Model-Based Deep Hand Pose Estimation

Xingyi Zhou¹, Qingfu Wan¹, Wei Zhang¹, Xiangyang Xue¹, Yichen Wei²
¹Shanghai Key Laboratory of Intelligent Information Processing
School of Computer Science, Fudan University
²Microsoft Research
¹{zhouxy13, qfwan13, weizh, xyxue}@fudan.edu.cn, ²yichenw@microsoft.com

Abstract

Previous learning based hand pose estimation methods do not fully exploit the prior information in hand model geometry. Instead, they usually rely a separate model fitting step to generate valid hand poses. Such a post processing is inconvenient and sub-optimal. In this work, we propose a model based deep learning approach that adopts a forward kinematics based layer to ensure the geometric validity of estimated poses. For the first time, we show that embedding such a non-linear generative process in deep learning is feasible for hand pose estimation. Our approach is verified on challenging public datasets and achieves state-of-the-art performance.

1 Introduction

Human hand pose estimation is important for various applications in human-computer interaction. It has been studied in computer vision for decades [Erol et al., 2007] and regained tremendous research interests recently due to the emergence of commodity depth cameras [Supancic III et al., 2015]. The problem is challenging due to the highly articulated structure, significant self-occlusion and viewpoint changes.

Existing methods can be categorized as two complementary paradigms, model based (generative) or learning based (discriminative). Model based methods synthesize the image observation from hand geometry, define an energy function to quantify the discrepancy between the synthesized and observed images, and optimize the function to obtain the hand pose [Oikonomidis et al., 2011; Qian et al., 2014; Makris et al., 2015; Tagliasacchi et al., 2015]. The obtained pose could be highly accurate, at the expense of dedicated optimization [Sharp et al., 2015].

Learning based methods learn a direct regression function that maps the image appearance to hand pose, using either random forests [Keskin et al., 2012; Tang et al., 2013; Xu and Cheng, 2013; Sun et al., 2015; Li et al., 2015] or deep convolutional neutral networks [Tompson et al., 2014; Oberweger et al., 2015a; 2015b]. Evaluating the regression function is usually much more efficient than model based optimization. The estimated pose is coarse and can serve as an initialization for model based optimization [Tompson et al., 2014; Poier et al., 2015; Sridhar et al., 2015].

Most learning based methods do not exploit hand geometry such as kinematics and physical constraints. They simply represent the hand pose as a number of independent joints. Thus, the estimated hand joints could be physically invalid, e.g., the joint rotation angles are out of valid range and the phalange length varies during tracking the same hand. Some works alleviate this problem via a post processing, e.g., using inverse kinematics to optimize a hand skeleton from the joints [Tompson et al., 2014; Dong et al., 2015]. Such post-processing is separated from training and is sub-optimal.

Recently, the deep-prior approach [Oberweger et al., 2015a] exploits PCA based hand pose prior in deep convolutional network. It inserts a linear layer in the network that projects the high dimensional hand joints into a low dimensional space. The layer is initialized with PCA and trained in the network in an end-to-end manner. The approach works better than its counterpart baseline without using such prior. Yet, the linear projection is only an approximation because the hand model kinematics is highly non-linear. It still suffers from invalid hand pose problem.

In this work, we propose a model based deep learning approach that fully exploits the hand model geometry. We develop a new layer that realizes the non-linear forward kinematics, that is, mapping from the joint angles to joint locations. The layer is efficient, differentiable, parameter-free (unlike PCA) and serves as an intermediate representation in the network. The network is trained end-to-end via standard back-propagation, in a similar manner as in [Oberweger et al., 2015a], using a loss function of joint locations.

Our contributions are as follows:

- For the first time, we show that the end-to-end learning using the non-linear forward kinematics layer in a deep neutral network is feasible. The prior knowledge in the generative model of hand geometry is fully exploited. The learning is simple, efficient and gets rid of the inconvenient and sub-optimal post processing as in previous methods. The estimated pose is geometrically valid and ready for use.
- Our approach is validated on challenging public datasets. It achieves state-of-the-art accuracy on both joint location and rotation angles. Specifically, we show
that using joint location loss and adding an additional regularization loss on the intermediate pose representation are important for accuracy and pose validity.

The framework of our approach is briefly illustrated in Figure 1. Our code is public available at https://github.com/tenstep/DeepModel

2 Related Work

A good review of earlier hand pose estimation work is in [Erol et al., 2007]. [Supancic III et al., 2015] provides an extensive analysis of recent depth based methods and datasets. Here we focus on the hybrid discriminative and generative approaches that are more related to our work. We also discuss other approaches that formulate handcraft operations into differentiable components.

