Monocular RGB Hand Pose Inference from Unsupervised Refinable Nets Supplementary

Endri Dibra, Silvan Melchior, Ali Balkis, Thomas Wolf, Cengiz Öztireli, Markus Gross
Department of Computer Science, ETH Zürich
{{edibra,cengizo,grossm}@inf, {silvanm,abalkis,wolftho}@student}.ethz.ch

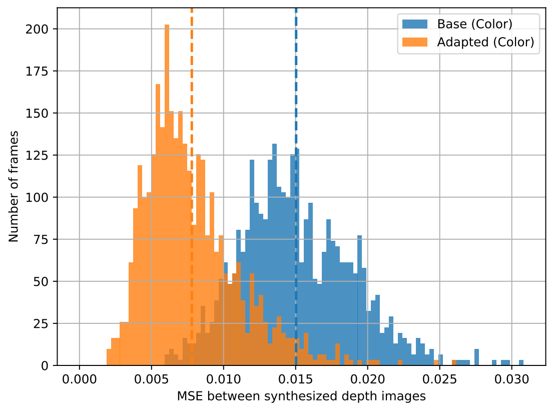

Figure 4. MSE histogram computed over pixel (depth) image difference for the network before and after refining unsupervised. We attach a corresponding video (ColorAdapted).

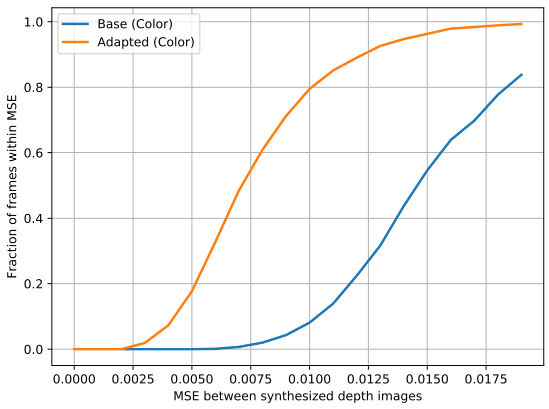

Figure 5. ROC curve corresponding to Fig.4.

1. Introduction

In this supplementary material, we first provide further qualitative experiments on the accuracy of our hand pose inference method (including videos). Then we provide some complementary explanation of our dataset creation and experiments, where the limitations are discussed. In the end we provide the full architecture and training details for the detection, segmentation and inference part, along with the gradient derivations and implementation details of our unsupervised learning component (Sec.3.3 from the paper).

2. Real Hands Results

In Fig.2, we show further results of poses inferred from $SynthNet$, with input images from the HGR dataset [4, 7, 3] and one additional individual performing various poses in Fig.3. We would like to stress that $SynthNet$ is only trained with our synthetic data and has not seen any sample from the HGR dataset.

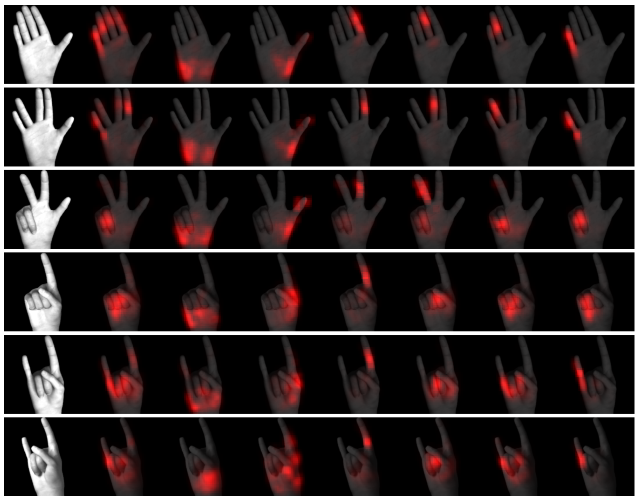

Figure 6. Error Contribution Heatmap. Shown are six images from our dataset. For each input image, we visualize the contribution to the mean squared error of the complete pose, arm (wrist), thumb, pointer, middle, index and pinky (from left to right).

