Self-Supervised 3D Hand Pose Estimation from monocular RGB via Contrastive Learning
[Supplementary Material]

1. Maximization of sim($\hat{z}_i$, $\hat{z}_j$)

In order to minimize the PeCLR cost function:

$$\mathcal{L}_{i,j} = -\log \frac{\exp\left(\text{sim}((\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_j)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp\left(\text{sim}(\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_k)/\tau\right)}, \quad (1)$$

We need to maximize the numerator $\text{sim}(\tilde{\boldsymbol{z}}_i, \tilde{\boldsymbol{z}}_j)$ where $\tilde{\boldsymbol{z}}_i = (t_i^g)^{-1}\boldsymbol{z}_i$. Here we show that this leads to the desired property of equivariance. For convenience, we restate the property of equivariance. Given an image $\boldsymbol{I}_i^n$, a transformation t_i^g, a model f is equivariant wrt. to t_i^g if:

$$t_i^g f(\boldsymbol{I}_i^n) = f(t_i^g(\boldsymbol{I}_i^n)). \quad (2)$$

Recall that for vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$, $\max_{\boldsymbol{x},\boldsymbol{y}} \text{sim}(\boldsymbol{x}, \boldsymbol{y}) = 1$. For a given $\boldsymbol{x}$, any $\boldsymbol{y} = a\boldsymbol{x}, a \in \mathbb{R}$ fulfills this property. Due to this, any scaling effect is removed and f can output any multiple of $\boldsymbol{x}$ to satisfy the equation. Hence we assume t_i^g to contain rotation and translation transformations. For simplicity, we set $a = 1$, hence $\boldsymbol{x} = \boldsymbol{y}$. In the following, we will drop the superscript g and n for ease of notation. Recall that $t_i(\boldsymbol{I}) = \boldsymbol{I}_i$, $f(\boldsymbol{I}_i) = \boldsymbol{z}_i$. We abuse notation slightly, where t_i corresponds to a function performing geometric transformation applied to an image or an affine matrix which can be applied to a vector. In other words, if t_i corresponds to a rotation by 90°, then $t_i(\boldsymbol{I})$ rotates the image by 90° and $t_i\boldsymbol{x}$ is a matrix vector multiplication, resulting in rotating vector $\boldsymbol{x}$ by 90°. We have:

$$\begin{aligned}
\hat{\boldsymbol{z}}_i &= \hat{\boldsymbol{z}}_j \\
(t_i)^{-1}\boldsymbol{z}_i &= (t_j)^{-1}\boldsymbol{z}_j \\
\boldsymbol{z}_i &= t_i(t_j)^{-1}\boldsymbol{z}_j \mid \text{def. } \hat{t}_{ij} := t_i(t_j)^{-1} \\
\boldsymbol{z}_i &= \hat{t}_{ij}\boldsymbol{z}_j \\
f(\boldsymbol{I}_i) &= \hat{t}_{ij}\boldsymbol{z}_j \\
f(t_i(t_j)^{-1}\boldsymbol{I}_j) &= \hat{t}_{ij}\boldsymbol{z}_j \\
f(\hat{t}_{ij}\boldsymbol{I}_j) &= \hat{t}_{ij}f(\boldsymbol{I}_j)
\end{aligned} \quad (3)$$

Hence, fulfilling Eq.1 leads to the desired property of equivariance in Eq.2 for rotation and translation in theory. In practice, due to the re-scaling procedure described in Sec. 3 in the main paper (and elaborated on in Sec. 2), it will be proportionate equivariant to the translation term. Note that this does not take into account clippings that occur when image rotate and translate out of bounds.

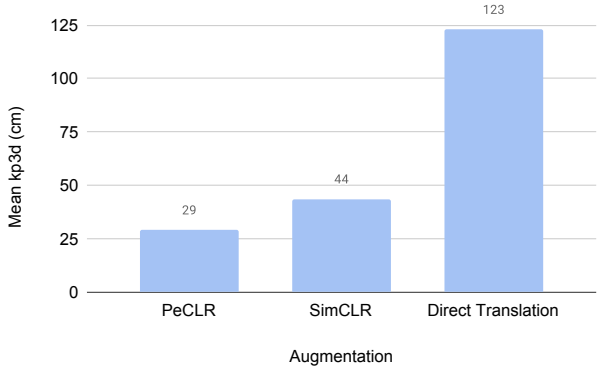


Figure 1: Comparing normalized and absolute scale inversion. Here, PeCLR represents the normalized translation and direct translation is a model inverting scale without normalization. For comparison, we show the results on SimCLR too. We see that directly applying a translation has a detrimental effect on performance, performing worse than SimCLR. However, using scaling PeCLR leads to superior performance.

