2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Depth-based 3D Hand Pose Tracking

Kha Gia Quach*, Chi Nhan Duong*, Khoa Luu† and Tien D. Bui*
*Department of Computer Science and Software Engineering
Concordia University, Montreal, Quebec, Canada
Email: {k_q, c_duon, bui}@encs.concordia.ca
† CyLab Biometrics Center and the Department of Electrical and Computer Engineering,
Carnegie Mellon University, Pittsburgh, PA, USA
Email: kluu@andrew.cmu.edu

***Abstract*—In this paper, we propose two new approaches using the Convolution Neural Network (CNN) and the Recurrent Neural Network (RNN) for tracking 3D hand poses. The first approach is a detection based algorithm while the second is a data driven method. Our first contribution is a new tracking-by-detection strategy extending the CNN based single frame detection method to a multiple frame tracking approach by taking into account prediction history using RNN. Our second contribution is the use of RNN to simulate the fitting of a 3D model to the input data. It helps to relax the need of a carefully designed fitting function and optimization algorithm. With such strategies, we show that our tracking frameworks can automatically correct the fail detection made in previous frames due to occlusions. Our proposed method is evaluated on two public hand datasets, i.e. NYU and ICVL, and compared against other recent hand tracking methods. Experimental results show that our approaches achieve the state-of-the-art accuracy and efficiency in the challenging problem of 3D hand pose estimation.**

I. Introduction

The problem of human hand pose estimation has been studied for decades. In particular, markless hand tracking has gained a lot of attention due to its numerous applications in human-computer interaction with virtual augmented reality, 3D gaming, gestural inputs, etc. Major progresses have been made recently in hand pose estimation thank to the depth sensors and deep learning models [1]. However, the problem is still far from solved. The challenges remain but shift toward dealing with occlusions, low-resolutions, clutters, etc. In addition, real-time requirement makes the problem even harder. This is because natural hand movement exhibits many degrees of freedom ($\geq 25^o$), fast motions with rapid changes in direction, self-similar parts and self-occlusion. All these factors make estimating the pose of articulated objects from depth video data hard, unless the desired method is robust against such self-similarity and self-occlusion.

A. Motivation

There are two common approaches in the problem of hand pose estimation, i.e. the model-base generative tracking and the discriminative hand pose detection. Both approaches have their drawbacks, e.g. tracking framework cannot recover from initial error while detectors may fail in case of occlusions. Thus, the motivation of our work is to develop new strategies to combine the benefits of both generative tracking and discriminative detection into a unified framework that yields high efficiency and robust performance. In addition, this paper presents two ways to build a hybrid framework that can track human hand with a single depth-image sequence.

B. Contributions

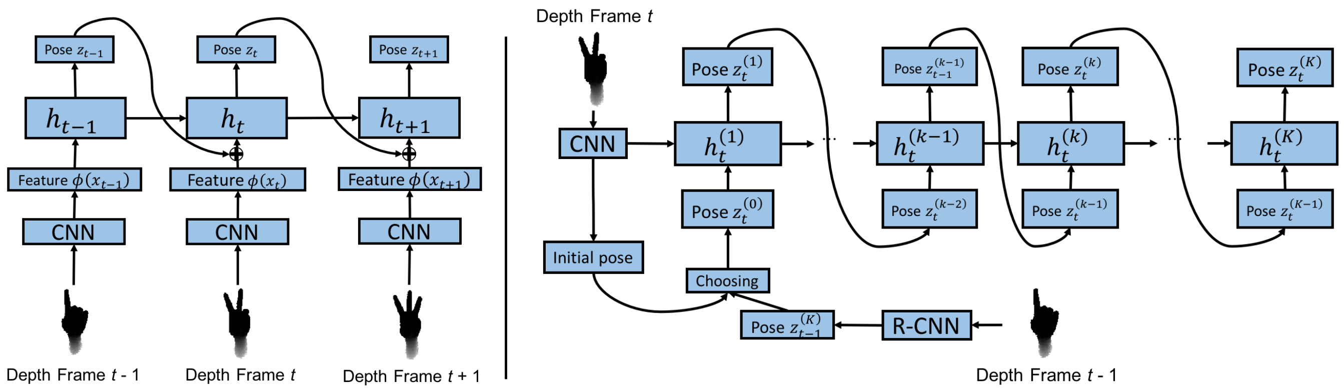
In this paper, we propose a hand pose tracking framework based on CNN and RNN. The first contribution in our first strategy is a new tracking-by-detection strategy which combines the Convolutional Neural Networks (CNN) and the Recurrent Neural Networks (RNN) inspired from the work on visual object tracking [2]. In this approach, we take into account prediction history to extend single frame detection into a multiple frame tracking method. The RNN outputs hand pose prediction computed based on the visual features generated by the CNN and the memory state of RNN, as shown in Fig. 1(a). In other words, the RNN plays a role of memorizing hand pose prediction from previous frames. The results of CNN-based hand pose estimation are now affected by previous frames rather than the current input frame solely. Finally, the prediction can optionally emphasize attention areas in the input before feeding it into convolutional network in the next frame.

