Supplemental Material:
PCLs: Geometry-aware Neural Reconstruction of 3D Pose with Perspective Crop Layers

Frank Yu[1] Mathieu Salzmann[2] Pascal Fua[2] Helge Rhodin[1]

[1]UBC, Vancouver, Canada
[2]EPFL, Lausanne, Switzerland
{frankyu, rhodin}@cs.ubc.ca

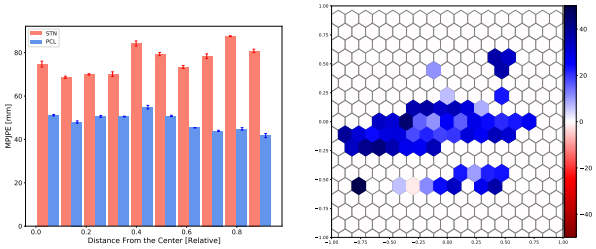

Figure 1: **Improvement of reconstruction error,** binned with respect to the image position. **Left:** MLP+RC suffers from perspective effects away from the center, while MLP+PCL effectively compensates these leading to improvements of approximately 50% . **Right:** The consistent difference of MLP+RC and MLP+PCL is also reflected over a 2D tiling, showing the average MPJPE error difference of cells with 10 or more frames on the validation set.

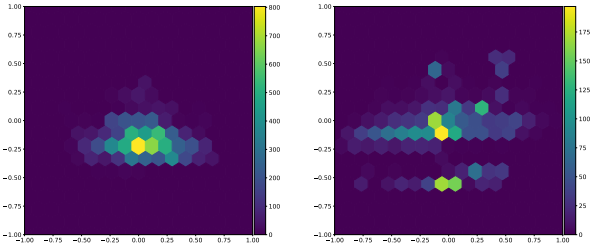

Figure 2: **Tiled 2D histogram of hip joint location** in normalized image coordinates over the test dataset for H3.6M (left) and MPI-INF-3DHP (right). The MPI-INF-3DHP set shows a wider and less regular distribution.

Appendix

A. MPI-INF-3DHP Additions

In this section, we repeat the detailed error distribution analysis done on the H3.6M dataset for the MPI-INF-3DHP dataset. Figure 1 depicts significant improvements from using PCL. The error by PCL is stable or even improving with the distance to the center, while the MLP+RC model degrades due to perspective effects. PCL even gains an improvement at the image center. This is unexpected on the first glance but can be explained with the MPI-INF-3DHP dataset using different cameras for training and testing. The MLP+RC model seemingly overfits to the perspectives seen during training while the automatic correction of PCL leads to better generalization irrespective of the image location.

In Figure 2, we can see the distribution of hip joints in the testing dataset for H3.6M and MPI-INF-3DHP. While H3.6M has most of the images around the center of the frame, MPI-INF-3DHP contains poses that are widely spread out across the image. This along with the fact that MPI-INF-3DHP uses a wider field-of-view camera explains the significant improvement we see from introducing PCL on MPI-INF-3DHP.

B. Ablation: Model Efficiency from PCL

To demonstrate the effectiveness of using PCL (taking perspective distortions into account), we reduce the dimension of the hidden state of our 2D-3D keypoint lifting model and report the performance on the validation set for H3.6M. From Table 1 we can see that even with about half the number of parameters as the baseline MLP we are able to capture a more precise reconstruction.

C. Ablation: Axis-based Rotation Experiment

To study the effects of using PCL as a post-process on exisiting trained models, we train the baseline 2D-3D keypoint lifting MLP and apply the PCL-defined rotation ma-

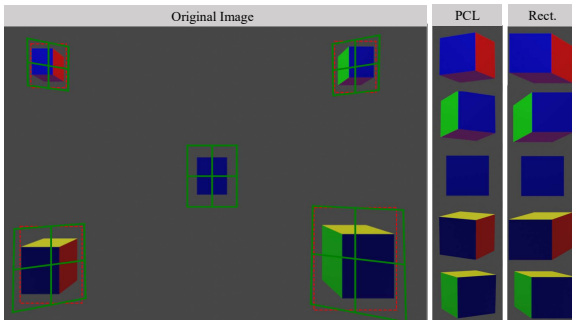

Figure 3: **Toy-Cube examples.** While cubes projected to the sides of an image appear distorted and look stretched when cropped, PCL undoes these perspective effects.