2. Normalizing translation

We investigate the effect of our proposed translation normalization procedure. We briefly recap the main motivation behind normalizing translation, as mentioned in Sec. 3 of the main paper. Recall that PeCLR inverts all transformation performed on images in latent space.

Scaling and rotation are transformations that are performed relative to the magnitude. On the other hand, translation is performed in terms of an absolute quantity. Because images are translated in terms of pixels, inverting the translation in latent space by the same quantity may be detrimental. This is due to the differing magnitudes of the pixel and latent space. Therefore we translate the latent

space sample $\boldsymbol{z}^n$ by a quantity proportional to its magnitude.

To achieve this, we compute the proportional translation of the image with respect to its size (i.e $\frac{\boldsymbol{v}}{L}$, where $\boldsymbol{v}$ is the translation vector applied to the image of length L). The proportional translation is then multiplied by the magnitude of the latent space, defined as $L_z = \max(\boldsymbol{z}_i) - \min(\boldsymbol{z}_i)$. In summary, the resulting translation whose inverse is applied to $\boldsymbol{z}_i$ is computed as follows:

$$\hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{L} L_z \tag{4}$$

Next, we evaluate this choice of normalization. This is done by evaluating the feature representation with and without our proposed normalization in the same manner as in Sec 4.3 in the main paper. Fig. 1 compares performance of PeCLR with and without translation normalization, as well as SimCLR. We observe that the error of applying direct translation, which omits the normalization scheme results in high errors, performing worse than SimCLR. However, using normalization leads to the best representation, outperforming both SimCLR and direct translation. This quantitatively motivates the use of our normalization procedure.

3. Training details

Here, we give more details on the training procedure of PeCLR. **Self-supervised pre-training** is performed for 100 epochs, which is empirically determined to perform best. Following [3], we use ADAM wrapped with LARS and a batch size of 2048. In order to fit the model on a RTX 2080 Ti, we accumulate gradients across smaller batches before back-propagating and use mixed precision for training. Learning rate is set to lr$= \sqrt{\text{batch size}} * 1e$-4, where a linear warmup is performed for the first 10 epochs. Proceeding that, we use cosine annealing for the remainder of training. While pre-training with multiple datasets, we perform weighted sampling so that a batch consisted of roughly equal amount of samples of each dataset. For PeCLR, we augment the image samples using rotations $r \in [-45, 45]$, translation $t \in [-15, 15]^2$ and scaling $s \in [0.6, 2.0]$. We pick these ranges empirically and find them to perform best. Increasing these ranges degrade performance and sometimes lead to stability issues. As appearance transformation, we applied color jitter via adjust hue, saturation and brightness. The former two were scaled by a factor $s \in [0.01, 1.0]$ whereas for brightness, we sample scaling factor $s \in [0.5, 1.0]$, bias $b \in [5, 20]$ and compute $av_{\text{brightness}} + b$, where $v_{\text{brightness}}$ is the brightness value.
Supervised fine-tuning is performed for 100 epochs. The adam optimizer with a learning rate of $5e$-4 is used in conjunction with cosine annealing. The batch size is set to 128. Data augmentation is employed, using rotations $r \in [-90, 90]$, translation $t \in [-20, 20]^2$ and scaling $s \in [0.7, 1.3]$.

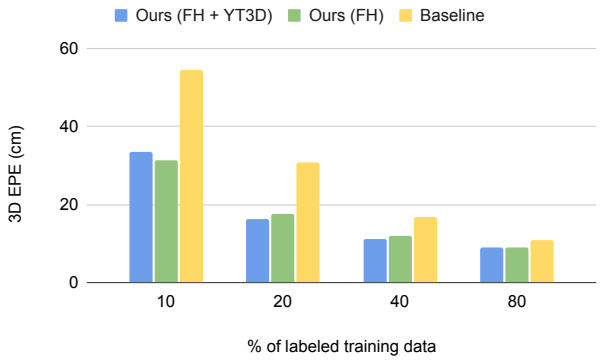


Figure 2: Semi-supervised performance on FH using RN50. We observe that by pre-training with PeCLR we achieve greater accuracy in contrast to only training supervised. However, the accuracy improvement between using FH and FH+YT3D is smaller as compared to using RN50.