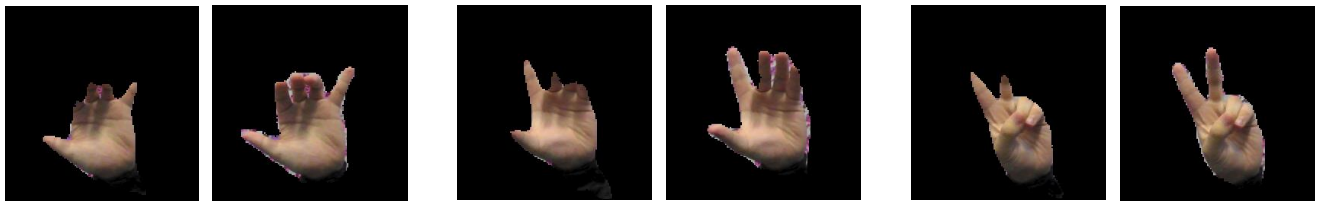

Figure 1. Three examples of segmentation improvement on $StereoDS$ before (left) and after (right) adding our synthetic training data. Added as a reference to Table 1 from the paper.

3. Refinement and Intersection Handling

Our method has limitations too. Firstly, since it does not have an explicit intersection handler, at prediction time, some intersections occur. Secondly, due to the thumb discrepancy between real and synthetic data, there is a difficulty of estimating closed fists (with the thumb in) directly from synthetic data, as well as poses like in Fig.2 ($1st$ Column, $3rd$ row).

Kinematic Constraints. Due to our database construction the cases where fingers intersect are minimized, however they exist. One way to tackle this could be to add an intersection handling term as part of the loss function. This however would help only at training time, during a potential unsupervised refinement. In order to make sure that no intersection happens during prediction, we, for completeness and comparison (Fig.7 and video), provide a small extension of the pose prediction pipeline, by calculating a new pose ϕ through minimizing the energy function in equation 1. The first norm penalizes large deviations from the pose $\overline{\phi}$ predicted by the CNN. A cylindric model is utilized to penalize intersections in the second term, where each finger consists of three cylinders. A cylinder p is determined by a radius r_p (obtained from our hand model) and a segment $s_{p1}^{p2}(\phi)$ serving as axis (computed from the rotation determined from ϕ). We denote with d the distance function between two segments.

$$E_1(\phi) = \phi - \overline{\phi}^2 + \lambda \left[\sum_{(p,q)\in I} max\{0, r_p + r_q - d(s_{p1}^{p2}, s_{q1}^{q2})\}^2 \right]^2 \quad (1)$$

To solve the optimization problem imposed by equation 1, we utilize the Ceres Solver [1]. Fig.8 shows qualitative examples of refinements with $\lambda = 10$. The additional step helps to correct small failures. A bit of intersection is still allowed, in order to simulate flesh interaction with our model. A video (BaseVsRefinedVsConstrained) is also attached showing predictions before (base) and after refinement with the kinematic constraints (constrained).

Heat Map Visualizations. In order to understand the difficulty of estimating the thumb position, we visualize features learned by our network, by relating image positions to error contributions. We adopt a technique introduced by Zeiler *et al.* [8] to regression. A black box is moved over different poses. For each position, the increase of error compared to the original image is measured and finally visualized as heatmap. We calculate the mean squared error over different sets of parameters. Fig.6 shows heat-map examples for a variety of poses. As it can be noticed, most of the fingers demonstrate a high error throughout the whole finger when that part is missing (as we would expect), except for the thumb, that shows a high error in its lower part, however the upper part is not picked by the network. We believe this is due to the discrepancy between synthetic and most real thumbs, which could be alleviated by providing the network with real examples, based on Sec.3.2 from the paper, and demonstrated below.

Refinement Video. We attach a video that shows the input image, the prediction from $SynthNet$ (base), the prediction after a semi-supervised refinement as in Sec.3.2 from the paper ($SynthNet$ which has seen real RGB unlabelled data (Refined)), and the prediction after the inter-penetration constraint handling (Constrained). Fig.7 contains a frame from the video, where it is observed how the fist closing is handled for all the cases. As it can be noticed, refining the network produces the most plausible results, demonstrating the advantage of the model. (Please note that temporal smoothing is applied to the video).