Model	Linear Size	# Parameters	MPJPE (mm)
MLP + RC	1024	4,296,755	48.4
MLP + PCL	**1024**	**4,296,755**	**43.8**
MLP + PCL	896	3,300,915	45.4
MLP + PCL	768	2,436,147	46.5
MLP + PCL	512	1,099,827	50.8

Table 1: Shown are the results on 2D-3D keypoint lifting on H3.6M while decreasing the model complexity of our PCL embedded model (MLP + PCL) while maintaining that of the baseline (MLP + RC). We can see here that despite having roughly 50% of the parameters of the baseline, the PCL-equipped model is still able to outperform the baseline.

trix on the predicted 3D pose. To this end, we experiment on MPI-INF-3DHP since that dataset contains more perspective distortions and will have more pronounced effects from adding the PCL-defined rotation. Along with this we also perform a "half-rotation" and "full-rotation" defined as rotation along only the x-axis and rotation about the x and y-axis respectively. Furthermore, we show results for when the 3D root is given and when scale is estimated from the 2D pose. From Table 2, we can see that when we compensate the baseline model with PCL-defined rotations it improves but still falls short of our model trained end-to-end with PCL, showing the importance of incorporating PCL during model training.

D. Ablation: Cube Dataset

The Cube Dataset contains images of a single coloured cube with edge length 0.5m at random locations and orientations within the frame. Figure 3 shows an example of the cube used in this dataset as well as demonstrating the perspective effects that occur when objects move away from the image center. To further demonstrate the effect that ignoring perspective effects has on 3D pose estimation we introduce a variation of this dataset in which the cube

MPI INF 3DHP	2D GT + 3D Root GT	2D GT
STN	69.4	74.1
STN + Half Rotation (x)	67.6	73.1
STN + Full Rotation (x, y)	66.3	70.0
PCL	**45.6**	**50.1**

Table 2: Shown are the results on 2D-3D keypoint lifting on MPI-INF-3DHP when applying half and full rotations. We can see that even after including information from PCL into the baseline model, it still falls short in terms compared to the model that is trained with PCL.

		Matching train-test set		Unseen test set
Model	Detector	centered	general	
CNN + STN	GT Loc. + GT Scale	6.9	13.0	13.3
CNN + PCL (Ours)	GT Loc. + GT Scale	6.9	**8.2**	**8.5**
CNN + STN	Trained End-to-End	5.9	11.2	-
CNN + PCL (Ours)	Trained End-to-End	5.9	**6.8**	-

Table 3: Shown are the reported MPJPE in millimeters for all tests conducted on the Cube Dataset and its variation. For this metric, lower values are better. We can see that our method produces more accurate results while at the same time generalizing better to unseen instances.

has a random orientation but is always placed at the center of the frame. We refer to this variation as the Centered Cube Dataset. We now train models on the Centered Cube Dataset with a central crop and analyze their generalization capabilities to crops from the general dataset with the position of the cube given. On this dataset, the baseline attains an MPJPE of 13mm and our PCL variant improves to 8.2mm. The improvement for end-to-end training is equivalent, with an 4.4mm improvement from 11.2 to 6.8mm. While this test is simplistic, the synthetic nature allows us to analyze the generalization capabilities of PCL by training on a version of the cube dataset where the cube is always centered in the training images and tested on the original version with general position, with the ground truth 2D crop location provided. PCL shows good generalization, outperforming the baseline by 36.1% on the unseen test set. Table 3 compares the performance of PCL and STN models on these two datasets. We also investigated the effect of illumination by switching from point lights to ambient illumination, which had negligible effect on the reconstruction quality.