4. Semi-supervised learning: RN50

We conduct the semi-supervised experiment in Sec. 4.4 in the main paper with RN152. In this section, we experiment with RN50 in the same setting as in Sec. 4.3 to show the benefit of using deeper networks.

Following the same experiment steps in Sec. 4.4, we first pre-train our RN50 encoder self-supervised on FH with PeCLR. Next, the encoder is fine-tuned supervised on varying amounts of labeled data from FH. We term the resulting model M_{FH}. We compare our proposed pre-training strategy against a baseline method M_b, which is trained solely supervised on the labeled data of FH, excluding the pre-training step. Lastly, a third model is trained to demonstrate the advantage of self-supervised representation learning with large training data, pre-trained on both FH and YT3D, named $M_{FH+YT3D}$.

Fig. 2 shows the absolute 3D EPE for models across all settings. We observe both PeCLR models $M_{FH}, M_{FH+YT3D}$ outperform the baseline model M_b across different labeling percentages. However, the improvement of $M_{FH+YT3D}$ over M_{FH} is lessened when using the RN50 in comparison to the RN152 model. For example, in the 20% labeled setting, by using FH and YT3D for pre-training the RN152 model can gain an improvement of 20% in 3D EPE with respect M_{FH} . On the other hand, the inclusion of additional data lead to an improvement of 8.6% for RN50. This result is consistent with [3], which also shows increased performance for larger models.

HO-3D				
Method	3D PA-EPE (cm)↓	PA-AUC ↑	3D EPE (cm)↓	AUC ↑
Hasson et al. [6]	3.18	0.46	3.27	0.44
Hampali et al. [5]	3.04	0.49	8.42	0.27
Supervised	1.33	0.74	2.85	0.50
PeCLR (**ours**)	**1.09**	**0.78**	**2.26**	**0.58**

Table 1: **HO3D evaluation.** We pre-train a RN50 using PeCLR on YT3D and FH and then fine-tuned supervised on FH. We compare with a model which is solely trained on HO3D (supervised) and note an improvement of 18% in 3D PA-EPE. Performance of [5, 6] acquired from [4].

Dexter+Object	AUC ↑
Mueller (2018)* [8]	0.48
Spurr (2018) [10]	0.51
Zimmermann (2018) [12]	0.57
Baek (2019)* [1]	0.61
Iqbal (2018)* [7]	0.67
Boukhayma (2019) [2]	0.76
Zhang (2019) [11]	**0.82**
Spurr (2020) [9]	**0.82**
Supervised	0.77
PeCLR (Ours)	0.81

Table 2: **Comparison with related work.** We adapt the table from [9] as it is the most comprehensive comparison with related work. *These works report unaligned results.

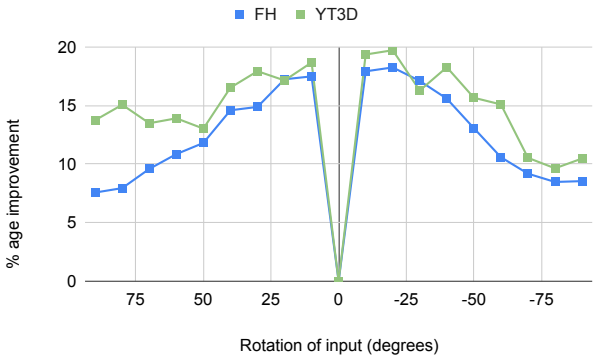


Figure 3: Percentage of improvement for rotational equivariance. Each point denotes the improvement of PeCLR over SimCLR for rotational equivariance, as measured for 2D EPE. We see that across all sampled rotations, PeCLR leads to increased equivariance on both the dataset the model was fine-tuned on (FH) as well as pre-trained (YT3D).