The second contribution is a new generative model-based tracking where each time step in RNN can be seen as efficient approximate model fitting step in traditional model-based tracking. The similarity function is learned from training data and integrated into the learning of RNN. CNN is for extracting features from the current depth-frames and producing a good initial hand pose if necessary, as shown in Fig. 1(b). Our methods are able to recover from error, fast, reliable and robust against occlusions.

The rest of the paper is organized as follows. First, we review two important categories of hand pose estimation methods in Section II, i.e. model-based tracking and discriminative-based detection. In Section III, we present a novel hand pose tracking framework followed by a description of the settings for training the model in Section III-B3. Results and analysis are given in Section IV. Finally, the conclusion in this paper is presented in Section V.

II. Previous Work

There has been a variety of approaches proposed over the last decade for hand pose estimation. These techniques rely on markers or wearable gloves, RGB input from single or



(a) Tracking-by-detection approach

(b) Model-based tracking approach

Fig. 1. The diagram of our hand pose tracking models

multiple cameras, and more recently depth camera or RGB-D input. Good overviews of earlier and recent work are given in [3] and [1]. In this section, we will review only more recent work relevant to our approach based on a single depth-frame captured by depth camera. Hand-pose estimation methods can be divided into two main categories: appearance-based (discriminative) methods and model-based (generative) methods.

The first group of approaches is based on discriminative models that aim at directly predicting the joint locations from RGB-D or depth images. These models are often used in detection-based framework. Some popular architectures in this group include *decision forests* [4], [5], [6], [7], [8], [9], *part-based models* [10], and *deep models*. We will focus on *deep models* and readers can refer to [1] for an overview of other detection-based approaches. Follow the current trend in Computer Vision, deep neural nets have been explored for hand pose estimation. Tompson et al. [11] proposed a three-stage pipeline including hand detection with a decision forest, joint location estimation with a deep network, and joint refinement with inverse kinematics (IK). They also introduced a large-scale dataset for training and evaluation called **NYU** hand dataset, so this paper/method is referred as **DeepNYU** [11]. Oberweger et al. [12] employed a similar deep network but proposed to learn a prior subspace to replace the IK stage. Farabet et al. [13] developed a pixel-labeling approach which predicts joint labels for each pixel and followed by a clustering stage to predict joint locations. This approach is inspired from a similar method in human pose recognition problem [14] but a deep network was used instead of a decision forest.

The second group consists of generative, model-based methods. This type of method usually has four main building blocks: a hand model, a similarity function measuring the difference between the observed image and the model, an optimization algorithm (e.g. Particle Swarm) to maximize the similarity function with respect to the model parameters, and an initial pose for the optimization starting point. This model-driven system usually found in tracking-based domain [15], [16], [17], [18], [19].

