How to Refine 3D Hand Pose Estimation from Unlabelled Depth Data ?

Endri Dibra†, Thomas Wolf†, Cengiz Öztireli, Markus Gross
Department of Computer Science, ETH Zürich
†Equal contribution from both authors
{edibra,cengizo,grossm}@inf.ethz.ch, wolftho@student.ethz.ch

Abstract

Data-driven approaches for hand pose estimation from depth images usually require a substantial amount of labelled training data which is quite hard to obtain. In this work, we show how a simple convolutional neural network, pre-trained only on synthetic depth images generated from a single 3D hand model, can be trained to adapt to unlabelled depth images from a real user's hand. We validate our method on two existing and a new dataset that we capture, both quantitatively and qualitatively, demonstrating that we strongly compare to state-of-the-art methods. Additionally, this method can be seen as an extension to existing methods trained on limited datasets, which helps on boosting their performance on new ones.

1. Introduction

Recent approaches in 3D hand pose estimation from a single depth image are predominantly based on convolutional neural network architectures [29, 15, 32, 6, 22, 30], typically requiring labelled data for training. While the accuracy of such methods has been disputed on a limited number of available datasets that are applicable to learning-based approaches, such as [29, 2, 23], the main problem seems to shift to a large degree towards scarcity of data labelling (e.g. 3D joint positions). This has been particularly demonstrated in [31], where simply having a bigger and more complete labelled dataset yields much better estimation results, but also in [8], where it is shown that just using nearest-neighbor search methods in the pose data space can already outperform many of the existing, CNN-based methods. Multiple ways of creating labelled data have been presented in the past, usually on the expense of additional set-up environments and man-work. For instance, labels have been generated via optimization [29], utilizing multiple cameras, integrating special sensors [31] or in a semi-supervised way [14, 25]. This is also reflected by the limited amount of public datasets available.

Next to convolutional data-driven approaches, there have been several generative, model-driven ones that perform iterative optimization. For instance, [13, 20, 24, 26, 27] optimize for point cloud correspondences while [17, 21, 29, 18] attempt to find a good pose, by iteratively rendering many synthetic depth images and comparing them to the input image. Such approaches usually perform better on unseen poses, as compared to data-driven ones, when applied to poses quite dissimilar from the ones in training datasets. A further advantage is independence from a big labeled training set. However, such methods usually require temporal information and a good initialization (typically classifying them as tracking methods), and are computationally more expensive.

Inspired by such optimization approaches and haunted by the problem of creating labelled data, we propose a novel method that bypasses the expensive effort of labelling ground truth data, while still leveraging from the speed of purely data-driven approaches, to achieve accurate 3D hand pose predictions.

To this end, we propose a pipeline, with the help of which pre-trained convolutional models (here on a purely synthetic dataset) can be refined to unseen and unlabelled depth images. This allows to boost existing data-driven methods, which are mainly optimized for the small amount of available training datasets, to real-world scenarios where labelled data is hard to obtain.

To summarize, the paper contributions are:

- We demonstrate how to train a CNN-based 3D hand pose estimation method, on unseen and unlabelled depth images, avoiding the need for annotated data.
- We propose a new training pipeline that can accurately estimate 3D hand pose with the ability to refine itself on unlabelled depth images, using a depth loss component with a physical and collision regularizer.
- We demonstrate, through extensive evaluations, the advantage of utilizing such a method to enhance estimations of a simple candidate CNN model.

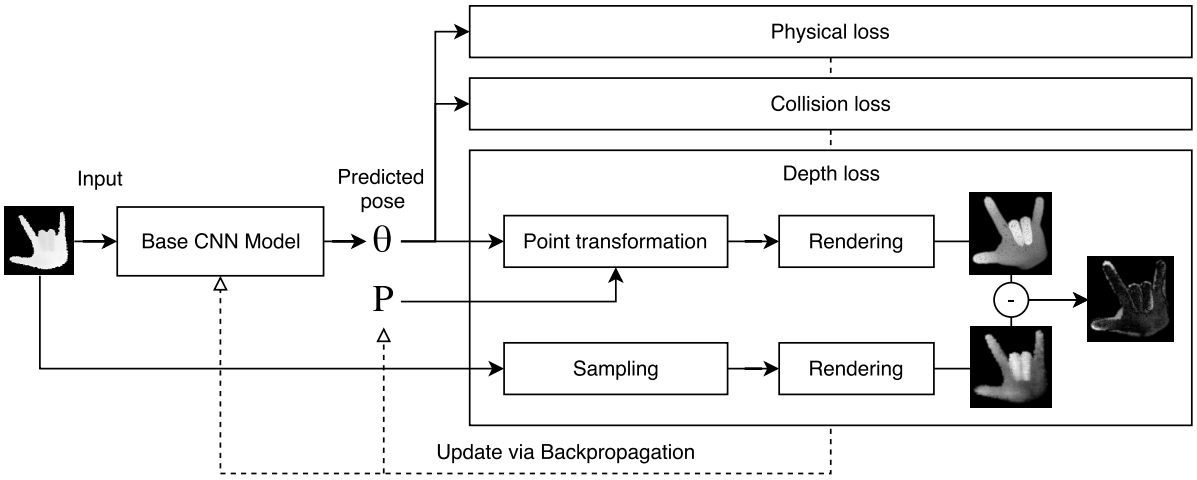

Figure 1. Overview of the training pipeline. Given a depth image as input, a base CNN model predicts the hand pose θ. Given θ, we calculate a loss consisting of a collision, physical and depth component. During training, we update the weights of the base model, as well as P, a point cloud that represents the hand shape and gets iteratively updated to the real one. Since we can calculate the loss using only the input image, θ and P, our model can be trained without labelled data.