5. Further results on other datasets

Here we fine-tune models on HO-3D and Dexter+Object. We compare both training solely supervised (our baseline) with using PeCLR pre-training. To provide a better overview, we compare with other related work. We first investigate the results on HO-3D shown in Tab. 1. The baseline network is solely trained supervised on HO-3D, whereas PeCLR is pre-trained self-supervised on FH and YT3D and fine-tuned supervised on HO-3D. The results are as reported by the online submission system. We report the aligned and unaligned 3D EPE / AUC. On this dataset, we see that our baseline already outperforms related work. PeCLR is capable of pushing the performance even further, yielding an improvement of 18% in aligned EPE. Similar improvements can be found for all other metrics. This demonstrates that PeCLR yields improvement even if the pre-training dataset contains a domain shift with respect to the target dataset.

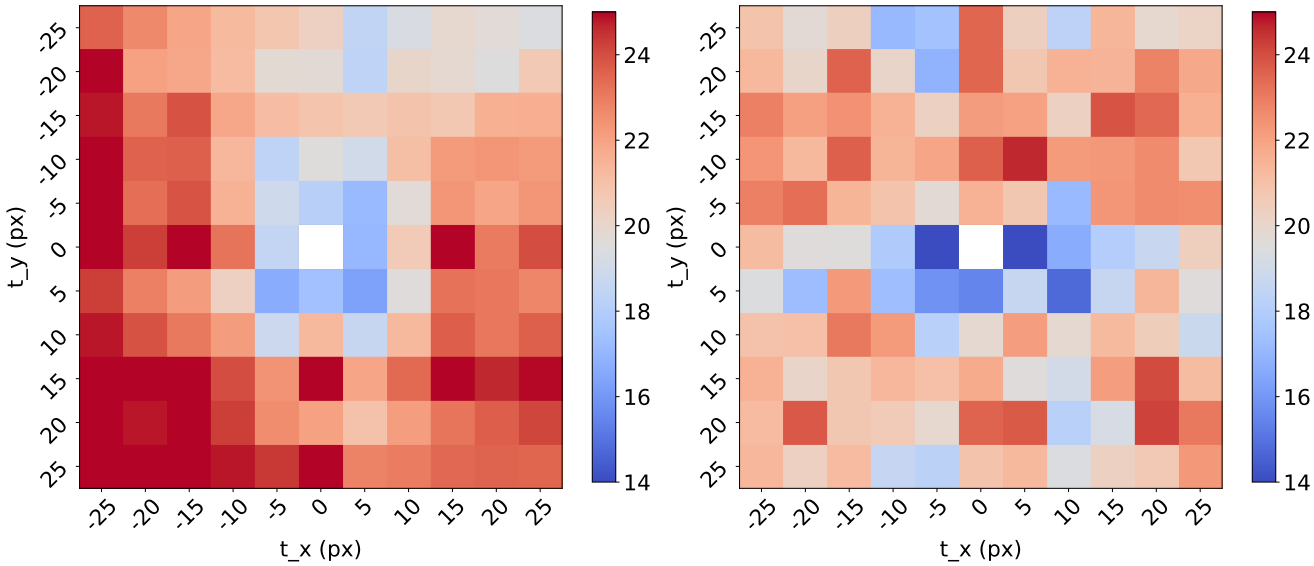
For Dexter+Object, we use the same network as in Sec. 4.5 in the main paper. Tab. 2 reports the aligned AUC for Dexter+Object dataset. Note that this dataset consists of completely unseen data. We adapt the table from [9] as it compares a wide range of works. We observe that the baseline network struggles to reach good performance (0.77 AUC). However, PeCLR yields improvements of 4.9%, almost reaching parity with state-of-the-art (0.81 AUC). This experiment indicates that PeCLR results in good cross-domain performance.

