Supplemental Material for: FreiHAND: A Dataset for Markerless Capture of Hand Pose and Shape from Single RGB Images

Christian Zimmermann[1], Duygu Ceylan[2], Jimei Yang[2], Bryan Russell[2], Max Argus[1], and Thomas Brox[1]

[1]University of Freiburg
[2]Adobe Research

Project page: https://lmb.informatik.uni-freiburg.de/projects/freihand/

1. Cross-dataset generalization

In this section we provide additional information on the single view pose estimation network used in the experiment and other technical details. The datasets used in the experiment show slight differences regarding the hand model definition: Some provide a keypoint situated at the wrist while others define a keypoint located on the palm instead. To allow a fair comparison, we exclude these keypoints, which leaves 20 keypoints remaining for the evaluation. In the subsequent sections we, first provide implementation details in 1.1, which includes hyperparameters used and the network architecture. Second, we analyze the influence that using a pretrained network has on the outcome of the experiment in 1.2.

1.1. Implementation details

We chose our hyper parameters and architecture similar to [4]. The network consists of an encoder decoder structure with skip connections. For brevity we define the building blocks *Block0* (see Table 2) *Block1* (see Table 3), *Block2* (see Table 4), *Block3* (see Table 5) and *Block4* (see Table 6). Using these, the network is assembled according to Table 1. All blocks have the same number of channels for all convolutions throughout. An exception is *Block4*, which has 128 output channels for the first two and 42 for the last convolution. The number 42 arises from 21 keypoints we estimate 2D locations and depth for. Skip connections from *Block1* to *Block3* always branch off after the last convolution (id 3) of *Block1* using the respective block that has the same spatial resolution.

We train the network for 300 k iterations with a batch size of 16. For optimization we use the Momentum solver with an initial learning rate of 0.001 and momentum of 0.9. Learning rate is lowered to 0.0001 after iteration 150 k.

id	Name	Dimensionality
	Input image	$128 \times 128 \times 3$
1	Block0	$128 \times 128 \times 64$
2	Block1	$64 \times 64 \times 128$
3	Block1	$32 \times 32 \times 128$
4	Block1	$16 \times 16 \times 128$
5	Block1	$8 \times 8 \times 128$
6	Block1	$4 \times 4 \times 128$
7	Block1	$2 \times 2 \times 128$
8	Block2	$4 \times 4 \times 256$
9	Block3	$8 \times 8 \times 128$
10	Block3	$16 \times 16 \times 128$
11	Block3	$32 \times 32 \times 128$
12	Block3	$64 \times 64 \times 128$
13	Block3	$128 \times 128 \times 128$
14	Block4	$128 \times 128 \times 42$

Table 1: Our single view network architecture used for 3D pose estimation in the cross-dataset generalization experiment.

id	Name	Kernel	Stride
1	Conv. + ReLU	3×3	1
2	Avg. Pool	4×4	1
3	Conv. + ReLU	3×3	1
4	Avg. Pool	4×4	1

Table 2: Block0.

1.2. Pretrained network

We provide an additional version of the proposed cross-dataset generalization experiment, which shows the influence of using a pretrained network. For this purpose, we use a *ImageNet* pretrained *ResNet50* backbone, which we

id	Name	Kernel	Stride
1	Conv. + ReLU	3×3	1
2	Avg. Pool	4×4	2
3	Conv. + ReLU	3×3	1

Table 3: Block1.

id	Name	Kernel	Stride
1	Conv. + ReLU	3×3	1
2	Upconv.	4×4	2

Table 4: Block2.

id	Name	Kernel	Stride
1	Concat(Input, Block1 skip)	-	-
2	Conv. + ReLU	1×1	1
3	Conv. + ReLU	3×3	1
4	Upconv.	4×4	2

Table 5: Block3.

id	Name	Kernel	Stride
1	Conv. + ReLU	7×7	1
2	Conv. + ReLU	7×7	1
3	Conv.	7×7	1

Table 6: Block4.