2. Related work

Generative, Optimization-Based Approaches. Many methods in this category utilize gradient-based optimization approaches and attempt to solve an iterative closest point (ICP) problem. In this context, Melax *et al*. [13] formulate the hand optimization as a constrained rigid body problem. Schroder *et al*. [20] suggest optimizing in a reduced parameter space and Tagliasacchi *et al*. [24] combine previous results, to show that ICP in combination with temporal, collision, kinematic and data-driven terms can be utilized to track with high robustness and accuracy. Following up on this, Sharp *et al*. [21] enhance this approach utilizing a smooth model and Tkach *et al*. [27] present a new hand model based on sphere meshes.

A non-gradient, particle swarm optimization (PSO) approach has been suggested by Oikonomidis *et al*. [17], minimizing "the discrepancy between the appearance and 3D structure of hypothesized instances of a hand model and actual hand observations". This requires extensive rendering of an explicit hand model in various poses. Tompson *et al*. [29] use an (offline) PSO based approach to find the ground truth for the NYU dataset [29]. Since PSO depends highly on a good initialization, Gian *et al*. [18] increase its robustness by combining it with ICP, while Taylor *et al*. [26] suggest minimizing a truncated L1 error norm between the synthesized and real depth image while also rendering a more realistic-looking mesh through linear blend skinning (LBS) [11]. In general, these techniques focus on tracking, requiring a good initialization or GPU implementations, while our method focuses on single depth image 3D pose estimation and can run real-time on CPU.

Discriminative, Data-Driven Approaches. Recently, many methods based on convolutional neural networks (CNNs) have been proposed. Oberweger *et al*. [15] evaluate different CNN architectures and propose a pose prior by adding a bottleneck layer, showing that a projection to a reduced subspace before the final regression boosts the prediction performance. Zhou *et al*. [32] propose a forward kinematic layer to create a loss function on the joint positions while predicting rotation angles of the joints. Using those angles, a physical loss is introduced, which penalizes angles outside a specified range, similar to what we do. Instead of directly using depth images as input, Ge *et al*. [6] show that projecting the point cloud onto three orthogonal planes and feeding the projections into three different CNNs enhances the prediction performance. Deng *et al*. [3] convert the depth map to a 3D volumetric representation first, and then feed it into a 3D CNN to produce the pose in 3D, requiring no further processing.

Hybrid Methods. Often, neural networks are used as an intermediate prediction step which requires optimization afterwards. Tompson *et al*. [29] predict various key positions and optimize for the actual pose using inverse kinematics. Ye *et al*. [30] combine a spatial attention mechanism and PSO in a cascaded and hierarchical way. Sinha *et al*. [22] utilize a CNN to reduce the dimensionality of the depth input and optimize for the final pose via a matrix completion approach considering also temporal information. Oberweger *et al*. [16] use a deep generative neural network to synthesize depth images, and a separate optimization network to iteratively correct the pose predicted by a third convolutional model.

Similary in spirit to the PSO approaches, starting from a rigged 3D hand model, we synthesize depth images in order to compare to the input depth images. We, however, do not do this externally, but rather integrate it in a conventional gradient based learning architecture, similar to [12, 1].
Our base CNN architecture initially predicts joint rotations from a base reference pose, similar to [32]. This allows us to completely reconstruct the articulated hand pose, whereas predicting just joint positions needs a further optimization step to do the same. Our method builds on top of [32], since we also calculate a forward kinematic chain from the predicted pose and we also utilize a physical loss (Sec. 3.5). However, we go a step further and do not minimize a loss on the joint positions but on the actual hand depth image, enabling us to adapt to unlabelled images.
At first glance our method might appear to be similar to the feedback loop proposed by Oberweger *et al*. [16], but there are some important differences we want to emphasize to avoid confusion. Oberweger *et al*. [16] synthesize depth images too, however such images are utilized to iteratively optimize a pose prediction during testing, whereas we optimize our base model during training only, by backpropagating errors on depth images. Our prediction employs only a single forward pass through the CNN. Furthermore, our method is completely independent of labelled real depth images, whereas [16] highly depends on well labelled data to adapt to a dataset (*e.g*. training on ICVL fails because of annotation errors). Our method allows for simple end-to-end training, but the method from [16] requires to train three different neural networks. We will elaborate more on the differences between the state-of-the-art methods in the results section (Sec. 4.5).
All in all, our method could be seen as a network extension to data-driven methods, in order to boost predictions by training at a minimal cost (from unlabelled depth data). Our goal thus, slightly differs from that of most of the above-mentioned works, which mainly focus on maximising pose estimation accuracy on available datasets.

