 videos and a total of 218,595 frames, with 68 facial landmark annotations per frame. As a pre-processing step, we train a face detector with Faster R-CNN [42] to perform face detection on every frame. For each video, the central positions of the detected facial regions are smoothed temporally with a Gaussian filter, and the maximum size of the detected bounding boxes is used to extract a face-centered sequence. This pre-processing step stabilizes face detections over time and interpolates face regions for the few frames with missed face detection.

We employ several types of data augmentation—horizontal mirroring of the images, playing the image sequences in reverse, and small random scaling and translation of the face windows. We use the same network architecture and ℓ_2 loss function as for head pose estimation

Table 4: A comparison of the averaged angular errors (°) for the state-of-the art methods on the BIWI dataset.

Error	DeepHeadPose [30] (RGB+D)	Martin [26] (RGB+D)	3DModel [28] (Depth)	DeepHeadPose [30] (RGB)	Ours (RGB)	Ours (RGB+SynHead)
pitch	4.76	2.5	2.1	5.18	3.48	**1.35**
yaw	5.32	3.6	2.1	5.67	3.14	**1.54**
roll	-	2.6	2.4	-	2.60	**1.35**

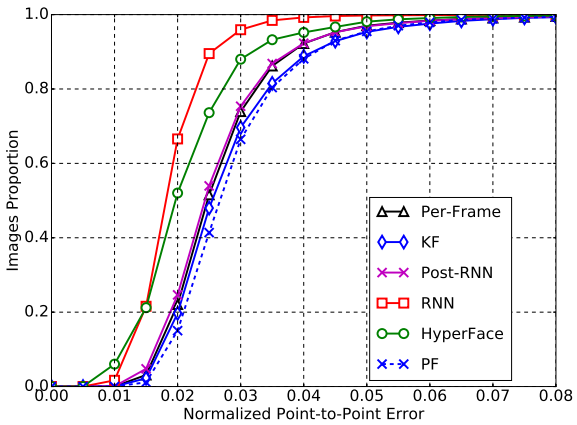

Figure 7: The cumulative error distribution plot on 300-VW (split3). More plots are in the supplementary material.

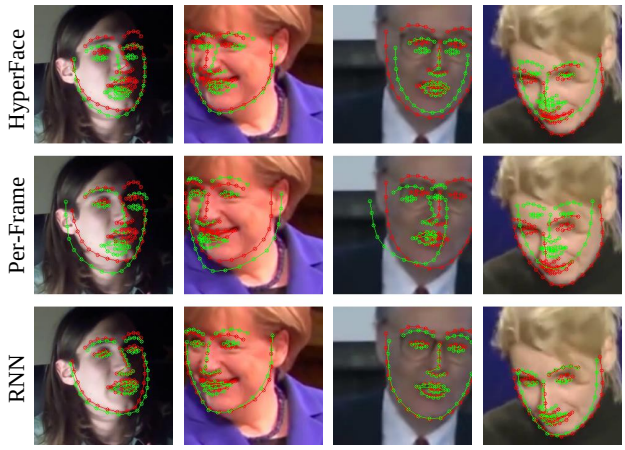

Figure 8: Example results of facial landmark localization. The ground truth is in red and the estimation is in green. The proposed RNN-based method outperforms HyperFace [40] and our per-frame estimation, especially for challenging conditions, *e.g.*, head/camera motion, uneven illumination, facial expressions, and occlusions. A video with more comparisons is in the supplementary material.

except that the output layer has 136 neurons corresponding to the locations of the 68 facial landmarks.

We randomly split the 300-VW dataset three times into 80% (91 videos) for training and 20% (23 videos) for testing. For quantitative performance evaluation, we use the same error metric and statistical measures as defined in [7]. For each frame, we compute the normalized point-to-point error e_n which is the mean Euclidean distance of the 68 points, normalized by the diagonal distance of the ground truth bounding box. The cumulative error distribution plot (*i.e.*, the proportion of frames with the normalized point-to-point error e_n less than a given threshold) is used to compare the performance. Two statistical measures, *i.e.*, area under the curve (AUC) and failure rate (FR), are also used for evaluation. AUC is the area under the curve of the cumulative error distribution plot for $0 < e_n \leq 0.08$. The proportion of frames with $e_n > 0.08$ is defined as FR, which is simply the percentage of failure cases for the given task.