train to learn a direct mapping from images to normalized 3D pose. From the original *ResNet50* we use the average pooled final features and process them using 2 fully connected layers with 2048 neurons each using ReLU activations and a final linear fully connected layer outputting the 63 parameters (= 21× 3D coordinates). Hyperparameters are identical to 1.1, except for the use of *ADAM* solver and an initial learning rate of 10^{-5}, which is lowered to 10^{-6} after iteration 150 k. The results are presented in Table 7, which shows that the average ranks are mostly unchanged compared to the results reported in the main paper. We witness a tendency of lower performance on the respective evaluation set, but better generalization to other datasets.

2. MVNet

2.1. Training loss

We experimented with different losses for training *MVNet* and witnessed large differences regarding their applicability to our problem. In addition to the loss described in the main paper, which we refer to as *Scorevolume* loss,

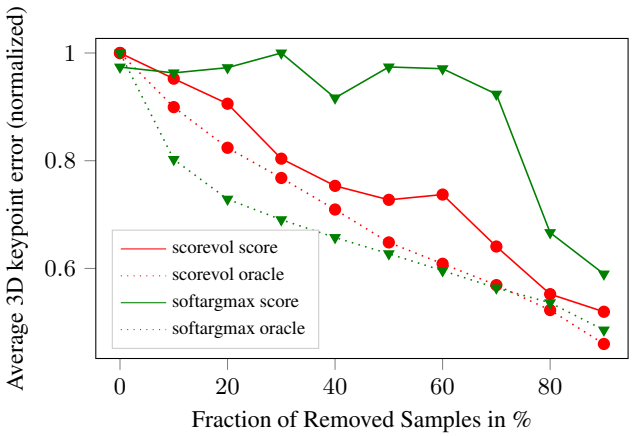

Figure 1: Shown is the average predictions 3D error of *MVNet* on a given dataset over the sparsification rate. When moving along the x-axis from left to right the remaining evaluation set gets smaller and the y-axis reports the error upon the remaining dataset. For each approach the resulting curves are shown, when the prediction score is used as solid lines, and when the ground truth error is used as dashed lines, which we refer to as oracle. A good score yields a line that stays close to the oracle line, which shows that *Scorevolume* trained networks learn much more meaningful scores than *Softargmax*.

we used the *Softargmax* loss formulation [4]. As reported in literature, we find that *Softargmax* achieves better results for keypoint estimation, but that the respective score for each prediction is less meaningful as a predictor of the expected error.

In both cases we define the score c of an prediction that *MVNet* makes as reported in the main paper and use the latent heat map of the *Softargmax* loss to calculate c. To analyze the relation between prediction score c and the expected error of the final prediction, we follow the methodology described in detail in [3], and report sparsification curves in Fig. 1. This plot analyses how the prediction error on a given dataset evolves by gradually removing uncertain predictions, as measured by the prediction score c. If the prediction score is a good proxy for the prediction error the curves should monotonically decrease to zero, because predictions with low score should identify samples with high error. The oracle curve shows the ideal curve for a respective loss and is created by accessing the ground truth error instead of using the prediction score, *i.e*. one is always removing the predictions with the largest error.

Fig. 1 shows that score of the *Scorevolume* loss shows much better behavior, because it stays fairly close to its oracle line. Which is in contrast to the *Softargmax* loss. We deduct from this experiment, that the scores that arise when training on a *Scorevolume* represent a more meaning-

train \ eval	STB	RHD	GAN	PAN	LSMV	FPA	HO3D	**Ours**	Average Rank
STB	**0.687**	0.247	0.151	0.263	0.220	0.138	0.207	0.244	5.6
RHD	0.480	**0.697**	0.200	0.353	0.490	0.156	**0.417**	0.403	2.9
GAN	0.184	0.198	**0.624**	0.217	0.229	0.182	0.188	0.233	5.8
PAN	0.447	0.367	0.221	**0.632**	0.454	0.205	0.264	0.345	3.0
LSMV	0.242	0.286	0.199	0.226	**0.640**	0.162	0.283	0.307	4.4
FPA	0.186	0.178	0.156	0.206	0.197	**0.705**	0.162	0.239	6.6
HO3D	0.254	0.311	0.198	0.206	0.338	0.148	-	0.313	5.4
Ours	0.520	0.399	0.205	0.395	0.509	0.208	0.416	**0.523**	2.0