3. Method

3.1. Overview

The overview of our method is depicted in Fig. 1. The main goal is to estimate the 3D hand pose, given a single depth image, utilizing an end-to-end CNN. Specifically we attempt to tackle cases, where the depth input data space does not necessarily represent the training space, due to e.g. variation in hand shape, pose space and sensor noise. We propose to achieve this by refining a base convolutional neural network on unlabelled depth images. The base model is an AlexNet-like [10] architecture (Sec.4) and pretrained purely on synthetic data to provide a (rough) pose estimate θ. To train on unlabelled data, we propose a combined loss function, containing a depth (Sec.3.3), collision (Sec.3.4) and a physical component (Sec.3.5), as shown in Eq.1 :

$$\mathcal{L} = \mathcal{L}^{depth}(\theta, P) + \mathcal{L}^{\text{coll}}(\theta) + \mathcal{L}^{\text{phys}}(\theta) \qquad (1)$$

where P is a point cloud representation and estimate of the hand shape in a neutral pose, that gets iteratively adapted to the real one during training.
The depth loss adapts the base model to reduce the L1 error norm between a synthesized depth image, generated through applying the prediction θ to an updated pointcloud P of a rigged hand model, and a second synthesized depth image based only on the input depth image. The collision and physical loss can be thought as regularizers that penalize unnatural looking poses.

3.2. Base CNN Model

We start by training a CNN model which can predict a pose θ from a depth image D. To represent θ, we adopt quaternions, however euler angles, rotation matrices or similar structures could possibly be utilized too. Without loss of generality, we chose our base network to be based on the AlexNet architecture [10], similar to Zhou *et al*. [32].
In order to initially train the network, we generate a lot of synthetic training data (Sec.4), consisting of pairs of depth images and poses in our format of θ. The data is generated from a rigged 3D hand model, with 16 control joints, as depicted in Fig.2 (1). Given the depth images of the synthetic training data, we train the base model to minimize the mean squared error between the pose from our dataset and the predicted pose.
Our trained CNN based model could be replaced by any other model that can predict a pose given a depth image. The only constraint in this case, is that θ must be informative enough to calculate a forward kinematic chain, yielding the exact information on how each joint transforms to the predicted pose (see Sec. 3.3.1).
The base network is only supposed to give a rough initial prediction. The following components enable it to get refined by training with unlabelled data.

3.3. Depth Component

In order to assess the prediction accuracy on unlabelled data, we opt at comparing the input depth image D to a synthesized depth image from our predicted pose θ and a pointcloud P sampled from the hand model. Hence rendering and synthesis of depth images given θ becomes a necessity, which we achieve by imitating the calculations of a common render application and utilizing linear blend skinning (LBS) [11] to transform the points according to θ.

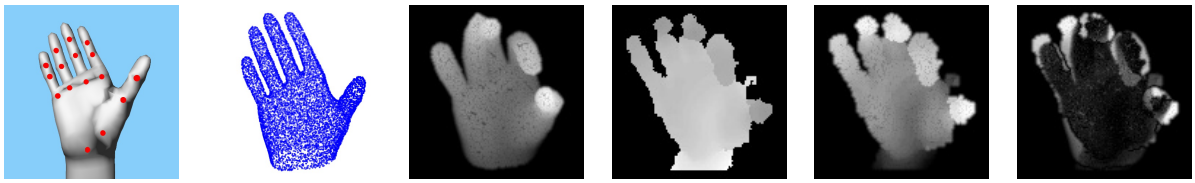

Figure 2. Overview of the different kinds of formats and hand images we process. From left to right we show (1) our rigged hand model with its 16 joints (2) the uniformly sampled point cloud P of the rigged mesh (3) a rendering after transforming P using the render component (Sec.3.3.1) (4) a typical noisy depth image input to the base CNN model. (5) a rendering of points sampled from (4). (6) the absolute difference between (3) and (5)

3.3.1 Point Transformation

Forward Kinematics. The point transformation behaves similar to a render application. This includes the forward kinematic chain which yields for each joint the transformation matrix, transforming from the model space of a base pose into the model space of the skinned pose θ. This is similar to [32] which we refer to for details. Please note that this step is differentiable, since only matrix multiplications and trigonometric functions are required. We denote with $M = [M_1, \dots, M_m]$ those transformation matrices, where m is the number of joints used (see supplementary for details).

Linear Blend Skinning. In contrast to [32] though, we are not just transforming each joint position to its final position, but a larger set of points $P = [p_1, \dots, p_n]$, where each point is associated with one or more joints. Let $w_{i,j}$ be the weight which defines how much the point p_i is bound to the joint j. Linear blend skinning (LBS) [11] $f^{\text{skin}}(P, M)$ transforms each point by a linear combination of the matrices M_j according to its weights:

$$\hat{p}_i := f_i^{\text{skin}}(P, M) = \sum_{j=1}^{m} w_{i,j} M_j p_i \tag{2}$$

We note that this method is not just differentiable with respect to M, which is important to allow backpropagation to the base model, but also with respect to P. This allows us to relax the static hand model to a dynamic one, that gets updated during training to automatically adapt to the hand shape. Hence, the 3D hand model can be adapted to the real hand shape by iteratively minimizing their projected difference (Sec.3.3.2).

3.3.2 Rendering

Instead of using triangles as primitives, that involve a difficult rasterization step and produce an image which is not differentiable, we make use of a method based on [1] to render a point cloud in a differentiable way.

Instead of transforming the vertices of the model, we actually transform a point cloud P. P and its weights W are once uniformly sampled from the hand model, which acts therefore only as a hand shape prior.