4. Unsupervised Refinement Curves and Video

We attach two videos (ColorAdapted and ColorVsDepth). The first one compares the network trained only synthetic data $SynthNet$ (top) to the one adapted unsupervised to real unlabelled pairs of monocular RGB and depth data from an individual, $RefNet$ (bottom). This refers to Sec.3.3 and Sec.5.3 in the paper as well as Fig.6. Please note that not only the pose prediction quality is enhanced, but also jitter is removed significantly (we apply no temporal smoothing to the video). We additionally show in Fig.4 and Fig.5 the MSE error computed over pixel (depth) image difference for the network before and after refining unsupervised, and notice a considerable improvement.

A second video is provided where our method is compared

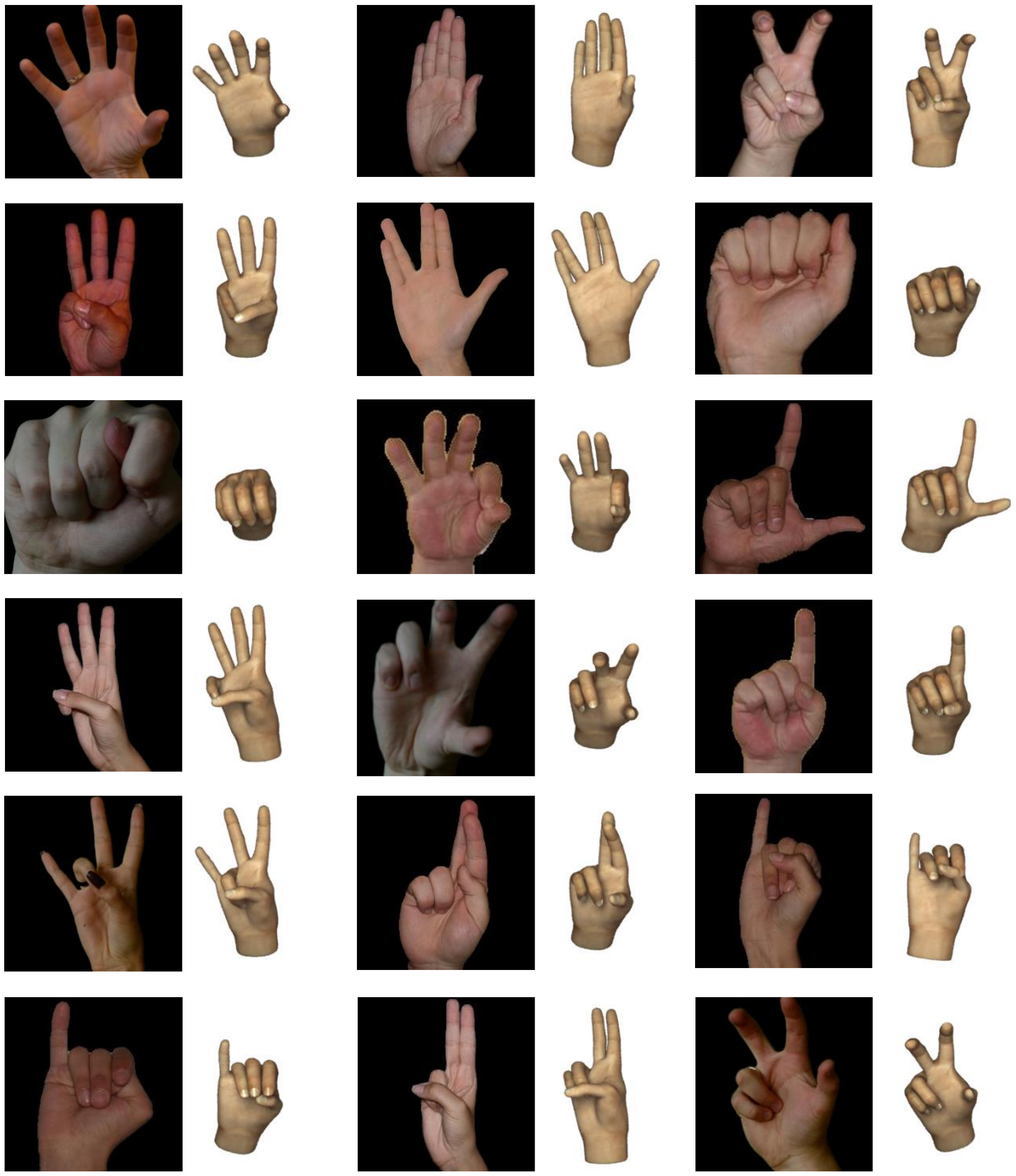

Figure 2. Qualitative results on various hand poses, shapes and color.