Table 7: This table shows, cross-dataset generalization measured as area under the curve (AUC) of percentage of correct keypoints following [9]. In contrast to the table reported in the main paper an ImageNet pretrained *ResNet50* network is used for direct regression of normalized 3D pose. Entries are marked if they rank: **first**, second or third for each dataset.

ful measure of the algorithms uncertainty and therefore it should be used for our labeling procedure.

2.2. Implementation details

The part of network for 2D feature extraction is initialized with the network presented by Simon *et al*. [6]. We use input images of size 224×224, that show hand cropped images. For hand cropping we use a MobileNet architecture [2] that is trained on Egohands [1] and finetuned on a small, manually labeled, subset of our data. From the 2D CNN we extract the feature encoding $\boldsymbol{f}_i$ of dimension $28 \times 28 \times 128$ after 12 convolutional layers, which is unprojected into a $64 \times 64 \times 64 \times 128$ voxel grid $\boldsymbol{F}_i$ of size 0.4 meters. The voxel grid is centered at a 3D point hypothesis that is calculated from triangulating the detected hand bounding box centers we previously used for image cropping.

For joining the unprojected information from all cameras we average F_i over all views i and use a U-Net [5] like encoder-decoder architecture in 3D. We train the network for $100\,\mathrm{k}$ training steps using *ADAM* solver. Batch size is 8 for the 2D CNNs and 1 for the 3D CNN. *Scorevolume* loss is used with ground truth Gaussian targets having standard deviation of 2 voxel units.

3. Extended Evaluation of iterative procedure

In Fig. 2, we show how the 3D keypoint estimation accuracy of *MVNet* evolves over iterations. For comparison, we also shown how *MVNet* performs when trained on the Panoptic (*PAN*) dataset alone. While this gives insufficient performance, joint training on *PAN* and our dataset yields a large gain in performance in the first iteration and every other iteration provides further improvement.

4. Image Compositing

Here we study different methods for post processing our recorded images in order to improve generalization of composite images. Green screen indicates that images are used as is *i.e*. no additional processing step was used. *Cut&Paste* refers to blending the original image with a randomly sampled new background image using the foreground segmentation mask as blending alpha channel. Harmonization is a deep network based approach presented by Tsai *et al*. [7], which should improve network performance on composite images. Additionally, we experimented with the deep image colorization approach by Zhang *et al*. [8]. For this processing step we convert the composite image of the *Cut&Paste* method into a grayscale image and input it in the colorization method. Here we have the options *Auto*, in which the network hallucinates all colors and *Sample* where we provide the network with the actual colors in 20 randomly chosen sample points on each foreground and background. Examples of images these approaches yield are shown in Fig. 4. Table 8 reports results for the network described in 1.1 and Table 9 shows results when the pretrained baseline

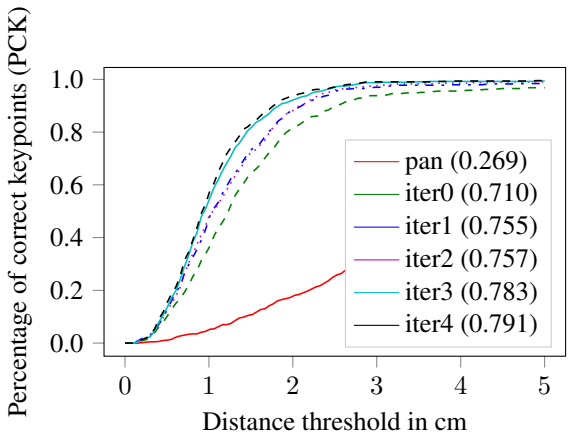

Figure 2: We show performance measured as 3D PCK of *MVNet* over different iterations of our procedure. It shows that training *MVNet* only on the Panoptic dataset is not sufficient to generalize to our data an that each iteration improves performance. In brackets the 3D AUC is reported.