To render the point cloud, we take the point with the minimal z-value for each of the image coordinates (i, j). Instead of picking individual widely spaced points, in order to make it differentiable and obtain a nice surface, we weight the z-value of each point with a 2D basis function ϕ around its position. Let $p_i = [p_{i,x}, p_{i,y}, p_{i,z}]$. The rendered depth image approximation is defined as:

$$f_{i,j}^{\text{depth}}(P) = \max_k(\text{depth}_{i,j}(p_k)) \tag{3}$$

where we assume the points to be in the $[0, 1]$ range and the z-values to represent the depth with respect to the camera:

$$\text{depth}_{i,j}(p) = (1 - p_z)\phi_{i,j}(p) \tag{4}$$

We choose $\phi \in C^1$ to have finite spatial support of a circle with radius r. Let $\text{dist}_{i,j}^2(p) = (j - p_x)^2 + (i - p_y)^2$. We can define ϕ as:

$$\phi_{i,j}(p) = \left(1 - \left(\frac{\text{dist}_{i,j}(p)}{r}\right)^2\right)^2 \mathbb{1}_{\text{dist}_{i,j}^2(p) < r^2} \tag{5}$$

Even though the rendered images look like depth images, there is still a visible disparity between the synthesized and real images, as it can be seen in between Fig.2 (3) and (4). Therefore we also sample a point cloud P_D from the real depth image D and render it using f^{depth}, as in Fig.2 (5). The actual loss taken in the end is the L1 norm of the difference between both synthesized images, Fig.2 (3) and (5):

$$\mathcal{L}^{\text{depth}} = \sum_{i,j} |f_{i,j}^{\text{depth}}(f^{\text{skin}}(P, M)) - f_{i,j}^{\text{depth}}(P_D)| \tag{6}$$

3.4. Collision Component

Inspired by [24] and [13], we also attempt to avoid finger interpenetration by penalizing over a self-collision approximation on the hand mesh. We approximate the hand with cylinders and check for each cylinder if a joint position is inside. Let $B = [b_1, \dots b_m]$ denote the joint positions calculated using the joint transformation matrices M (see 3.3.1).

Let $C \subset \mathbb{N} \times \mathbb{N}$ be all joint indices paired with their parent. Hence, each pair $(i, j) \in C$ describes a bone. We define the loss as:

$$\mathcal{L}^{\text{coll}} = \frac{1}{m} \sum_{(i,j) \in C} \sum_{k=1}^{m} \mathbb{1}_{i \neq k, j \neq k} f^{\text{coll}}(b_i, b_j, b_k) \quad (7)$$

where $f^{\text{coll}}(a, b, p)$ is the penetration depth of a point p into a cylinder with the endpoints a and b and a fixed radius. The radius could be determined using the point cloud P of our hand shape but we obtained reasonable results by choosing a fixed value.

3.5. Physical Component

The physical loss is defined similarly to [32] and [24]. We first transform our pose θ (which we represent in quaternions) to euler angles using the function $\varphi(\theta)$. In euler angles we can specify a valid range $[\underline{\varphi}, \bar{\varphi}]$ for each angle. The bounds are determined manually by looking at the model while varying the pose (please see supplementary). The loss penalizes poses outside the specified range:

$$\mathcal{L}^{\text{phys}}(\theta) = \max(\underline{\varphi} - \varphi(\theta), 0) + \max(\varphi(\theta) - \bar{\varphi}, 0) \quad (8)$$

4. Experiments and Results

4.1. Architecture and Training Details

We utilize a slightly modified AlexNet [10] CNN architecture as our candidate base model. Similar to [4, 5], we adopt it for regression, however we experienced that removing one of the two fully connected layers achieves a faster learning with similar performance. Hence, we only have one fully connected layer with 4096 neurons using a ReLU activation function. Before linearly regressing to θ, we add a dropout layer with (keep) probability 0.75. For details on the architecture please check the supplementary material.

As input we expect a batch of 120×120 pixels depth images, with the depth values scaled in a $[0, 1]$ range. The hand is cropped and centered, by padding on the sides when necessary such that the aspect ratio is preserved. We choose to regress to quaternions and therefore $\theta \in \mathbb{R}^{16 \times 4}$, since there are 16 joints.

The network is pre-trained on synthetic depth images with randomly generated poses. The poses are sampled from a feasible angle range and collisions are avoided utilizing a similar approximation as in Sec. 3.4.

We train the base model on synthetic data until convergence, with the Adam Optimizer [9] and a learning rate of 10^{-3}. The complete network is trained on unlabelled data with a learning rate of 10^{-5}, for 10 epochs for each training set. We choose a batch size of 200 and 1000 for training on the labelled synthetic and unlabelled real data respectively.

The complete pipeline is implemented in tensorflow. Linear blend skinning, the rendering and collision function are implemented in separate custom operations utilizing cuda kernels to provide high efficiency for offline training (please see supplementary for more details).

4.2. Datasets

We evaluate our method on two public datasets (NYU and ICVL) and a separate one created by us. The NYU [29] dataset is recorded using a Microsoft Kinect sensor and provides therefore, in comparison to the other dataset, very noisy images. The training set has 72757 images from one person and three simultaneous views. The test set is a sequence of 8252 images from two persons. The ground truth annotations are fitted with an offline PSO approach and consist of 36 3D spatial hand features per frame. Similar to previous works [32], we also use only a subset of 16 3D positions for evaluation.