6. Inspecting equivariance of PeCLR and SimCLR

We investigate the equivariance of the resulting model after fine-tuning for both PeCLR and SimCLR. We quantify equivariance by measuring deviations from Eq. 2. Specifically, we report:

$$\mathcal{L}_{equiv}(\boldsymbol{I}^n) = ||t_i^g f(\boldsymbol{I}^n) - f(t_i^g(\boldsymbol{I}^n))||_2. \quad (5)$$

We investigate the rotation and translation augmentations since they are affected by PeCLR. To quantify the difference



(a) FreiHand

(b) YouTube3DHands

Figure 4: Percentage of improvement for translational equivariance. Each point denotes the improvement of PeCLR over SimCLR for translational equivariance, as measured for 2D EPE. We see that across all sampled translation on the grid, PeCLR leads to increased equivariance on both the dataset the model was fine-tuned on (FH, Fig. 4a) as well as pre-trained (YT3D, Fig. 4b).

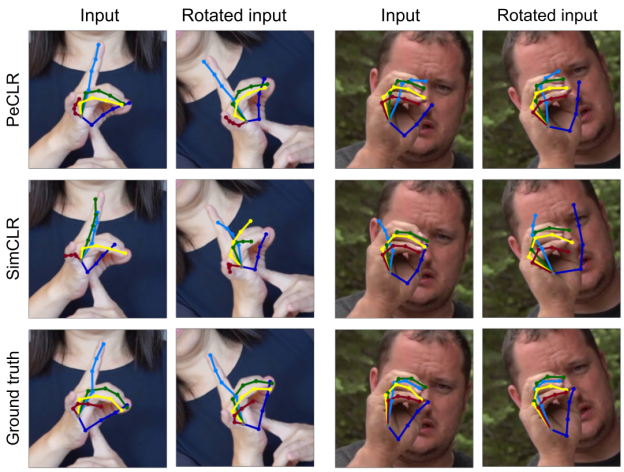


Figure 5: Qualitative samples of SimCLR and PeCLR pre-trained models on YouTube3DHands.

in performance between PeCLR and SimCLR, we visualize the following measure of improvement:

$$\mathcal{L}_{improv}(I^n) = \frac{\mathcal{L}_{equiv}^{SimCLR}(I^n) - \mathcal{L}_{equiv}^{PeCLR}(I^n)}{\mathcal{L}_{equiv}^{SimCLR}(I^n)} \quad (6)$$

This measure allows quantifying improvement relative to the scale of the error. For a given augmentation, we sample points equidistantly on their respective parameter ranges. For rotation we sample points equidistantly in the range $[-80°, 80°]$. For translation, we set the ranges at $[-25, 25]^2$. Each point is evaluated on the whole dataset. Here we evaluate on both YT3D and FH. Both models have been pre-trained self-supervised on both datasets and ine-tuned supervised on FH. We first visualize the results for the rotation augmentation as shown in Fig. 3. For both datasets, we see that $\mathcal{L}_{improv}$ is positive for the entire range tested, indicating that PeCLR performs better on equivariance tasks. The amount of improvement declines as we enter more extreme ranges. The same trend can be observe for both the dataset the models have been fine-tuned on (FH) as well as only pre-trained (YT3D). These results are supported by qualitative analysis, as can be seen in Fig. 5.

Fig. 4 shows the effect of translation on equivariance for both models. Similar to rotation, we observe overall improvement of PeCLR over SimCLR across all ranges sampled, as characterized by $\mathcal{L}_{improv}$ when more extreme translation is applied.

This experiment demonstrates that the equivariance property holds even after fine-tuning the network.

7. Qualitative results

Here we demonstrate further qualitative results on FH and YT3D. Furthermore, Fig. 6 and Fig. 7 visualize predictions on HO-3D and D+O from the models described in Sec. 5.