Oikonomidis et al. [15] proposed a model-based method using particle swarm optimization (PSO). This method achieves 15 fps with GPU acceleration and uses skin color segmentation which is sensitive to lighting. Oikonomidis et al. [20] later presented a more advanced sampling strategy that improves tracking efficiency without compromising quality. However, gradient-based optimization approaches converge faster and more accurately than PSO when close to the solution, and are therefore well suited for realtime applications. Melax et al. [16] proposed a tracking method based on depth by using efficient parallel physics simulations. Although this method is fast, finger articulations are often incorrectly tracked. Oberweger et al. [17] proposed a model-based method, called **DeepPrior**, that can learn to generate realistic depth images of hands and to predict updates that improve the current estimate of the hand pose given the input depth image and a generated image for this estimate. The method uses a feedback loop which is constructed by Deep Networks learned from training data rather than a carefully designed similarity function and an optimization algorithm. Paliouras et al. [19] aim at automating the definition of the objective function in the optimization problem of model-based methods. All these works require careful initialization in order to guarantee convergence and therefore rely on tracking based on the last frames' pose or separate initialization methods. Such tracking-based methods have difficulty handling drastic changes between two frames, which are common as the hand tends to move fast.

There have been some works trying to combine both discriminative and generative models into a unified framework [21] [22] [23]. Qian et al. [21] proposed a hybrid method that uses a simple hand model consisting of a number of spheres in combination with discriminative fingertip detection. This method achieves 25 fps but requires the fingertips to be visible, otherwise tracking would be hard since it could easily be occluded or misdetected.

III. Our Proposed Hand Pose Tracking Framework

In this section, we first present a tracking-by-detection approach to hand-pose tracking problem using deep learning techniques [24], particularly CNN and RNN. Then we describe a model-based tracking to simulate the process of model fitting by cascade RNN regression.

The first proposed model is a RNN that takes depth video frames as inputs and returns the estimated hand pose (See Fig. 1(a)). This model can be equivalently written in the tracking probability form as follows,

$$p(\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_T | \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T) = \prod_{t=1}^{T} p(\mathbf{z}_t | \mathbf{z}_{<t}, \mathbf{x}_{\leq t}) \tag{1}$$

where $\mathbf{z}_t$ and $\mathbf{x}_t$ are the estimated hand poses and the input depth-frame at time t, respectively. $\mathbf{z}_{<t}$ is a history of all previous locations before time t, and $\mathbf{x}_{\leq t}$ is a history of all input frames up to time t.

The second proposed framework consists of two main components: *pose cascade regression* and *tracking-detection gating mechanism* (see Fig. 1(b)). The pose regression module is a recurrent neural network that takes estimated hand pose as input and returns the improved hand pose estimation at each step/stage. The tracking-detection gating mechanism allows the model to choose between previously estimated pose and the pose prediction from CNN.

A. Tracking-by-detection approach

1) Modeling: We consider at each time step t of RNN, the input depth-frame $\mathbf{x}_t$ is first gone through a convolutional network to extract visual features. This gives us a feature vector $\phi(\mathbf{x}_t)$ of the input depth-frame $\mathbf{x}_t$ as follows,

$$\phi(\mathbf{x}_t) = \text{CNN}_{\theta_c}(m(\mathbf{x}_t)) \tag{2}$$

where CNN is a convolutional network with its parameters θ_c and $m(\cdot)$ denotes the preprocessing step including hand localization and segmentation and returning the local window containing only the current tracked human hand. The preprocessing steps are discussed in details in Section III-A2.

This feature vector is then fed into a RNN and the RNN updates its internal state $\mathbf{h}_t$ based on the previous hidden state $\mathbf{h}_{t-1}$, previous estimated hand pose $\tilde{\mathbf{z}}_{t-1}$ and the current frame feature vector $\phi(\mathbf{x}_t)$ as follows,

$$\mathbf{h}_t = \text{RNN}_{\theta_r}(\mathbf{h}_{t-1}, \tilde{\mathbf{z}}_{t-1}, \phi(\mathbf{x}_t)) \tag{3}$$

where RNN_{θ_r} can be any recurrent activation functions such as gated reccurent units (GRUs) [25], long short-term memory units(LTSMs) [26], or a simple logistic function with their parameters θ_r.