Table 5 summarizes the results of the different methods on the three splits. As shown in this table, the RNN-based methods, including Post-RNN and RNN, improve the performance of per-frame estimation. In addition, compared to the recent work HyperFace [40], which is a multi-tasking network for frame-wise facial analysis and has state-of-the-art performance, our RNN-based method has better performance, demonstrating the effectiveness of joint estimation and tracking. Finally, we observe that RNN significantly reduces the failure rate compared to other methods. The Kalman filter (KF) or the particle filter (PF), apart from reducing the noise on the observed measuerements, they are not able to reduce the observed error from the ground truth, which is also consistent with the findings of [7]. Figure 7 shows the cumulative error distribution plots for one split. More plots are in the supplementary material.

Figure 8 shows examples of facial landmark localization on the 300-VW dataset. Compared to per-frame estimation, RNN is more robust to occlusions, large head poses and facial expressions. In particular, it is more advantageous when the head is moving fast, which shows the ability of RNN to learn motion information implicitly. A video with more comparisons is in the supplementary material.

We also compared with a recent work [37] that uses RNN for face landmark detection from videos. We note that [37] was specifically designed for landmark estimation, and its computational cost is about eight times higher than ours, because [37] used the spatial recurrent learner for progressive refinement and used VGG16 for both the encoder and decoder. On the 300-VW dataset, with the same splits and error metric, Table 7 shows our method has better accuracy.

Finally, we perform another experiment to compare with the top ranking methods from the 300-VW challenge [20]. This challenge uses 50 videos for training, and splits the

Table 5: Comparison of the variants of our methods (*i.e.*, Per-Frame estimation, Kalman filter (KF) and particle filter(PF), Post-RNN, and end-to-end RNN) and HyperFace [40] on the 300VW dataset for the three 80%:20% splits. The areas under curve (AUC) and failure rates (FR) are reported.

Method	Split 1		Split 2		Split 3	
	AUC	FR(%)	AUC	FR(%)	AUC	FR(%)
Per-Frame	0.66	2.12	0.71	0.14	0.67	0.49
KF	0.63	2.65	0.70	0.43	0.65	0.68
PF	0.65	2.20	0.69	0.25	0.64	0.59
Post-RNN	0.66	2.16	0.72	1.29	0.67	0.51
RNN	**0.74**	**0.28**	**0.77**	**0.01**	**0.77**	**0.04**
HyperFace [40]	0.73	1.34	0.74	0.09	0.73	5.56

Table 6: Comparison against the participants of the 300VW challenge [20] and HyperFace [40]. The areas under curve (AUC) and failure rates (FR) are reported.

Method	Category 1		Category 2		Category 3	
	AUC	FR(%)	AUC	FR(%)	AUC	FR(%)
Ours (RNN)	0.718	**1.20**	0.703	**0.20**	0.617	4.83
HyperFace [40]	0.642	5.56	0.662	0.68	0.563	7.23
Yang *et al*. [55]	0.791	2.400	0.788	0.322	0.710	**4.461**
Uricar and Franc [49]	0.657	7.622	0.677	4.131	0.574	7.957
Xiao *et al*. [53]	0.760	5.899	0.782	3.845	0.695	7.379
Rajamanoharan and Cootes [39]	0.735	6.557	0.717	3.906	0.659	8.289
Wu and Ji [51]	0.674	13.925	0.732	5.601	0.602	13.161

Table 7: Comparison with [37]: mean error on 300-VW

Split 1	Split 2	Split 3	Average	[37]
5.54%	6.74%	6.22%	**6.16%**	6.25%

remaining 64 videos into three categories (based on image complexity) for testing. It is more challenging to train networks on this setting, because it has much less training data than the previous splits (50 videos vs. 91 videos), and the prior distributions of the training data and the testing data are quite different.

We still achieve superior performance, especially in the failure rate (FR), as summarized in Table 6. FR is the percentage of images whose errors are larger than a given threshold. FR is an important measure for applications where very small errors in landmark locations are not as critical as the success rate of a system, such as for face recognition and facial expression understanding. Our method significantly outperforms HyperFace which is also a deep-learning-based approach for generic facial analysis. Other competing algorithms require complicated procedures designed specifically for landmark localization, such as semantic facial part detection [53], shape-space division of heads [39], head transformation tracking and reinitialization [55]. In contrast, our method is a generalized end-to-end approach that can be easily adapted to different applications of dynamic facial analysis. As we demonstrated for head pose estimation (Section 4 and Table 4), we believe that with more training data, either real or synthetic, we can expect large improvements in the performance of facial landmark localization as well.