The ICVL dataset [2] is less noisy due to the usage of the Intel Creative Interactive Gesture Camera. Two test sequences from two different persons with a total of 1596 frames are provided for testing and about 180K frames for training from several persons. The ground truth of 16 3D bone centre locations is obtained utilizing the tracking method proposed by Melax *et al.* [13]. We segment the images of NYU and ICVL by cutting out a padded block around the 3D annotations. Furthermore, it is important to mention that our 3D joint positions, that are dependent on our fixed 3D model skeleton, deviate a lot from the ones used in both datasets, which is important for a fair comparison.

Our own dataset is created using the Intel RealSense Camera and consists of 2000 depth images for testing and 50000 depth images for training, from only one person wearing a black wristband. This allows for a simple brightness based segmentation to cut out the wrist, which makes it easy to separate the foreground from the background. We also capture a color image for each frame for qualitative comparison. Note that we do not provide any annotations. We compare to different methods by computing ROC curves, that denote the fraction of frames below a maximum 3D join prediction error.

4.3. Feasibility of Learning Hand Pose and Shape

It has been shown in [12], how to estimate the human body shape from depth (and color) image differences using a gradient based method, but except for the concurrent optimization based approaches on differentiable offline [19] and online [28] calibration for hands, to the best of our knowledge there exist no CNN based works in the field of hand pose estimation. In methods utilizing PSO [17, 29, 18, 21], we have seen that an error metric on depth images is meaningful enough to intelligently sample, compare and prune candidate poses, however the gradient idea has not been exploited. Encouraged by such results, we show in two experiments that we can optimize for the hand pose and shape.

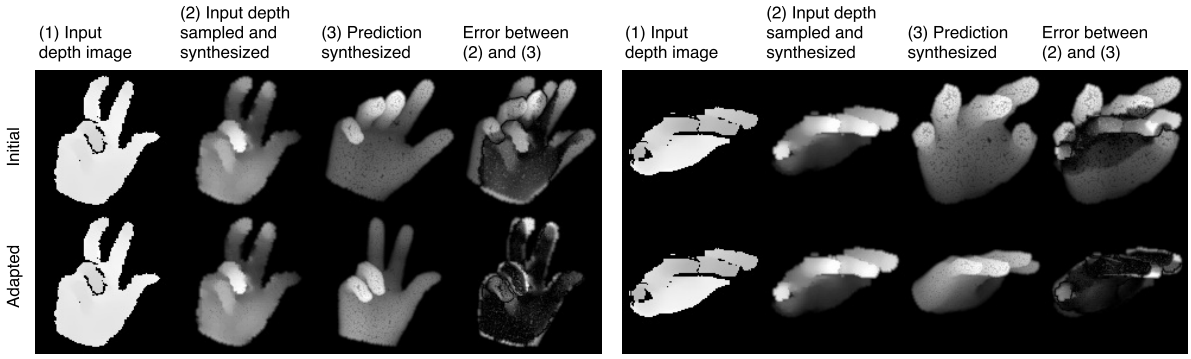

Figure 3. Firstly, we overfit our model on single depth images to give evidence that learning the pose without any annotations is possible.

Hand Pose Optimization. In our first experiment we attempt to overfit our model on single images, in order to show that reasonable results are obtainable despite our loss function being highly non-convex. We take images from our training set and train the network for 70 epochs (update steps). In Fig. 3, for each block we show the initial prediction of our base model (on top) and the prediction after training (at the bottom). We typically obtain good results, where the error between the depth images is minimized, except for cases when the initial prediction is too far from the actual pose, impeding convergence to a desired minimum.

Hand Shape Optimization. We also explore whether the point cloud P is adequately adapted to the hand shape. For that, we perform the same experiment as before, however we keep the weights of the model fixed, such that only P gets updated. As shown in Fig. 4, all our test runs converge slowly to an optimum which almost completely vanishes the loss over the synthesized depth images.

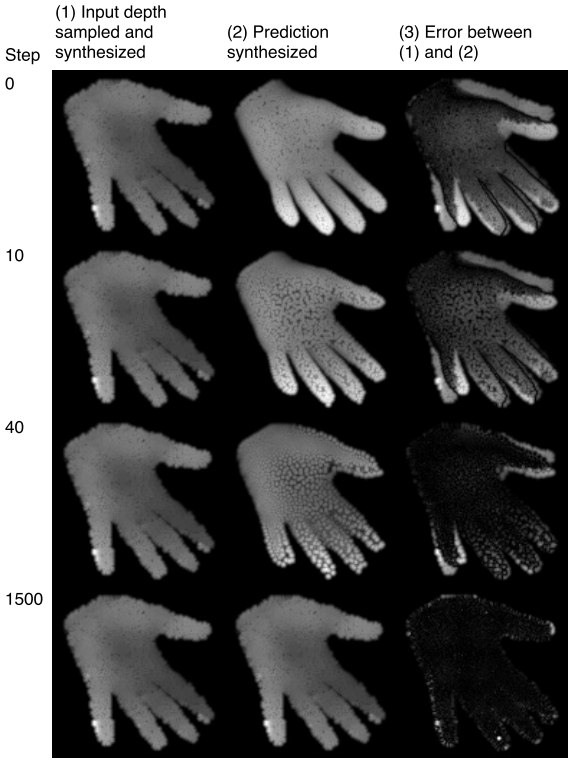

Figure 4. In a second experiment we optimize for the hand point cloud P only and demonstrate how the updates converge close to a global optimum. See video for all the in-between steps.