Using the updated hidden state $\mathbf{h}_t$, the RNN computes the predictive distribution over the hand pose's history as follows,

$$p(\mathbf{z}_t | \mathbf{z}_{<t}, \phi(\mathbf{x}_{\leq t})) = \text{out}_{\theta_o}(\mathbf{h}_t) \tag{4}$$

where θ_o is a set of parameters defining the output neural network. The whole process (Eqs. (2) - (4)) is iteratively applied as a stream of new frames comes. Fig. 1(a) graphically illustrate this process.

2) Pre-processing: To extract CNN visual features, we first detect the hand location in the input depth-frame. Then, a fixed-size metric cube is segmented from the depth image around the hand. The cube is resized to a 128×128 and the depth values within the cube are normalized to $[-1, 1]$. The depth values are clipped to the cube sides front and rear. This preprocessing step was also done in [6] to provide invariance to different hand-to-camera distances.

3) Training: The CNN that we constructed to extract visual features consists of three convolutional layers, three max-pooling layers and three fully-connected layers. This is a generic network similar to the one in [27].

Let C denote a convolutional layer, P a pooling layer and FC denote a fully connected layer. Both C and FC layers consist of Rectified Linear Unit (ReLU) [28] activation functions. For C layers, the size is defined as $(w \times h \times d)$, where the first two define the dimension of filters and the last one is the number of filters. For P layers, the size is defined as $(w \times h)$. The convolutional network can be described concisely as $C(5 \times 5 \times 8) \rightarrow P(4 \times 4) \rightarrow C(5 \times 5 \times 8) \rightarrow P(2 \times 2) \rightarrow C(3 \times 3 \times 8) \rightarrow P(1 \times 1) \rightarrow FC(1024) \rightarrow FC(1024) \rightarrow FC(30)$.

For the RNN, we consider a simple network with one hidden layer with 30 units, the output layer having the dimension of $J \times 3$, where J is the number of joints, and the input layer has the dimension of the last layer of CNN, i.e. 30, plus the dimension of the output layer.

We optimize the network parameters by using error back-propagation and apply the stochastic gradient descent (SGD) algorithm. The batch size is 128. The learning rate decays over the epochs at the rate of 0.95 and starts with 0.01. The networks are trained for 1000 epochs.

At each SGD update, we generate a minibatch of examples by the following steps:

- Choose a group of consecutive frames
- Extract their CNN features and ground truth pose
- Provide groundtruth pose and corresponding CNN features as the input sequences for RNN
- Receive groundtruth pose shifted (i.e. starting from frame $t+1$) as the output sequences for RNN

After these steps, each training example is a pair of data, containing a sequence of CNN features, and a sequence of ground-truth joint locations. We use this minibatch of N generated examples to compute the gradient of the minibatch log-likelihood.

B. Model-based tracking approach

1) Modeling: For each input depth-frame t, we also first compute its CNN's feature vector $\phi(\mathbf{x}_t)$. However, the role of RNN is now to evolute from the initial state of the hand pose to the correct one given the feature vector $\phi(\mathbf{x}_t)$ of the current depth-frame. Note that the input depth-frame also has gone through the pre-processing step as described in Section III-A2.

This feature vector is also fed into a RNN and the RNN updates its internal state $\mathbf{h}_t^{(k)}$ based on the previous hidden state $\mathbf{h}_t^{(k-1)}$ and previous estimated hand pose $\tilde{\mathbf{z}}_t^{(k-1)}$ as follows,

$$\mathbf{h}_t^{(k)} = \mathrm{RNN}_{\theta_r}(\mathbf{h}_t^{(k-1)}, \tilde{\mathbf{z}}_t^{(k-1)}) \tag{5}$$

where the first input in the sequence of RNN (with K stages) $\tilde{\mathbf{z}}_t^{(1)} = \tilde{\mathbf{z}}_{t-1}^{(K)}$ is the estimated pose of the previous frame $t-1$. The initial hidden state of RNN $\mathbf{h}_t^{(1)} = \phi(\mathbf{x}_t)$.