6. Conclusion and Discussion

In this paper, we have proposed and evaluated an RNN-based method for dynamic facial analysis, including for head pose estimation and facial landmark localization from videos. In comparison to traditional Bayesian filters, the RNN-based method learns to jointly estimate the frame-wise measurements and to temporally track them with a single end-to-end network. Moreover, it does not rely on complicated and problem-specific *tracker-engineering* or *feature-engineering*, which are required in prior methods. This makes the RNN-based method a generic approach that can be extended to other tasks of facial analysis in videos, beyond the ones that we demonstrated in this paper.

There are several directions we would like to explore in the future. First, we demonstrated that for head pose estimation, a large-scale synthetic dataset significantly improves the performance of learning-based methods. We will work more along this direction and evaluate its benefits for other dynamic facial analysis tasks. Second, Bayesian filters have the ability to adapt their estimation models over time as new data arrives, while the current RNN is fixed once the training phase is finished. Thus another direction to explore is online learning of the RNN.

References

[1] http://www.3dscanstore.com/. 5

[2] T. Baltrusaitis, P. Robinson, and L. Morency. 3d constrained local model for rigid and non-rigid facial tracking. In *CVPR*, pages 2610–2617, 2012. 2

[3] B. Benfold and I. Reid. Colour invariant head pose classification in low resolution video. In *BMVC*, 2008. 5

[4] M. D. Breitenstein, D. Kuettel, T. Weise, L. Van Gool, and H. Pfister. Real-time face pose estimation from single range images. In *CVPR*, pages 1–8, 2008. 5

[5] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. *International Journal on Computer Vision (IJCV)*, 2014. 2

[6] K. Cho, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*, 2014. 2

[7] G. G. Chrysos, E. Antonakos, P. Snape, A. Asthana, and S. Zafeiriou. A comprehensive performance evaluation of deformable face tracking in-the-wild. *ArXiv preprint arXiv:1603.06015*, 2016. 1, 2, 7

[8] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models – their training and application. *Computer Vision and Image Understanding*, 1995. 2

[9] T. F. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2001. 2

[10] D. F. Dementhon and L. S. Davis. Model-based object pose in 25 lines of code. *International journal of computer vision*, 15(1-2):123–141, 1995. 2

[11] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015. 2

[12] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool. Random forests for real time 3d face analysis. *Int. J. Comp. Vision*, 101(3):437–458, 2013. 5

[13] G. Fanelli, J. Gall, and L. V. Gool. Real time head pose estimation with random regression forests. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 2, 5, 6

[14] Y. Gal. A theoretically grounded application of dropout in recurrent neural networks. *arXiv*, 2015. 4

[15] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013. 2

[16] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel. Backprop kf: Learning discriminative deterministic state estimators. In *NIPS*, 2016. 2

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 2

[18] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007. 5

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 2014. 4

[20] J.Shen, S.Zafeiriou, G. S. Chrysos, J.Kossaifi, G.Tzimiropoulos, and M. Pantic. The first facial landmark tracking in-the-wild challenge: Benchmark and results. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2015. 2, 6, 7, 8

[21] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 3

[22] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[23] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011. 5

[24] R. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. *Arxiv print. Arxiv 1511.05121*, 2015. 2

[25] R. Labbe. *Kalman and Bayesian Filters in Python*. 2015. 3

[26] M. Martin, F. V. D. Camp, and R. Stiefelhagen. Real time head model creation and head pose estimation on consumer depth cameras. In *3DV*, 2014. 7

[27] M. Martin, F. Van De Camp, and R. Stiefelhagen. Real time head model creation and head pose estimation on consumer depth cameras. In *3DV*, volume 1, pages 641–648, 2014. 2

[28] G. P. Meyer, S. D. Mello, I. Frosio, D. Reddy, and J. Kautz. Robust model-based 3d head pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2, 5, 6, 7

[29] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[30] S. S. Mukherjee and N. Martin. Deep head pose: Gaze-direction estimation in multimodal video. *IEEE Transactions on Multimedia*, 2015. 2, 4, 6, 7

[31] E. Murphy-Chutorian and M. M. Trivedi. HyHOPE: Hybrid head orientation and position estimation for vision-based driver head tracking. In *IEEE Intelligent Vehicles Symposium*, 2008. 1