4.4. Self Comparison

Given the previous results on single images, it is important to demonstrate that our model can also adapt to complete datasets. Our attempt is not to overfit to any dataset, but generalize to similar inputs by adapting to the sensor noise and hand shape in the training set. To show this, we start with a **quantitative** self comparison on the NYU [29] and ICVL [2] datasets. We train our base model twice, unsupervised for each of the training sets, where first we use the depth component only and then all losses altogether.
The results on the validation set are shown in Fig. 5. We can see a significant improvement in both datasets by training with the depth component only. Incorporating the physical and collision component gives only a very small improvement on ICVL, but a second big improvement on NYU. This illustrates that when learning from noisy data, we are more dependent on prior information, *e.g.* by enforcing non-self-intersection and physical constraints. The images of ICVL, however, are mostly of higher quality, allowing to infer this information already from the depth image.
In Fig. 6, we show a **qualitative** self comparison on two random pose predictions from our dataset, before and after training. A more accurate 3D pose of the real hand is observed in the latter case. To give quantitative evidence, we train our model on our own training data for 10 epochs and show in Fig. 7, that the model generalizes well to the validation set. We also notice that the variance drops from 0.0035 to 0.0026, indicating a more stable estimation as

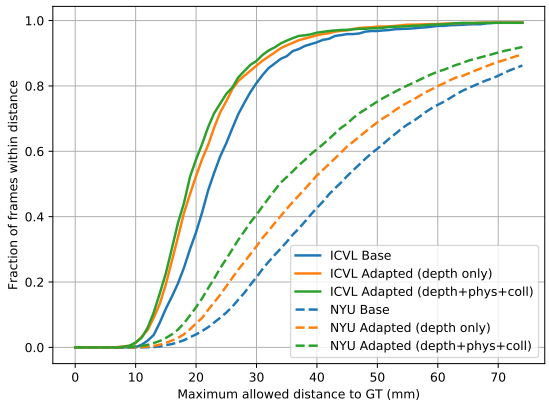

Figure 5. Self comparison on the NYU [29] and ICVL [2] dataset.

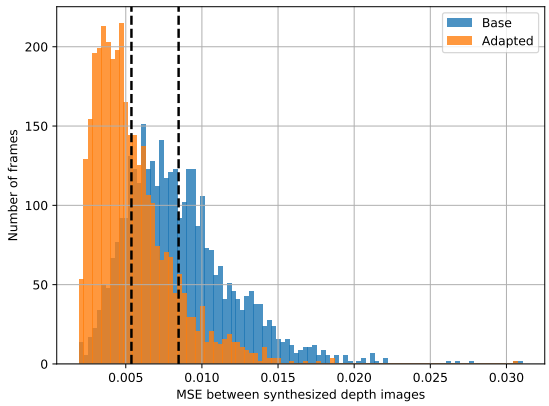

Figure 7. Distribution of the MSE between the synthesized depth images (see (3) and (5) in Fig 2) over the frames of our validation set, before and after training on our training set. The vertical lines show the mean values before (right) and after prediction (left).

also backed up by visual inspection, where the jitter in video sequences gets reduced (please see supplementary video).

4.5. Comparison to State-of-the-art

Despite the fact that our target is to refine an initial CNN based model and adapt it to unlabelled depth data, we also compare to state-of-the-art methods that attempt to estimate the 3D pose from a single depth image on standardized datasets.

To start, we give some more context about the methods we compare to and claim that a direct comparison is quite difficult. We compare to REN [7], DeepPrior [15] and DeepModel [32] on the NYU and ICVL dataset. Additionally we compare to Feedback [16] and LRF [2] on the NYU and ICVL dataset respectively.

All these methods utilize the ground truth annotations of the training data to refine their models, whereas we deliberately do not make use of them. Furthermore, except for [15], the rest of the methods attempt to minimize the difference between the given annotations and their predictions. The feedback loop, proposed by Oberweger *et al*. [15], minimizes a loss based on the depth images, as we do. Instead of a point transformation and rendering architecture though, they synthesize depth images (given a pose) via a generative CNN. This has the advantage of not needing an explicit model, as we do, but on the other hand they show a high dependency on good annotations, whereas we are completely independent of them.

Since we do not optimize for joint positions and use a different model than the ones that have been used for creating the annotations in the NYU and ICVL dataset, we observed an error between our joint position prediction and the ground truth, even for accurate predictions, as evaluated by the depth image differences. Therefore, a bias can be assumed in our joint position prediction. We estimate this bias as the minimum error for each joint, over the whole training set, between our joint position predictions and the ground truth. The evaluations on the validation sets are plotted in Fig. 8, where a second curve for our method is added, showing the error of the same prediction with the bias subtracted. Since this bias might be too optimistic, we believe that our real joint prediction error should be somewhere between the two curves, showing that our method compares closely to several state-of-the-art methods on both datasets.

4.6. Method Speed

We conducted our experiments on an Intel i7 860 (from 2010), 8GB of RAM with an Nvidia Geforce GTX 1060. A forward pass through the network for predicting a single input image takes 3.5ms, which is practically real-time. Training unsupervised for 10 epochs, as we did, takes around 30 minutes.

5. Discussion and Conclusions

Through quantitative and qualitative evaluations, we showed that utilizing our method, a base convolutional model trained purely on synthetic data can be automatically refined to new unlabelled depth images.