Using the updated hidden state $\mathbf{h}_t^{(k)}$, the RNN computes the predictive distribution over the hand pose's history as follows:

$$p(\mathbf{z}_t^{(k)} | \mathbf{z}_t^{(<k)}, \phi(\mathbf{x}_t^{(\leq k)})) = \mathrm{out}_{\theta_o}(\mathbf{h}_t^{(k)}) \tag{6}$$

where θ_o is a set of parameters defining the output neural network. This whole cascade fitting process is iteratively applied to improve the estimated pose at each time step for the current depth-frame. Fig. 1(b) graphically illustrates this process.

2) Hand pose subspace modeling: Besides using joints coordinates to represent hand pose, we also model hand pose via PCA. A hand pose subspace is learned from training data. In other words, we learn two functions *PoseToVec* and *VecToPose* to perform foward and inverse transformation between feature vector embedding space and joint coordinate space.

3) Training: The number of hidden units in each stage of the RNN is the same as the first framework, however, the dimensions of the inputs and outputs are both $J \times 3$.

We also use SGD but the way that we generate training samples are completely different. At each SGD update, we generate a minibatch of examples by the following steps:

- For each video frame, extract its CNN features
- The input of the first stage of RNN is the hand pose estimated from CNN
- The output at each stage of RNN is generated based on the distance between the first stage's input pose and the ground truth pose of the current frame so that the final stage will equal to the ground truth pose.

According to our experiments, we use this training strategy throughout this paper since it achieves our training target which is improving pose estimation stage-by-stage (See. Fig. 2 and Fig. 3).

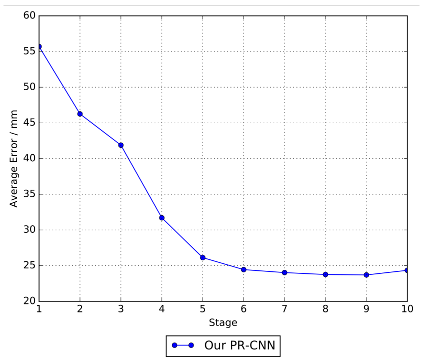


Fig. 2. Average joint errors over 10 recurrent stages

IV. Experiments

In this section, we evaluate the performance of our proposed framework on two different datasets including real and synthetic data. We also perform two different sets of experiments which include normal and occlusion cases to show the robustness of our proposed approaches.

A. Datasets

There exist few public datasets for hand pose estimation. Most early works perform quantitative evaluation only on synthetic data. Several recent hand tracking works release large enough training and testing data which allow a fair comparison. We evaluated our method on both real data and synthetic (occluded) data including the NYU Hand [11] and ICVL [6] datasets.

Real data: **ICVL** dataset [6] consists of 180K ground truth annotated training images captured from 10 different subjects of varying hand sizes. The dataset also contains two testing sequences (denoted A and B) each containing 1000 frames capturing various poses with severe scale and viewpoint changes. **NYU** dataset [11] contains 72K training and 8K test frames which are well annotated (i.e. containing $J = 36$ annotated joints). We follow the evaluation protocol of [17], [11] and use the same subset of $J = 14$ joints. The training set contains hand poses of one person, while the test set has hands from two persons. It also shows a high variability of poses, however, this can make pose estimation challenging. The dataset was captured using a structured light RGB-D sensor but we used only the depth images for our experiments.

B. Evaluation metrics and Methods for comparisons

Metric there are three quantitative metrics. The first two consists of the average and max per-joint (in millimeters) error across all frames. The third one is success rate, i.e. the percentage of correct frames whose average (or max) joint distance error is lower than a threshold (∼20mm). This strict measure has been described and used in [6].