is used instead. The two tables show, that post processing methods are more important when networks are trained from scratch. In this case, Table 8 shows that using each of the processing options yields roughly the same gain in performance and using all options jointly performs best. This option is chosen for the respective experiments in the main paper. When a pretrained network is used Table 9 reports already good results, when the network is only trained on green screen images. Interestingly, in this scenario the more elaborate post processing methods yield only a minor gain compared to the *Cut&Paste* strategy. We hypothesize these results are related to a significant level of robustness the pretrained weights possess. Please note that we can't use these algorithms in a similar manner for datasets that don't provide segmentation masks of the foreground object. Only *RHD* provides segmentation masks, which is why we show the influence the discussed processing methods have on its generalization. Table 10 shows that the discussed methods don't improve performance for *RHD* trained networks the same way. The results indicate that these strategies should not been seen as general data augmentation, but rather specific processing steps to alleviate the problem of green screen color bleeding we witness on our training dataset.

method \ eval	RHD	**Ours**
Green Screen	0.246	0.440
Cut&Paste	0.350	0.508
Harmonization [7]	0.443	0.628
Colorization Auto [8]	0.458	0.634
Colorization Sample [8]	0.494	0.643
Joint	**0.518**	**0.678**

Table 8: Network architecture as used in the main paper. When training networks from scratch, it is important to introduce background variation by inserting random backgrounds into the green screen recordings. Using post processing algorithms improves generalization.

method \ eval	RHD	**Ours**
Green Screen	0.338	0.436
Cut&Paste	0.386	0.468
Harmonization [7]	**0.416**	0.513
Colorization Auto [8]	0.368	0.478
Colorization Sample [8]	0.381	0.496
Joint	0.399	**0.523**

Table 9: Instead of the network architecture used in the main paper, we use a *ResNet50* Baseline as described in 1.2. When the network is initialized with weights that already show a certain level of robustness the importance of background removal and recombination with post processing is less pronounced, but still improves results substantially.

method \ eval	RHD	**Ours**
Original	**0.767**	0.508
Harmonization [7]	0.723	**0.517**
Colorization Auto [8]	0.726	0.472
Colorization Sample [8]	0.748	0.501
Joint	0.756	0.514

Table 10: Network architecture as used in the main paper, but instead of training on our dataset we train on *RHD* and apply the same post processing methods to it. We chose *RHD* as reference because it is the only dataset that also provides foreground segmentation masks, which are needed for the processing methods. The table shows that none of them yields clear improvements over using the original unaltered *RHD* dataset for training.

5. FreiHAND details

For the dataset we recorded 32 people and asked them to perform actions in front of the cameras. The set of non object actions included: signs from the american sign language, counting, move their fingers to their kinematic limits. The set of objects contains different types of workshop tools like drills, wrenches, screwdrivers or hammers. Different types of kitchen supply were involved as well, f.e. chopsticks, cutlery, bottles or BBQ tongs. These objects were either placed into the subjects hand from the beginning of the recording or hang into our setup and we recorded the process of grabbing the object.

Actions that contain interaction with objects include the following items: Hammer, screwdrive, drill, scissors, tweezers, desoldering pump, stapler, wrench, chopsticks, caliper, power plug, pen, spoon, fork, knive, remote control, cream tube, coffee cup, spray can, glue pistol, frisbee, leather cover, cardboard box, multi tool and different types of spheres (f. e. apples, oranges, styrofoam). The action were selected such that all major grasp types were covered including power and precision grasps or spheres, cylinders, cubes and disks as well as more specialized object specific grasps.