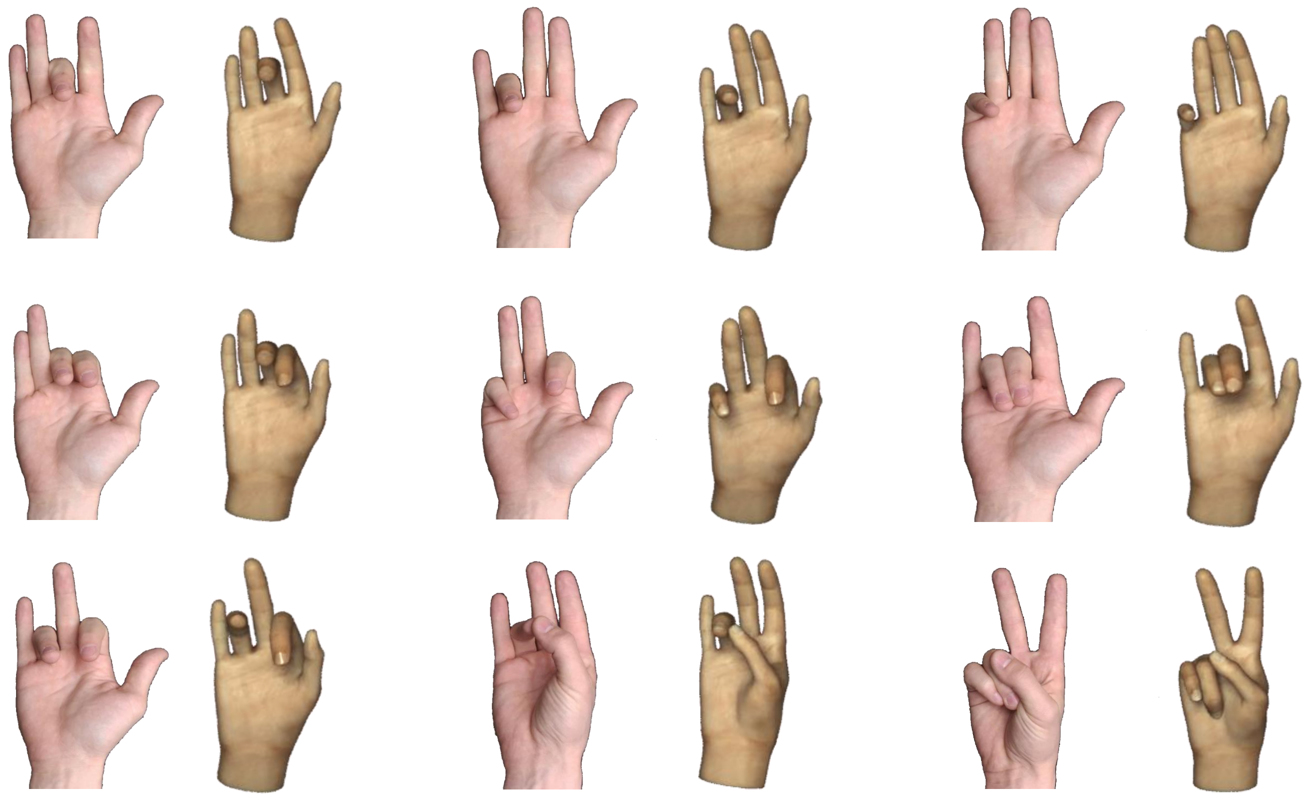

Figure 3. Qualitative results on one individual in various hand poses.