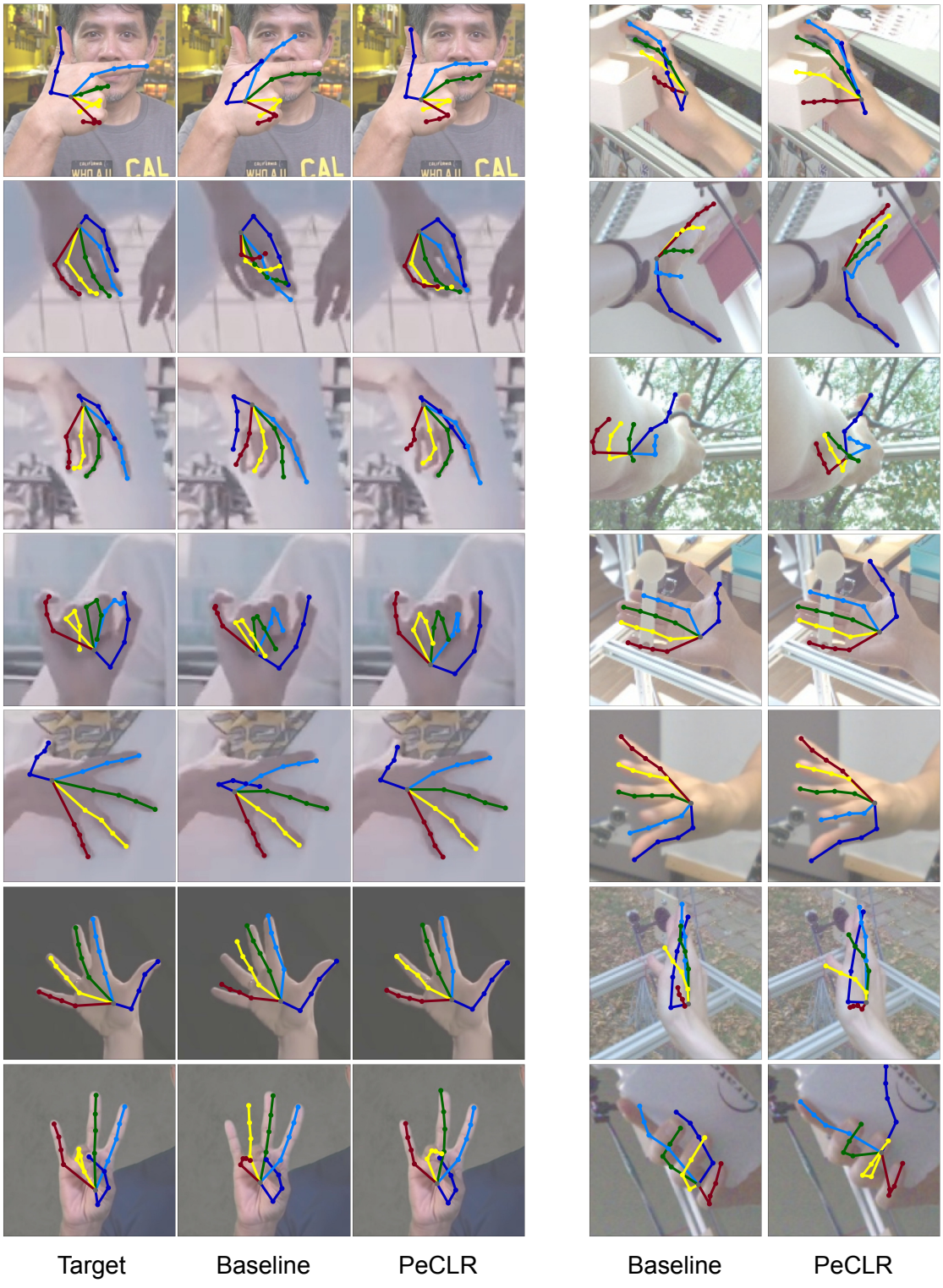


Figure 6: Predictions are shown on the test sets of YT3D (left) and FH (right) without (Baseline) or with PeCLR pre-training. Note that the ground truth of the test set is not publicly available for FH, thus we only visualize the predictions.