Hybrid approaches on hand pose Many works use discriminative methods for initialization and generative methods for refinement. [Tompson et al., 2014] predicts joint locations with a convolutional neural network. The joints are converted to a hand skeleton using an Inverse Kinematics (IK) process. [Sridhar et al., 2015] uses a pixel classification random forest to provide a coarse prediction of joints. Thus a more detailed similarity function can be applied to the following model fitting step by directly comparing the generated joint locations to the predicted joint locations. Similarly, [Poier et al., 2015] firstly uses a random regression forest to estimate the joint distribution, and then builds a more reliable quality measurement scheme based on the consistency between generated joint locations and the predicted distribution. All these approaches separate the joint estimation and model fitting in two stages. Recently, [Oberweger et al., 2015b] trains a feedback loop for hand pose estimation using three neutral networks. It combines a generative network, a discriminative pose estimation network and a pose update network. The training is complex. Our method differs from above methods in that it uses a single network and seamlessly integrates the model generation process with a new layer. The training is simple and results are good.

Non-linear differentiable operations In principle, a network can adopt any differentiable functions and be optimized end-to-end using gradient-descent. [Loper and Black, 2014] proposed a differentiable render to generate RGB image given appearance, geometry and camera parameters. This generative process can be used in neutral network. [Chiu and Fritz, 2015] leverages the fact that associated feature computation is piecewise differentiable, therefore Histogram of Oriented Gradient (HOG) feature can be extracted in a differentiable way. [Kontschieder et al., 2015] reformulates the split function in decision trees as a Bernoulli routing probability. The decision trees are plugged at the end of a neural network and trained together. As we know, we are the first to adopt a generative hand model in deep learning.

3 Model Based Deep Hand Pose Estimation

3.1 Hand Model

Our hand model is from libhand [Šarić, 2011]. As illustrated in Figure 2, the hand pose parameters Θ ∈ RD have D = 26 degrees of freedom (DOF), defined on 23 joints. There are 3 DOF for global palm position, 3 DOF for global palm orientation. The remaining DOF are rotation angles on joints.

Without loss of generality, let the canonical pose in Figure 2 be a zero vector, the pose parameters are defined as relative to the canonical pose. Each rotation angle θi ∈ Θ has a range [θ˘i, θ^i], which are the lower/upper bounds for the angle. Such bounds avoid self-collision and physically infeasible poses. They can be set according to the anatomical studies [Albrecht et al., 2003], in our experiments we use a statistical bond from the training set, since our hand model is different from them.

We assume the bone lengths are known and fixed. Learning such parameters in a neutral network could be problematic as the results on the same hand could vary during tracking. Ideally, such parameters should be optimized once and fixed for each hand in a personal calibration process [Khamis et al., 2015]. In our experiment, the bone lengths are set according to the ground truth joint annotation in NYU training dataset [Tompson et al., 2014].

From Θ and bone lengths, let the forward kinematic function F : RD → RJ×3 map the pose parameters to J 3D joints (J = 23 in Figure 2). The kinematic function is defined on the hand skeleton tree in Figure 2. Each joint is associated with a local 3D transformation (rotation from its rotation angles and translation from its out-coming bone length). The
global coordinate of a joint is obtained by transforming the origin via a series of the local transformations along the path from the hand root joint to the joint under consideration. The implementation details are provided in Appendix.

The forward kinetic function $F$ is differentiable and can be used in a neural network for gradient-descent like optimization. Yet, it is highly non-linear and its behavior during optimization could be different from the other linear layers in the network. In this work, we show that it is feasible to use such a non-linear layer during deep neural network training.

3.2 Deep Learning with a Hand Model Layer

Taking an input depth image, our approach outputs the 3D hand joints and hand pose parameters $\Theta$. We use the same pre-processing as in previous work [Oberweger et al., 2015a; 2015b], assuming the hand is already detected (this can be done by a pixel-level classification random forest [Tompson et al., 2014] or assuming the hand is the closest object to the camera [Qian et al., 2014]). A fixed-size cube around the hand is extracted from the raw depth image. The spatial size is resized to $128 \times 128$ and the depth values are normalized to $[-1, 1]$.

Our network architecture is similar to the baseline network in deep prior approach [Oberweger et al., 2015a], mostly for the purpose of fair comparison. It is illustrated in Figure 1. It starts with 3 convolutional layers with kernel size 5, 5, 3, respectively, followed by max pooling with stride 4, 2, 1 (no padding), respectively. All the convolutional layers have 8 channels. The result convolutional feature maps are $12 \times 12 \times 8$. There are then two fully connected (fc) layers, each with 1024 neurons and followed by a dropout layer with dropout ratio 0.3. For all convolutional and fc layers, the activation function is ReLU.

After the second fc layer, the third fc layer outputs the 26 dimensional pose parameter $\Theta$. It is connected to a hand model layer that uses the forward kinematic function $F$ to output the 3D joint locations. A Euclidian distance loss for the joint location is at last. Unlike [Tompson et al., 2014; Oberweger et al., 2015a], we do not directly output the joint locations from the last fc layer, but use an intermediate hand model layer instead, which takes hand geometry into account and ensures the geometric validity of output.