Methods We demonstrate the efficacy of our frameworks by comparing them to the *Deep-model* based methods including **DeepNYU** [11] and **DeepPrior** [17]

C. Experiments on Real Dataset

First, we perform on the normal testset of the two real databases, i.e. ICVL and NYU. This experiment shows how good the model can handle **distinct poses** in **ICVL** testset and **unconstrained poses** using **NYU** testset. Second, we perform another experiment where we add random occlusions in the

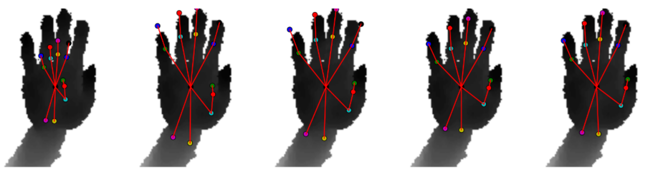
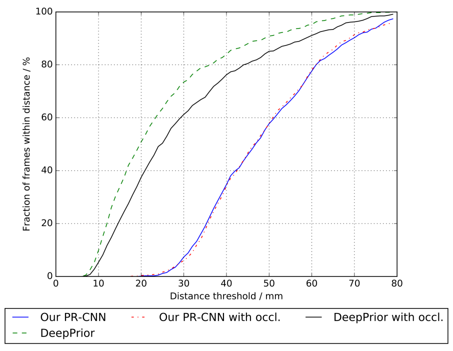
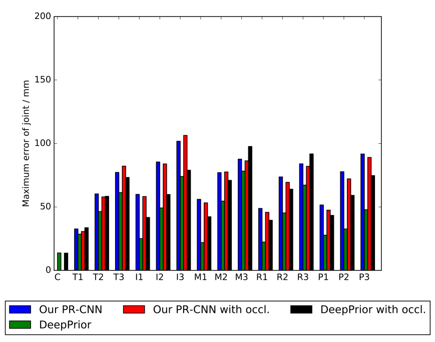


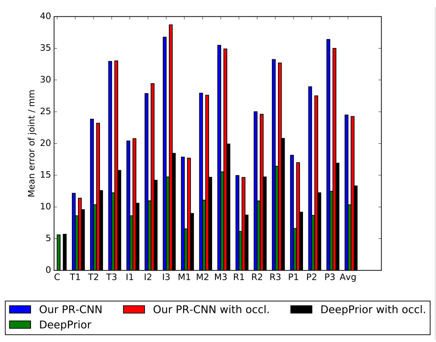
Fig. 3. Joint prediction over different stages (1, 2, 4, 8, 10)



(a) Success rate over different thresholds

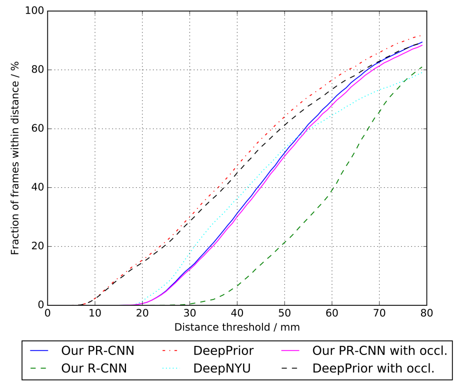


(b) per-joint max errors

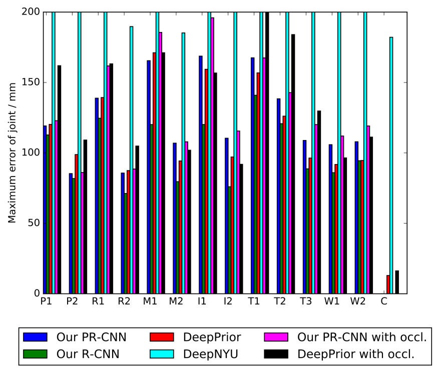


(c) per-joint average errors

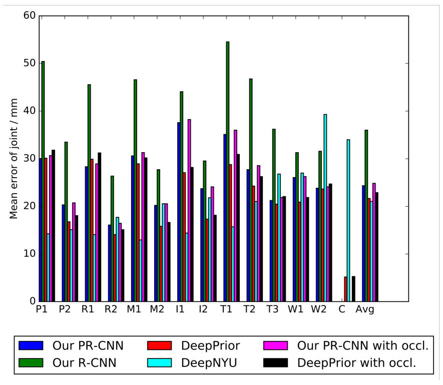
Fig. 4. Comparison of different methods on the ICVL dataset (Best viewed in color).