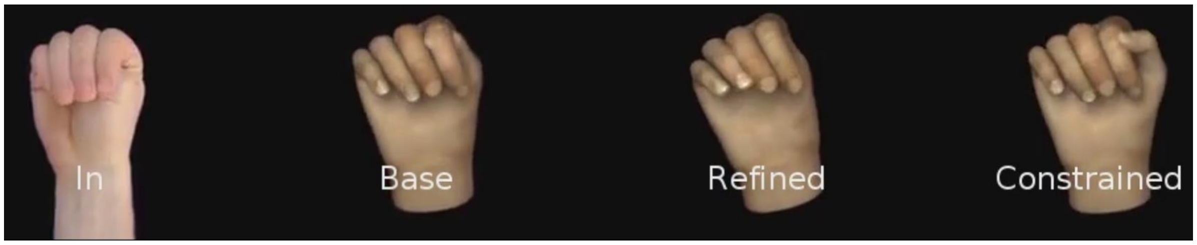

Figure 7. One frame from the attached video. From left to right : input image, $SynthNet$ prediction, $SynthNet$ refined semi-supervised. $SynthNet$ with interpenetration constraint handling prediction.

to a method based on depth images only [2] (our input is monocular RGB while theirs is depth), and a similar prediction accuracy is noticed. For the reader courtesy, we additionally provide the per-frame MSE over the entire video, the MSE histogram and ROC curve in Fig.9,10 and 11. Please note that what is important to be compared from the graphs, except for the increased accuracy due to adaptation, is Base (Depth) vs Adapted (Color). Base (Depth) can be thought of as a CNN method from the literature trained on depth images while Adapted (Color) is our method that has been trained on our synthetic RGB images, and has only seen depth images and utilized them in an unsupervised manner, at little capturing cost. We believe that this example shows the potential of CNN RGB based methods to work on par with Depth-based ones.

5. Base Poses

As mentioned in the paper we create our synthetic dataset by starting from base poses, which are later interpolated. We enumerate the thumb poses separately and firstly focus on the other four fingers. We fix three possible opening states of a finger: fully open, partially closed (metacarpophalangeal joint still stretched) and fully closed. Furthermore, we assume that if some fingers are not fully stretched, they are closed the same way (either all of them partially closed or fully closed). The side-movements are

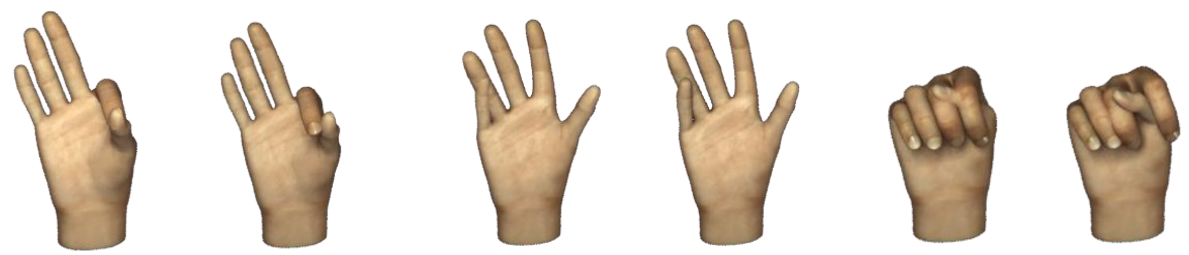
Figure 8. Kinematic Constraints Demonstration. The left image of each pair shows an initial prediction, the right image a refined version using kinematic constraints.

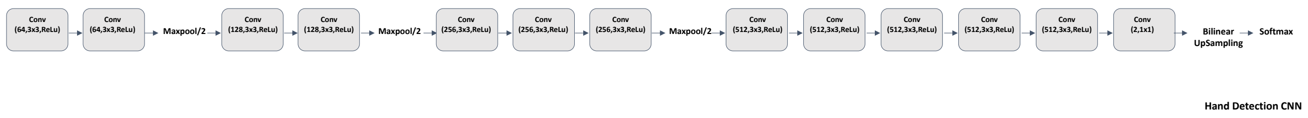

Figure 12. CNN architecture of the detection part.

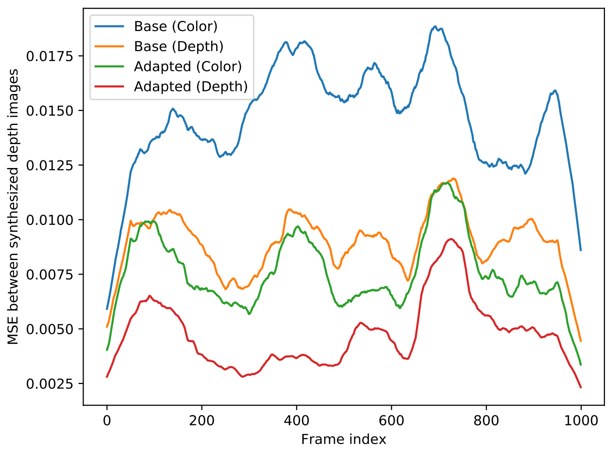

Figure 9. MSE variation per frame of our (Color) and [2] method (Depth) both before (Base) and after (Adapted) after unsupervised refinement.