The joint location loss is standard Euclidian loss.

$$L_{jt}(\Theta) = \frac{1}{2} ||F(\Theta) - Y||^2$$

(1)

where $Y \in \mathbb{R}^{J \times 3}$ is the ground truth joint location.

We also add a loss that enforces the physical constraint on the rotation angle range, as

$$L_{phy}(\Theta) = \sum_i |\max(\theta_i - \theta_i, 0) + \max(\theta_i - \bar{\theta}_i, 0)|.$$ (2)

Therefore, the overall loss with respect to the pose parameter $\Theta$ is

$$L(\Theta) = L_{jt}(\Theta) + \lambda L_{phy}(\Theta)$$ (3)

where weight $\lambda$ balances the two loss and is fixed to 1 in all our experiments.

In optimization, we use standard stochastic gradient descent, with batch size 512, learning rate 0.003 and momentum 0.9. The training is processed until convergence.

3.3 Discussions

In principle, any differentiable functions can be used in the network and optimized via gradient descent. Yet, for non-linear functions it is unclear how well the optimization can be done using previous practices, such as parameter setting. Our past experiences in network training are mostly obtained from using non-linearities like ReLU or Sigmoid. They are not readily applicable for other non-linear functions.

Our experiment shows that our proposed network is trained well. We conjecture a few reasons. Our hand model layer is parameter free and has no risk of over-fitting. The gradient magnitude of the non-linear 3D transformation (mostly $\sin$ and $\cos$) is well behaved and in stable range (from $-1$ to $1$). The hand model layer is at the end of the network and does not interfere with the previous layers too much. Our approach can be considered as transforming the last Euclidian loss layer into a more complex loss layer when combining the last two layers together.

The joint loss in (1) is well behaved as the errors spread over different parts. This is important for learning an articulated structure like hand. Intuitively, roles of different dimensions in pose parameter $\Theta$ are quite different. The image observation as well as joint locations are more sensitive to the global palm parameters (rotation and position) than to the finger parameters. This makes direct estimation of $\Theta$ hard to interpret and difficult to tune. In experiment, we show that using joint loss is better than directly estimating $\Theta$.

The physical constraint loss in (2) helps avoiding invalid poses, as verified in the experiment.

4 Experiment Evaluation

Our approach is implemented in Caffe [Jia et al., 2014]. The hand model layer is efficient enough and performed on the CPU. On a PC with an Intel Core i7 4770 3.40GHZ, 32GB of RAM, and an Nvidia GeForce 960 GPU, one forward pass takes about 8ms, resulting in 125 frames per second in test.
4.1 Evaluation of Our Approach

Our approach uses an intermediate model based layer. Learning is driven by joint location loss. To validate its effectiveness, it is compared to two baselines. The first one directly estimates the individual joints. It is equivalent to removing the model parameters and hand model layer in Figure 1. It is actually the baseline in deep prior approach [Oberweger et al., 2015a]. We refer this baseline as **direct joint**. The second one is similar to first one, except that the regression target is not joint location but the pose parameters (the global position and rotation angles in $\Theta$). We refer this baseline as **direct parameter**. Note that this baseline is trained using the ground truth pose parameters we obtained, as described earlier. Further, we refer our approach without using the physical constraint loss in Equation (2) as **ours w/o phy**.

As shown in Figure 3 and Table 1, our approach is the best in terms of all evaluation metrics, demonstrating that the hand model layer is important to achieve good performance for both joint and pose parameter estimation.

For the direct joint approach, we estimate the angle parameters using the similar PSO based method described above. That is, the pose parameters are optimized to fit the estimated joints, in a post-processing step. As the direct joint learning does not consider geometric constraints, one can expect that such fitting for the model parameters is poor. Indeed, the average difference between the optimized joint angles and ground truth joint angles is large, indicating that the estimated joints in many frames are geometrically invalid, although the joint location errors are relatively low (see Table 1).

We also experimented with adding the physical constraint loss but observed little difference.
Figure 4: Example results on NYU and ICVL datasets. The estimated 3D joints are overlaid on the depth image. Our method is robust to various viewpoints and self-occlusion.

Figure 5: Comparison of our approach and state-of-the-art methods on NYU test dataset. It shows the fraction of frames with maximum joint error below certain thresholds.

Direct parameter approach has decent accuracy on angles since that is the learning objective. However, it has largest joint location error, probably because small error in angle parameters does not necessarily imply small error in joint location. For example, a small error in global rotation could result in large error in finger tips, even when the finger rotation angles are accurate.