E. Implementation Details

We normalize the 2D input poses and 3D output poses with their mean and standard deviation. On the input side, the mean and standard deviation are computed after the PCL layer and in the case of rectangular cropping (RC) after the crop and scaling operation. On the output side, the mean and standard deviation for PCL and the baselines are com-

puted in the pelvis-centered coordinates. We found it more effective to multiply the output by the computed standard deviation and adding the mean instead of doing the inverse operation on the label. This ensures that the network output has mean zero and unit standard deviation, which fares well with network layer initialization, while the loss operates on the scales of the original output space.

F. Derivations of the PCL Virtual Camera

Rotation Derivation. The rotation that maps from the virtual to the real camera, $\mathbf{R}_{\text{virt}\to\text{real}}$, stems from the definition of rotation matrices. We use the right-handed rule, i.e., counter-clockwise rotation for positive angles and a right-handed coordinate system with the y-axis pointing downwards, x-axis rightwards, and positive z pointing in camera direction. The definition of $\mathbf{R}_{\text{virt}\to\text{real}} = \mathbf{R}_y\mathbf{R}_x$ reads thereby

$$\mathbf{R}_y\mathbf{R}_x = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$
$$= \begin{bmatrix} \cos(\phi) & \sin(\phi)\sin(\theta) & \cos(\theta)\sin(\phi) \\ 0 & \cos(\theta) & -\sin(\theta) \\ -\sin(\phi) & \sin(\theta)\cos(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix}, \quad (1)$$

where ϕ and θ are, respectively, the vertical and horizontal rotations depicted in Figure 2 of the main document. The equation given in the main document follows from the following trigonometric relations,

$$\sin(\phi) = \frac{\mathbf{p}_x}{\sqrt{1+\mathbf{p}_x^2}}, \qquad \cos(\phi) = \frac{1}{\sqrt{1+\mathbf{p}_x^2}},$$
$$\sin(\theta) = \frac{-\mathbf{p}_y}{\sqrt{1+\mathbf{p}_x^2+\mathbf{p}_y^2}}, \qquad \cos(\theta) = \frac{\sqrt{1+\mathbf{p}_x^2}}{\sqrt{1+\mathbf{p}_x^2+\mathbf{p}_y^2}}, \quad (2)$$

where $\mathbf{p}$ is the point on the original image plane to which the virtual camera is rotated.

Virtual Focal Length Selection The focal length of the virtual camera defines the zoom level of the PCL crop. It controls the crop size in the original images. Therefore, it needs to be set individually for every crop target, depending on its desired size and position in the image. In the following, we explain the derivation of the three options we propose. Figure 4 shows example crops of each method and their tightness of fit.

A. By setting $\mathbf{h}^{\text{virt}}$ to $\mathbf{f}$, the camera is only rotated, without any change in zoom. A crop is obtained by scaling with factor $\mathbf{s}$, that means, $\mathbf{f} = \frac{\mathbf{h}}{\mathbf{s}}$. Figure 4, second row, shows that this simple choice leads to inconsistent crop sizes. The object appears smaller the further away it is.