[32] E. Murphy-Chutorian and M. M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2009. 1

[33] E. Murphy-Chutorian and M. M. Trivedi. Head pose estimation and augmented reality tracking: An integrated system and evaluation for monitoring driver awareness. *IEEE Transactions on intelligent transportation systems*, 11(2):300–311, 2010. 1, 2

[34] J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2

[35] P. Padeleris, X. Zabulis, and A. A. Argyros. Head pose estimation on depth data based on particle swarm optimization. In *CVPRW*, pages 42–49, 2012. 2

[36] C. Papazov, T. K. Marks, and M. Jones. Real-time 3d head pose and facial landmark estimation from depth images using triangular surface patch features. In *CVPR*, pages 4722–4730, 2015. 2

[37] X. Peng, R. S. Feris, X. Wang, and D. N. Metaxas. A recurrent encoder-decoder network for sequential face alignment. In *ECCV*, 2016. 2, 7, 8

[38] U. Prabhu, K. Seshadri, and M. Savvides. Automatic facial landmark tracking in video sequences using kalman filter assisted active shape models. In *ECCV 2010 Workshops*, 2012. 1, 2

[39] G. Rajamanoharan and T. Cootes. Multi-view constrained local mmodel for large head angle face tracking. In *IEEE Proceedings of International Conference on Computer Vision, 300 Videos in the Wild (300-VW): Facial Landmark Tracking in-the-Wild Challenge & Workshop (ICCV-W)*, 2015. 2, 8

[40] R. Ranjan, V. M. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *arXiv preprint arXiv:1603.01249*, 2016. 2, 4, 7, 8

[41] A. Rekik, A. Ben-Hamadou, and W. Mahdi. 3d face pose tracking using low quality depth cameras. In *VISAPP*, pages 223–228, 2013. 2

[42] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015. 6

[43] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: Particle filters for tracking applications*, volume 685. Artech house Boston, 2004. 3

[44] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 face in-the-wild challenge: the first facial landmark localization challenge. In *ICCV-W: 300 Faces In-the-Wild Challenge (300-W)*, 2013. 2

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[46] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011. 2

[47] S. Tulyakov, R.-L. Vieriu, S. Semeniuta, and N. Sebe. Robust real-time extreme head pose estimation. In *ICPR*, pages 2263–2268, 2014. 2

[48] G. Tzimiropoulos and M. Pantic. Gauss-newton deformable part models for face alignment in-the-wild. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[49] M. Uricar and V. Franc. Real-time facial landmark tracking by tree-based deformable part model based detector. In *IEEE Proceedings of International Conference on Computer Vision, 300 Videos in the Wild (300-VW): Facial Landmark Tracking in-the-Wild Challenge & Workshop (ICCV-W)*, 2015. 1, 2, 8

[50] G. Welch. The use of the kalman filter for human motion tracking in virtual reality. *Presence: Teleoperators and Virtual Environments*, 18(1):72–91, 2009. 1

[51] Y. Wu and Q. Ji. Shape augmented regression method for face alignment. In *IEEE Proceedings of International Conference on Computer Vision, 300 Videos in the Wild (300-VW): Facial Landmark Tracking in-the-Wild Challenge & Workshop (ICCV-W)*, 2015. 2, 8

[52] S. Xiao, J. Feng, J. Xing, H. Lai, S. Yan, and A. Kassim. Robust facial landmark detection via recurrent attentive-refinement networks. In *ECCV*, 2016. 2

[53] S. Xiao, S. Yan, and A. Kassim. Facial landmark detection via progressive intialization. In *IEEE Proceedings of International Conference on Computer Vision, 300 Videos in the Wild (300-VW): Facial Landmark Tracking in-the-Wild Challenge & Workshop (ICCV-W)*, 2015. 2, 8

[54] X. Xiong and F. D. la Torre. Supervised descent method and its aapplication to face alignment. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[55] J. Yang, J. Deng, K. Zhang, and Q. Liu. Facial shape tracking via spatial-temporal cascade shape regression. In *IEEE Proceedings of International Conference on Computer Vision, 300 Videos in the Wild (300-VW): Facial Landmark Tracking in-the-Wild Challenge & Workshop (ICCV-W)*, 2015. 1, 2, 8

[56] X. Yang, P. Molchanov, and J. Kautz. Multilayer and multimodal fusion of deep neural networks for video classification. In *ACM Multimedia*, 2016. 4