In ours w/o phy, we have best performance on both joints and rotation angles. Yet, when we consider the joint angle constraint, we find that in 18.6% of the frames, there is at least one estimated angle out of the valid range. When using physical constraint loss (ours), this number is reduced to 0.9%, and accuracy on both joints and rotation are similar (Table 1). These results indicate that 1) using a hand model layer with a joint loss is effective; 2) the physical constraint loss ensures the geometric validity of the pose estimation.

Example results of our approach are shown in Figure 4.

4.2 Comparison with the State-of-the-art

In this section, we compare our method with the state-of-the-art methods. For these methods, we use their published original result.

On the NYU dataset, our main competitors are [Tompson et al., 2014; Oberweger et al., 2015a]. Both are based on convolutional neural networks and are similar to our direct joint baseline. We also compare with [Oberweger et al., 2015b]. It trains a feedback loop that consists of three convolutional neural networks. It is more complex and is currently the best method on NYU dataset. Results in Figure 5 and Figure 6 show that our approach clearly outperforms [Tompson et al., 2014; Oberweger et al., 2015a] and is comparable with [Oberweger et al., 2015b].

On the ICVL dataset, we compare with [Tang et al., 2014] and [Oberweger et al., 2015a]. Results in Figure 7 show that our method significantly outperforms [Tang et al., 2014] and is comparable with [Oberweger et al., 2015a]. We note that the ICVL dataset has quite inaccurate joint annotation and small viewpoint changes (as discussed in [Supancic III et al., 2015]). Both are disadvantageous for our model based approach because it is more difficult to fit a model to inaccurate joints and the strong geometric constraints enforced by the model are less effective in near-frontal viewpoints. We also
Figure 7: Comparison of our approach and state-of-the-art methods on ICVL test dataset. It shows the fraction of frames with maximum joint error below certain thresholds and the average error on individual joints.

note that we use the same geometric hand model as for NYU dataset and only learn the rotation angles. Considering such limitations, our result on ICVL is quite competitive.

5 Conclusions

We show that it is possible to integrate the forward kinematic process of an articulated hand model into the deep learning framework for effective hand pose estimation. Such an end-to-end training is clean, efficient and gets rid of the inconvenient post-processing used in previous approach. Extensive experiment results verify the state-of-the-art performance of proposed approach.

Essentially, our approach exploits the prior knowledge in geometric hand model in the learning process. It can be easily applied to any articulated pose estimation problem such as human body. More broadly speaking, any deterministic and differentiable generative model can be used in a similar manner [Loper and Black, 2014; Oberweger et al., 2015b]. We hope this work can inspire more works on effective integration of generative and discriminative methods.

Figure 8: Illustration of forward kinematic implementation. Joint A, B, C, D are 4 adjacent joints of the initial hand model (not necessarily collinear). The relative 3D coordinate of joint $D''$ with respect to A after two rotations centered at joint B and C among axis Z can be written as $p_{D''} = \text{Trans}_x(l_1) \times \text{Rot}_z(\theta_1) \times \text{Trans}_x(l_2) \times \text{Rot}_z(\theta_2) \times \text{Trans}_x(l_3) \times [0, 0, 0, 1]^T$.

Acknowledgments

This work was supported in part by NSFC under Grant 61473091 & 61572138, two STCSMs programs (No. 15511104402 & No. 15JC1400103), and EU FP7 QUICK Project under Grant Agreement (No. PIRSES-GA-2013-612652).

Appendix on hand model kinematics

The hand model layer takes the model parameters as input and outputs the corresponding joint coordinates. As the hand model is a tree structure kinematic chain, the transformation of each joint is a forward-kinematic process. We can consider the transformation of two adjacent joints as transform two local coordinate systems. Let the original coordinate of a point be $(0, 0, 0)$, represented in homogenous coordinate $[0, 0, 0, 1]^T$. $\text{Trans}_\phi(l)$ is the 4x4 transformation matrix that transforms $l$ among axis $\phi \in \{X, Y, Z\}$, and $\text{Rot}_\phi(\theta)$ is the 4x4 rotation matrix that rotate $\theta$ degrees among axis $\phi$. Generally, let $Pa(u)$ be the set of parent joints of joint $u$ on the kinematic tree (rooted at hand center), the coordinate of $u$ after $k$ relative rotation is:

$$p_{u^{(k)}} = \left( \prod_{t \in Pa(u)} \text{Rot}_{\phi_t}(\theta_t) \times \text{Trans}_{\phi_t}(\theta_t) \right) [0, 0, 0, 1]^T \quad (4)$$

Note that most of the joints have more than one rotation DOF, but the formulation is the same as equation (4), as the additional rotation matrices are multiplied on the left of the corresponding joints. The derivation of joint coordinate $u$ with respect to joint angle $t$ is replace the rotation matrix of joint angle $t$ (if exists) by it’s derivation and keep other matrix unchanged.

References