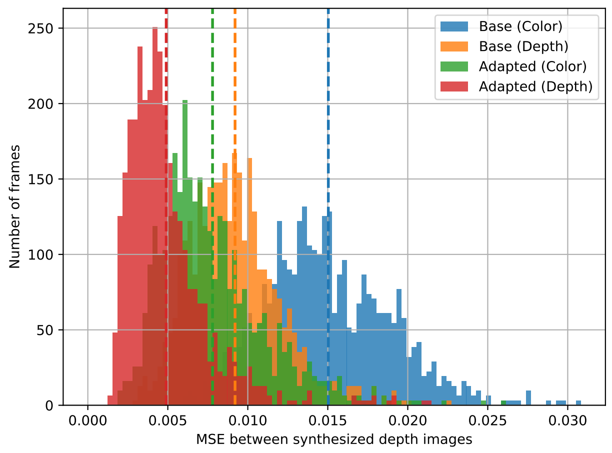

Figure 10. MSE histogram of our (Color) and [2] method (Depth) both before (Base) and after (Adapted) after unsupervised refinement.

combined with the opening state of a finger, *e.g.* they are ignored when the fingers are partially or fully closed, due to the human hand limitations [6]. A further simplifying assumption we made was to enumerate side-movements by counting the gaps between the fingers as explained below:

- If all four fingers are fully open, there are three gaps in between. Each gap can be either open or closed, giving a total of $2^3 = 8$ combinations.
- If three fingers are fully open, the number of gaps depends on the (partially) closed finger, being either two or one. Since the closed finger can be either fully or partially closed, we get a total of $2(2^2+2^1+2^1+2^2) = 24$ poses.
- If two fingers are fully open, we get a total of $2(2^1 + 2^0 + 2^1 + 2^0 + 2^0 + 2^1) = 18$ poses with the same considerations as in the previous case.
- If only one finger is fully open, we do not have side-movements with our assumptions, giving a total of $2(2^0 + 2^0 + 2^0 + 2^0) = 8$ poses.
- If all fingers are closed, they can be either partially of fully closed, giving a total of 2 poses.

Having in mind the interpolation to be performed, we fixed six different poses of the thumb, giving a total of $6 \times 60 = 360$ base poses. Additionally, we added four poses where the thumb touches one of the remaining fingers each, since

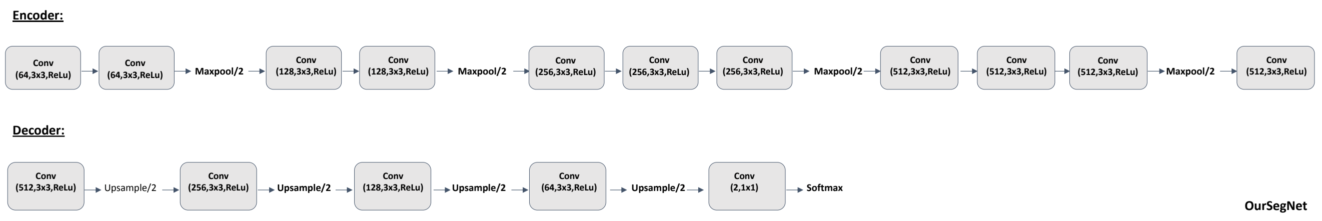

Figure 13. CNN architecture of $OurSegNet$.

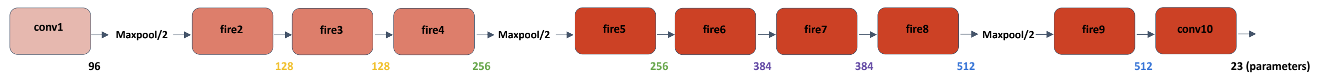

Figure 14. CNN architecture for pose inference $SynthNet$.