This method could be utilized both as an extension to previous data-driven methods (under minimal constraints), as well as a stand alone method for 3D pose estimation.

The ability of the network to adapt to new poses and shapes, while running real-time on CPU, unlocks further applications, such as personalized gesture recognition or hand-tracking, which could be integrated into smart-phones. Even though we tackle only single depth estimation, in the supplementary videos we show that it also applies to tracking (under minimal jittering).

However, this method has limitations, too. We assume that we adapt the base CNN to a single hand shape only. For op-

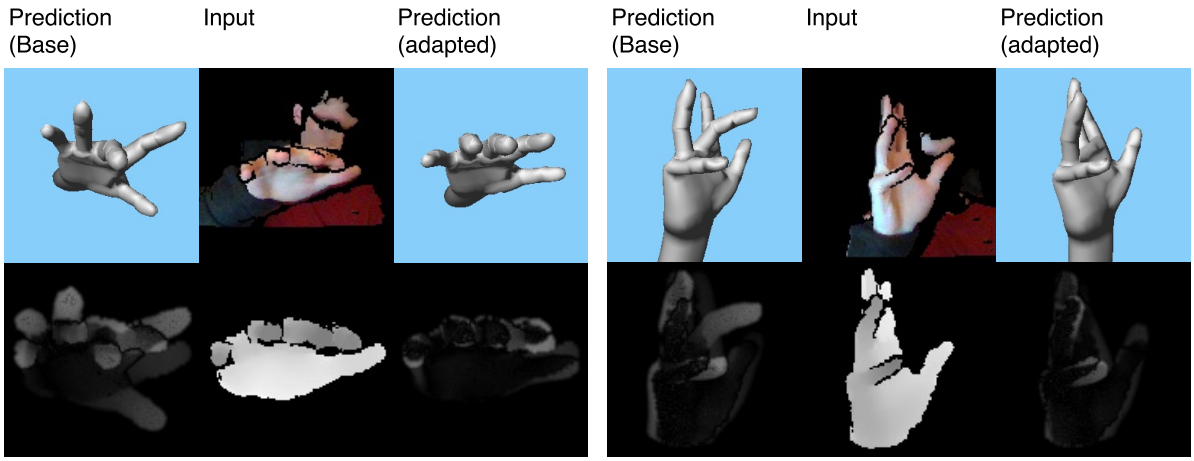

Figure 6. Two qualitative examples from our validation set are shown, after training with $50k$ images from our training set without annotations. For each block, top row shows a 3D rendering of our hand model in the predicted pose before (left) and after training (right). For visual comparison, we demonstrate the RGB input image (center), which is not utilized by this method. The bottom row shows the absolute depth errors of the poses from above (left and right) (see Fig. 2 for details) and the input depth image (center).

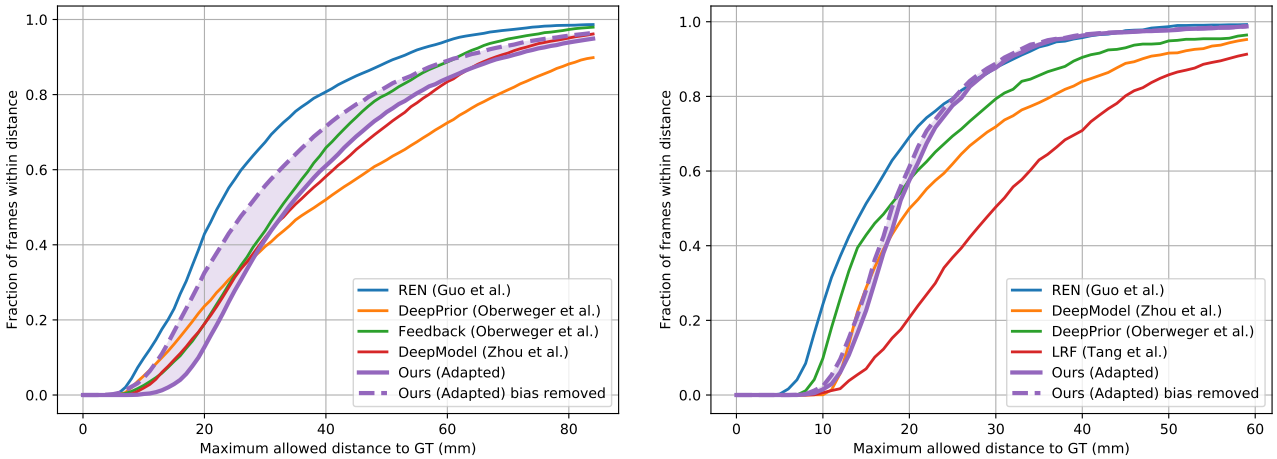

Figure 8. Comparison to state-of-the-art methods on the NYU [29] and ICVL [2] dataset. For one of our curves (dashed), we remove the bias introduced from the missmatch between our hand model and the ones used as groundtruths for each dataset.

timal performance, we require therefore a consistent hand shape and also a good hand segmentation. In order to cope with that, we could potentially extend our model to predict the hand shape for each input image, similar to what we do for the pose estimation. This is possible since our current model internally adapts to a hand shape, in order to help the pose refinement.

We also believe that retraining the network with images of a new user is a possible option, if a personalized hand tracker is desired, since training with 50K images takes only about 30 minutes, when trained from scratch.

We require our base CNN to make reasonable predictions, however we have shown that training a CNN merely on synthetic depth data yields sufficient initial estimations.