These recordings form the basis we run our iterative labeling procedure on, that created the dataset presented in the main paper. Some examples of it are shown in Fig. 5. Fig. 3 shows one dataset sample containing 8 images recorded at a unique time step from the 8 different cameras involved in our capture setup. One can see that the cameras capture

Part	Samples with object	Samples w/o object	male	female	total
Training (green screen)	2580	1490	2462	1608	4070
Evaluation (total)	339	156	290	205	495
Evaluation (plane space office)	119	40	63	96	159
Evaluation (outdoor)	114	47	88	73	161
Evaluation (meeting room)	106	69	139	36	175

Table 11: Distribution of labeled samples in our dataset across different aspects.

Ethnicity	Training	Evaluation
Caucasian (NA, Europe, ...)	14	11
South Asian (India, Pakistan, ...)	3	2
East Asia (China, Vietnam, ...)	2	0
Male subjects	15	6
Female subjects	9	5

Table 12: Gender and ethnicity distribution of recorded subjects in the dataset.

a broad spectrum of viewpoints around the hand and how for different cameras different fingers are occluded. Our datasets shape annotation is overlayed in half of the views.

Furthermore, we provide more qualitative examples of our single view shape estimating network in Fig. 6.

Distributions across genders and ethnicity's are reported in Table 12, whereas Table 11 shows the distribution of labeled samples across gender and object interaction.

The dataset was recorded using the following hardware: Two Basler acA800-510uc and six Basler acA1300-200uc color cameras that were hardware triggered using the GPIO module by numato. The cameras were equipped with fixed lenses with either 4mm or 6mm focal length. The recording setup is approximately forming a cube of edge length 1m with one of the color cameras being located in each of the corners. The subjects then reached inside the cubicle through one of the cubes' faces, which approximately put their hand at an equal distance to all the cameras. When recording for the evaluation split, we used ambient lighting. To improve lighting during green screen recording there were 4 powerful LED lights as used during recording with a video camcorder. These allowed to vary in terms of lighting power and light temperature during the recordings.

References

[1] Sven Bambach, Stefan Lee, David J. Crandall, and Chen Yu. Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1949–1957, 2015. 3

[2] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3

[3] Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667, 2018. 2

[4] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5 d heatmap regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 118–134, 2018. 1, 2

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[6] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1145–1153, 2017. 3

[7] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, Xin Lu, and Ming-Hsuan Yang. Deep image harmonization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3789–3797, 2017. 3, 4, 6

[8] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *arXiv preprint arXiv:1705.02999*, 2017. 3, 4, 6

[9] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. https://arxiv.org/abs/1705.01389. 3