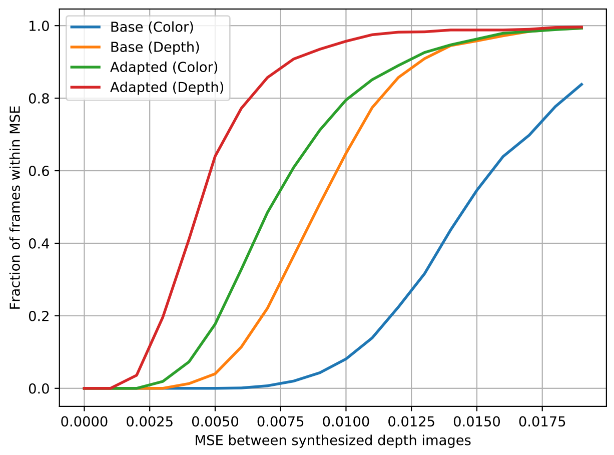

Figure 11. ROC curve corresponding to Fig.10.

the above setting of separating the thumb from the remaining fingers does not handle this. We also added 6 poses with crossed fingers, because the focus on the gap between the fingers does not capture this. In the end, 29 base poses were created inspired by the HGR dataset, giving a total of 399 base poses.

6. Architectures

We illustrate in Fig. 12, Fig. 13 and Fig. 14 the hand detection, segmentation and pose inference model in more detail.

6.1. Training Details

We train *HandSegNet* with Adam optimizer [5] and an initial learning rate of 10^-5 with decay 5×10^-4, changed to 10^-6 and 10^-7 over 10 epochs, and *OurSegNet* with Stochastic Gradient Descent and the same initial learning rate for 30 epochs. *SynthNet* was trained utilizing Adam optimizer, with an initial learning rate of 10^-4 over 10 epochs with a batch size of 100, while *RefNet* with Adam optimizer and learning rate of 4×10^-3 for over 40 epochs with a batch size of 1000.

7. Depth Loss Details

Here we provide implementation details for the linear blend skinning (LBS) and the depth renderer part. We implement custom operations of the form $f : A \mapsto B$ in Tensorflow, with a given gradient $g(f)$ and $g : B \mapsto \mathbb{R}$ an arbitrary loss function.

Linear Blend Skinning. Let J be the number of joints and $T = [T_1, \dots, T_J]$, where $T_i \in \mathbb{R}^{4\times4}$, the matrix that transforms from joint space of joint i to view space of the hand pose α. Let $PCL = [p_1, \dots, p_n]$ with $p_i \in \mathbb{R}^4$ a point cloud with points in homogeneous coordinates and $w_{i,j} \in W \in \mathbb{R}^{n\times J}$ the weights that define the boundness of point p_i to a joint $j \in [J]$. We derivate w.r.t. T by first writing f in terms of scalar values with coordinate axis $c \in [4]$:

$$f_{i,c}(PCL, T) = \sum_{d=1}^{4} \sum_{j=1}^{J} w_{i,j} T_{j,c,d} p_{i,d} \qquad (2)$$

and its derivative $f := f$:

$$\frac{\partial f_{i,c}(PCL, T)}{\partial T_{k,l,j}} = \mathbb{1}_{\{l=c\}} w_{i,k} p_{i,j} \qquad (3)$$

Utilizing the chain rule we can differentiate $g(f)$ as :

$$\frac{\partial}{\partial T_{k,l,j}} g(f(T, PCL)) = \sum_{i=1}^{n} \sum_{c=1}^{4} \frac{\partial g(f)}{\partial f_{i,c}} \frac{\partial f_{i,c}(PCL, T)}{\partial T_{k,l,j}} \tag{4}$$

$$= \sum_{i=1}^{n} \frac{\partial g(f)}{\partial f_{i,l}} w_{i,k} p_{i,T} \tag{5}$$

In order to update PCL, we also need to differentiate f w.r.t. PCL, by adding the notion of the batch dimension. Let now $f \in \mathbb{R}^{n\times 4} \times \mathbb{R}^{B\times n\times 4} \mapsto \mathbb{R}^{B\times n\times 4\times 4}$ with n the number of points and B the batch size. Since there is a T for each batch element now, it subsequently has four dimensions. We can re-write Eq. 2 as :

$$f_{b,i,c}(PCL, T) = \sum_{d=1}^{4} \sum_{j=1}^{J} w_{i,j} T_{b,j,c,d} p_{i,d} \tag{6}$$