Even with some of the assumptions violated (*e.g.* non-consistent segmentation when using ICVL or NYU, label mismatching), we could show comparable results to state-of-the-art on two public datasets.

Lastly, we envisage that such method could be applied, without loss of generality, to human pose estimation tasks under minimal changes to the underlying 3D model.

Acknowledgement. We thank Riccardo Roveri and Lukas Rahmann for their insights on the rendering function in Sec.3.3.2 and Andrea Tagliasacchi for his overall comments and suggestions.

References

[1] Tech report: Projection of unordered point sets and generation of distance field images with gaussian interpolation. Technical report. 3, 4

[2] A. T. T.-K. K. Danhang Tang, Hyung Jin Chang. Latent regression forest: Structured estimation of 3d articulated hand posture. *CVPR*, 2014. 1, 5, 6, 7, 8

[3] X. Deng, S. Yang, Y. Zhang, P. Tan, L. Chang, and H. Wang. Hand3d: Hand pose estimation using 3d neural network. *CoRR*, abs/1704.02224, 2017. 2

[4] E. Dibra, H. Jain, A. C. Öztireli, R. Ziegler, and M. H. Gross. Hs-nets: Estimating human body shape from silhouettes with convolutional neural networks. In *Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016*, pages 108–117, 2016. 5

[5] E. Dibra, H. Jain, A. C. Öztireli, R. Ziegler, and M. H. Gross. Human shape from silhouettes using generative hks descriptors and cross-modal neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, July 21-26, 2017*, 2017. 5

[6] L. Ge, H. Liang, J. Yuan, and D. Thalmann. Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns. In *Proc. CVPR*, 2016. 1, 2

[7] H. Guo, G. Wang, X. Chen, C. Zhang, F. Qiao, and H. Yang. Region ensemble network: Improving convolutional network for hand pose estimation. *CoRR*, abs/1702.02447, 2017. 7

[8] J. S. S. III, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan. Depth-based hand pose estimation: methods, data, and challenges. In *IEEE International Conference on Computer Vision, ICCV*, 2015. 1

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 3, 5

[11] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 2, 3, 4

[12] M. M. Loper and M. J. Black. OpenDR: An approximate differentiable renderer. In *Computer Vision – ECCV 2014*, volume 8695 of *Lecture Notes in Computer Science*, pages 154–169. Springer International Publishing, Sept. 2014. 3, 5

[13] S. Melax, L. Keselman, and S. Orsten. Dynamics based 3d skeletal hand tracking. In *Proceedings of Graphics Interface 2013*, GI '13, pages 63–70, Toronto, Ont., Canada, Canada, 2013. Canadian Information Processing Society. 1, 2, 4, 5

[14] M. Oberweger, G. Riegler, P. Wohlhart, and V. Lepetit. Efficiently creating 3d training data for fine hand pose estimation. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4957–4965, 2016. 1

[15] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands deep in deep learning for hand pose estimation. *CoRR*, abs/1502.06807, 2015. 1, 2, 7

[16] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a feedback loop for hand pose estimation. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 3316–3324, Washington, DC, USA, 2015. IEEE Computer Society. 2, 3, 7

[17] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proceedings of the British Machine Vision Conference*, pages 101.1–101.11, 2011. 1, 2, 5

[18] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106–1113, 2014. 1, 2, 5

[19] E. Remelli, A. Tkach, A. Tagliasacchi, and M. Pauly. Low-dimensionality calibration through local anisotropic scaling for robust hand model personalization. In *Proceedings of the International Conference on Computer Vision*, 2017. 5

[20] M. Schröder, J. Maycock, H. Ritter, and M. Botsch. Real-time hand tracking using synergistic inverse kinematics. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 1, 2

[21] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3633–3642, New York, NY, USA, 2015. ACM. 1, 2, 5

[22] A. Sinha, C. Choi, and K. Ramani. Deephand: Robust hand pose estimation by completing a matrix imputed with deep features. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2

[23] 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. 1

[24] A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly. Robust articulated-icp for real-time hand tracking. *Computer Graphics Forum (Symposium on Geometry Processing)*, 34(5), 2015. 1, 2, 4, 5

[25] D. Tang, T. H. Yu, and T. K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *2013 IEEE International Conference on Computer Vision*, pages 3224–3231, Dec 2013. 1

[26] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, S. Izadi, R. Banks, A. Fitzgibbon, and J. Shotton. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Trans. Graph.*, 35(4):143:1–143:12, July 2016. 1, 2

[27] A. Tkach, M. Pauly, and A. Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia)*, 2016. 1, 2

[28] A. Tkach, A. Tagliasacchi, E. Remelli, M. Pauly, and A. Fitzgibbon. Online generative model personalization for

hand tracking. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia)*, 2017. 5

[29] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics*, 33, August 2014. 1, 2, 5, 6, 7, 8

[30] Q. Ye, S. Yuan, and T. Kim. Spatial attention deep net with partial PSO for hierarchical hybrid hand pose estimation. *CoRR*, abs/1604.03334, 2016. 1, 2

[31] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T. Kim. Bighand2.2m benchmark: Hand pose dataset and state of the art analysis. *CoRR*, abs/1704.02612, 2017. 1

[32] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei. Model-based deep hand pose estimation. In *IJCAI*, 2016. 1, 2, 3, 4, 5, 7