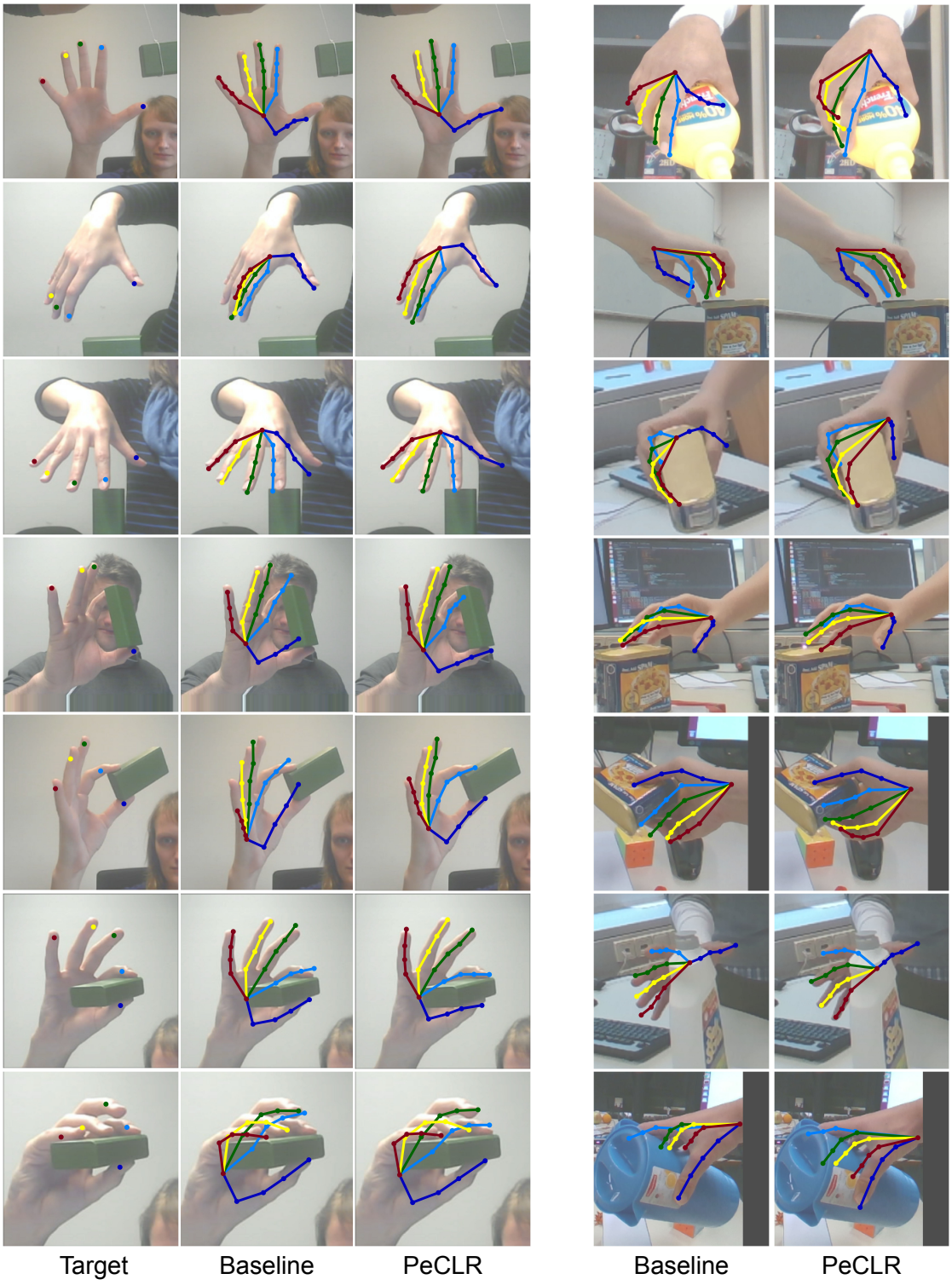


Figure 7: Predictions are shown on the test sets of D+O (left) and HO3D (right) without (Baseline) or with PeCLR pre-training. Note that the ground truth of the test set is not publicly available for HO3D, thus we only visualize the predictions.

References

[1] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019. 3

[2] Adnane Boukhayma, Rodrigo de Bem, and Philip H. S. Torr. 3d hand shape and pose from images in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019. 3

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, 2020. 2

[4] Shreyas Hampali. Icg - hand-object 3d pose annotation. https://www.tugraz.at/index.php?id=40231, 2021. 3

[5] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 2020. 3

[6] Yana Hasson, Gül Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019. 3

[7] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5d heatmap regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 3

[8] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular RGB. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018. 3

[9] Adrian Spurr, Umar Iqbal, Pavlo Molchanov, Otmar Hilliges, and Jan Kautz. Weakly supervised 3d hand pose estimation via biomechanical constraints, 2020. 3

[10] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. Cross-modal deep variational hand pose estimation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018. 3

[11] Xiong Zhang, Qiang Li, Hong Mo, Wenbo Zhang, and Wen Zheng. End-to-end hand mesh recovery from a monocular RGB image. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, 2019. 3

[12] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single RGB images. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017. 3