For a batch index b, point index i and coordinate index c. Taking the derivative of $f := f$ gives:

$$\frac{\partial f_{b,i,c}(PCL, T)}{\partial p_{k,l}} = \mathbb{1}_{\{k=i\}} \sum_{j=1}^{J} w_{i,j} T_{b,j,c,l} \tag{7}$$

Thus, $g(f)$ is given by:

$$\frac{\partial}{\partial p_{k,l}} g(f(T, PCL)) = \sum_{b=1}^{B} \sum_{i=1}^{n} \sum_{c=1}^{4} \frac{\partial g(f)}{\partial f_{b,i,c}} \frac{\partial f_{b,i,c}(PCL, T)}{\partial p_{k,l}}$$

$$= \sum_{b=1}^{B} \sum_{c=1}^{4} \frac{\partial g(f)}{\partial f_{b,k,c}} \sum_{j=1}^{J} w_{i,j} T_{b,j,c,l} \tag{8}$$

Depth Renderer. We have $f := f^{\text{depth}}$. The derivative of $g(f)$ for $c \in \{x, y, z\}$, and an image size $H \times W$ (here 120×120) is given by:

$$\frac{\partial}{\partial p_{l,c}} g(f(P)) = \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{\partial g(f_{i,j})}{\partial f_{i,j}} \frac{\partial f_{i,j}(PCL)}{\partial p_{l,c}} \tag{9}$$

with

$$\frac{\partial f_{i,j}(PCL)}{\partial p_{l,x}} = h(l, PCL)(j - p_{l,x}) \tag{10}$$

$$\frac{\partial f_{i,j}(PCL)}{\partial p_{l,y}} = h(l, PCL)(i - p_{l,y}) \tag{11}$$

For an efficient implementation of Eq. 9, we loop only over a range of $\{\lfloor p_{l,x} - r \rfloor, \ldots, \lceil p_{l,y} + r \rceil\}$ for each point p_l utilizing the finite spatial support of ϕ, which is defined for some indices i, j with $1 \leq i, j \leq H, W$ fulfilling $l = \text{argmax}_k(\text{depth}_{i,j}(p_k))$:

$$\frac{\partial f_{i,j}(PCL)}{\partial p_{l,z}} = -\mathbb{1}_{\{l=\text{argmax}_k(\text{depth}_{i,j}(p_k))\}} \phi(p_l) \tag{12}$$

where

$$h(l, PCL) = \mathbb{1}_{\{l=\text{argmax}_k(\text{depth}_{i,j}(p_k)) \wedge \phi(p_l)>0\}} \frac{4(1 - p_{l,z})}{r^2} \left(1 - \frac{(j - p_{l,x})^2 + (i - p_{l,y})^2}{r^2}\right) \tag{13}$$

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org. 2

[2] E. Dibra, T. Wolf, A. C. Öztireli, and M. H. Gross. How to refine 3d hand pose estimation from unlabelled depth data ? In *Fifth International Conference on 3D Vision, 3DV 2017, Qingdao, China, 2017*, 2017. 4, 5

[3] T. Grzejszczak, M. Kawulok, and A. Galuszka. Hand landmarks detection and localization in color images. *Multimedia Tools and Applications*, 75(23):16363–16387, 2016. 1

[4] M. Kawulok, J. Kawulok, J. Nalepa, and B. Smolka. Self-adaptive algorithm for segmenting skin regions. *EURASIP Journal on Advances in Signal Processing*, 2014(170):1–22, 2014. 1

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6

[6] J. Lee and T. L. Kunii. Model-based analysis of hand posture. *IEEE Comput. Graph. Appl.*, 15(5):77–86, Sept. 1995. 5

[7] J. Nalepa and M. Kawulok. Fast and accurate hand shape classification. In S. Kozielski, D. Mrozek, P. Kasprowski, B. Malysiak-Mrozek, and D. Kostrzewa, editors, *Beyond Databases, Architectures, and Structures*, volume 424 of *Communications in Computer and Information Science*, pages 364–373. Springer, 2014. 1

[8] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, pages 818–833, 2014. 2