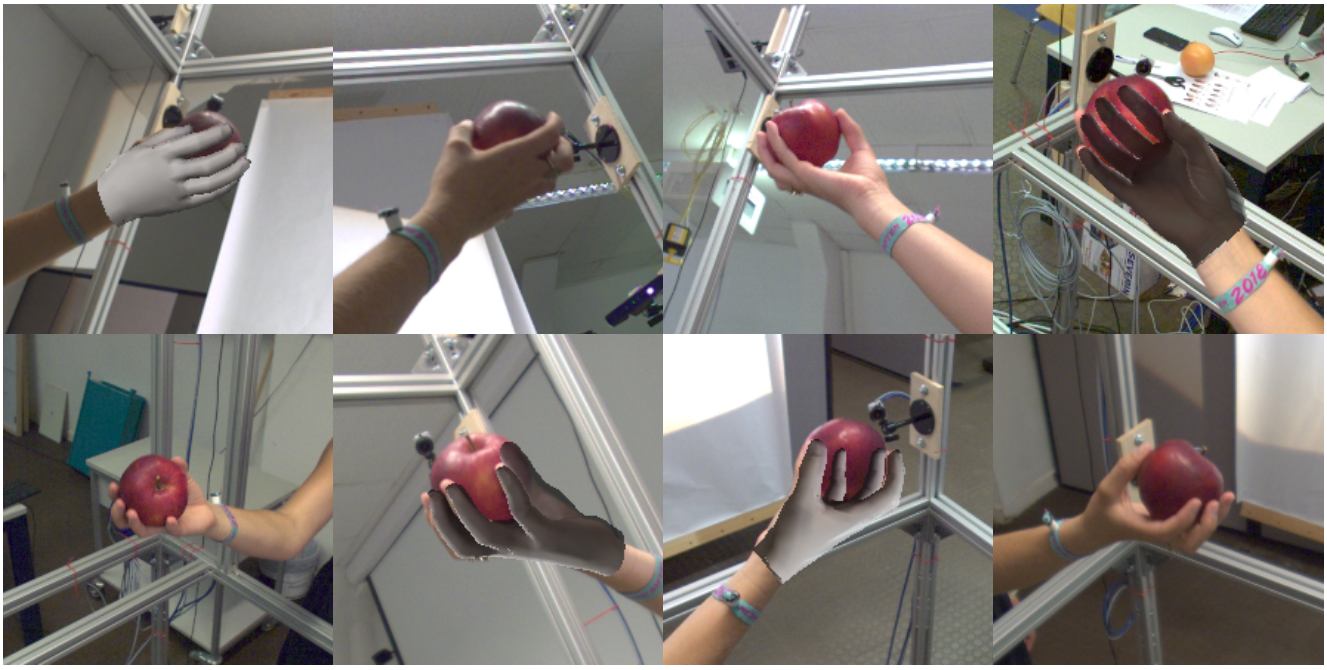

Figure 3: Here we show **one sample** of our dataset, which consists of **8 images** that are recorded at the same time instance. We overlay the shape label found with our method in some of the images.

Figure 4: Visualizations of our explored post processing options on the same sample. From left to right: Original frame, *Cut&Paste*, Harmonization [7], Colorization Auto [8], Colorization: Sample [8].

Figure 5: Examples from our proposed dataset showing images and hand shape annotations. It shows the final images that are used for training of single view methods with the original background replaced and [7] applied for post processing.

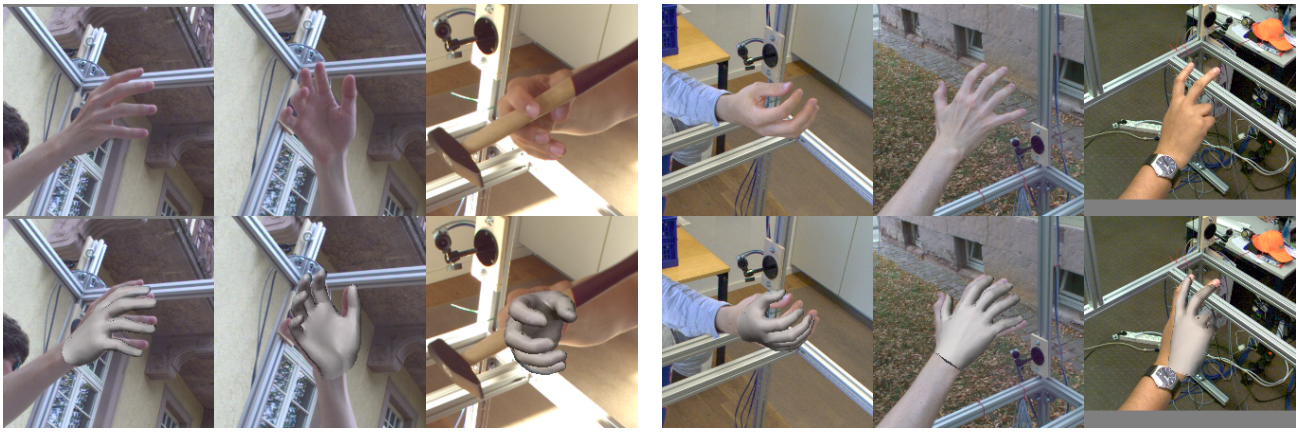

Figure 6: More qualitative examples of predicted hand shapes that our single view network makes. Again, we don't apply any alignment of the predictions with respect to the ground truth, which explains minor miss alignment whereas the hand articulation is captured correctly.