(a) Success rate over different thresholds



(b) per-joint max errors



(c) per-joint average errors

Fig. 5. Comparison of different methods on the NYU hand dataset (Best viewed in color).

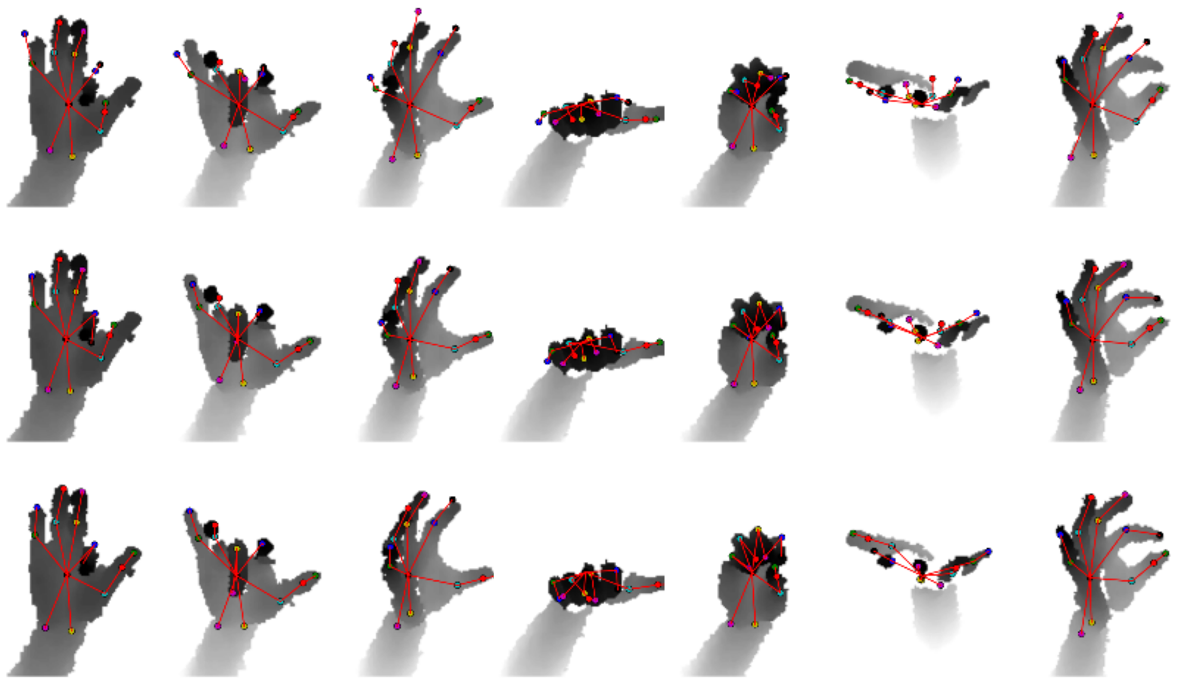


Fig. 6. Comparison of three methods (from top to bottom: our PR-CNN, Oberweger et al. and Tompson et al.) on the NYU dataset.

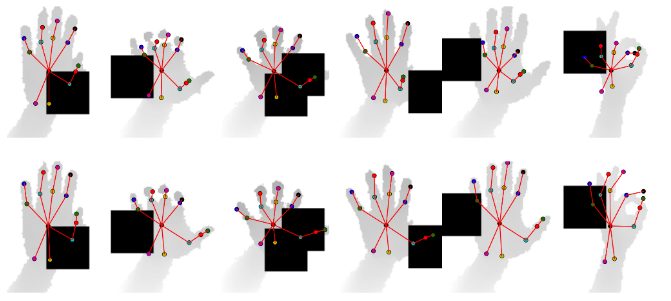


Fig. 7. Comparison of two methods (from top to bottom: Oberweger et al. and our PR-CNN) on the NYU dataset with occlusions.

depth input frames (input frames are chosen randomly). The results in terms of prediction errors for two experiments are shown in Fig. 4 for ICVL testset and Fig. 5 for NYU testset. Noting that the per-joint errors are also computed for each finger and palm which are indexed as C: palm, T: thumb, I: index, M: middle, R: ring, P: pinky, W: wrist. The qualitative results for the first and the second experiments are shown in Fig. 6 and Fig. 7, respectively. As we can see from the second experiment, the perfomrance of DeepPrior is affected significantly by occlusions while our PR-CNN method still work well in this case. Our methods achieve competitive results comparing to other state-of-the-art methods.

V. Conclusion

We proposed two different network architectures for tracking 3D hand poses. Both are using CNN and RNN. With the second architectures, i.e. model-based approach, we show that this framework can automatically correct the fail detection made in previous frames due to occlusions. We have compared the architectures on two datasets and shown that they are competitive with previous state-of-the-art both in terms of localization accuracy and robustness.

Acknowledgment

This work is supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada

References

[1] J. S. Supancic, III, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan, "Depth-based hand pose estimation: Data, methods, and challenges," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[2] Q. Gan, Q. Guo, Z. Zhang, and K. Cho, "First step toward model-free, anonymous object tracking with recurrent neural networks," *arXiv preprint arXiv:1511.06425*, 2015.

[3] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," *Computer Vision and Image Understanding*, vol. 108, no. 1, pp. 52–73, 2007.

[4] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun, "Hand pose estimation and hand shape classification using multi-layered randomized decision forests," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 852–863.

[5] C. Xu and L. Cheng, "Efficient hand pose estimation from a single depth image," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[6] D. Tang, H. Jin Chang, A. Tejani, and T.-K. Kim, "Latent regression forest: Structured estimation of 3d articulated hand posture," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[7] D. Tang, J. Taylor, P. Kohli, C. Keskin, T.-K. Kim, and J. Shotton, "Opening the black box: Hierarchical sampling optimization for estimating human hand pose," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[8] P. Li, H. Ling, X. Li, and C. Liao, "3d hand pose estimation using randomized decision forest with segmentation index points," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[9] C. Xu, A. Nanjappa, X. Zhang, and L. Cheng, "Estimate hand poses efficiently from single depth images," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 21–45, 2016.

[10] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[11] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 5, p. 169, 2014.

[12] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands Deep in Deep Learning for Hand Pose Estimation," in *Proc. Computer Vision Winter Workshop (CVWW)*, 2015.

[13] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1915–1929, 2013.

[14] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[15] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *BmVC*, vol. 1, no. 2, 2011, p. 3.

[16] S. Melax, L. Keselman, and S. Orsten, "Dynamics based 3d skeletal hand tracking," in *Proceedings of Graphics Interface 2013*. Canadian Information Processing Society, 2013, pp. 63–70.

[17] M. Oberweger, P. Wohlhart, and V. Lepetit, "Training a feedback loop for hand pose estimation," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[18] N. Kyriazis, I. Oikonomidis, P. Panteleris, D. Michel, A. Qammaz, A. Makris, K. Tzevanidis, P. Douvantzis, K. Roditakis, and A. Argyros, "A generative approach to tracking hands and their interaction with objects," in *Man–Machine Interactions 4*. Springer, 2016, pp. 19–28.

[19] K. Paliouras and A. A. Argyros, *Man–Machine Interactions 4: 4th International Conference on Man–Machine Interactions, ICMMI 2015 Kocierz Pass, Poland, October 6–9, 2015*. Cham: Springer International Publishing, 2016, ch. Towards the Automatic Definition of the Objective Function for Model-Based 3D Hand Tracking, pp. 353–363. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-23437-3_30

[20] I. Oikonomidis, M. Lourakis, and A. Argyros, "Evolutionary quasi-random search for hand articulations tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3422–3429.

[21] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[22] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt, "Fast and robust hand tracking using detection-guided optimization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[23] C. Choi, A. Sinha, J. Hee Choi, S. Jang, and K. Ramani, "A collaborative filtering approach to real-time hand pose estimation," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[25] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1653–1660.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.