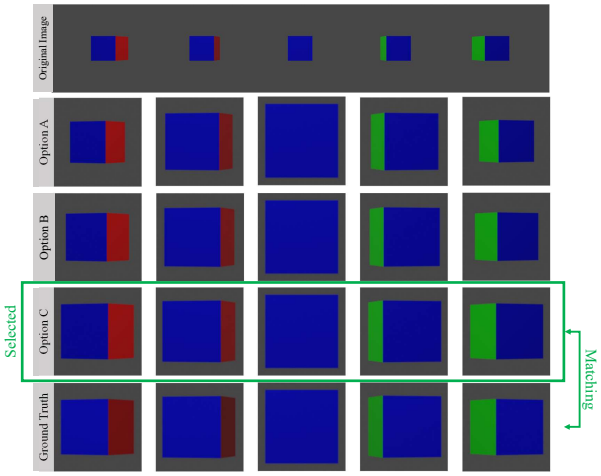

Figure 4: **Effect of virtual camera focal length.** The proposed options for setting the virtual focal length scale differently with respect to the image position. When given the pixel cube width as input (as a factor of the entire image resolution), only option C maintains the desired margin of 10 % between cube and crop boundary. The ground truth is a cube placed at the center of the screen rotated by the same angle that the virtual camera is rotated by. Remaining differences in color stem from the position-dependent illumination effects.

B. By multiplying the virtual length with $\|\mathbf{p}\|$, the distance of the crop target position on the image plane to the camera center, this distance-related effect is compensated. However, as the third row in Figure 4 shows, this match is not perfect as it does not account for the foreshortening effect when projecting from the original image plane onto the virtual one.

C. Our final choice models foreshortening with $\mathbf{h}_x^{\text{virt}} = \mathbf{f}_x\|\mathbf{p}\|\sqrt{\mathbf{p}_x^2+1}$ and $\mathbf{h}_y^{\text{virt}} = \mathbf{f}_y\frac{\|\mathbf{p}\|^2}{\sqrt{\mathbf{p}_x^2+1}}$. It is derived as follows.

Derivation of Option C. Let $\mathbf{p} = (x, y, z)^\top$ be the target crop position on the image plane, a 3D position. By construction, $\mathbf{p}$ will be at the image center. Therefore, projecting the infinitesimal motion offset $\mathbf{p} + (\delta x, \delta y, 0)^\top$ and comparing the ratio of the offset in the original and projection yields the desired scale estimate. Formally, we write

$$(u, v, 1)^T = \mathbf{P}\left(\mathbf{p} + (\delta x, \delta y, 0)^\top\right), \quad (3)$$

where $\mathbf{P}$ projects points in the original coordinate system to the virtual one, as defined in the main document. For the sake of simpler equations we do computations in camera coordinates with the origin at the image center and the focal

length $\mathbf{f} = 1$. In this case, $\mathbf{P} = \mathbf{R}_{\text{virt}\to\text{real}}^{-1}$. Using the definition of $\mathbf{R}_{\text{virt}\to\text{real}}$ above, the identity $\mathbf{R}_{\text{virt}\to\text{real}}^{-1} = \mathbf{R}_{\text{virt}\to\text{real}}^{\top}$, and computing the partial derivatives with respect to δx and δy at $\delta x = \delta y = 0$ we obtain

$$\frac{\partial (u, v, 1)^T}{\partial u}\Big|_{\delta u = 0} = \frac{1}{\sqrt{(1 + x^2)}\|\mathbf{p}\|} \tag{4}$$

and

$$\frac{\partial (u, v, 1)^T}{\partial v}\Big|_{\delta v = 0} = \frac{\sqrt{(1 + x^2)}}{\|\mathbf{p}\|^2}. \tag{5}$$

These equations compute the horizontal and vertical pixel scale ratio between the original and virtual image at $\mathbf{p}$. To maintain the scale, the focal length must be set to the inverse of this scaling factor, which is Option C for f^{virt} that also incorporates the original focal length and the desired crop scale.

Note that the equations for $\mathbf{h}_x^{\text{virt}}$ and $\mathbf{h}_y^{\text{virt}}$ are not equivalent with x and y exchanged because the x and y axes in the virtual camera do not, in general position, project to perpendicular lines in the original view. Figure 1 of the main paper provides examples of this perspective effect. In our definition, the up-direction is kept fixed and the horizontal axis is rotated, therefore, behaving differently in the slant compensation. An equivalent formulation could be derived with the horizontal axis fixed and the vertical axis rotated.

Maintaining the original aspect ratio To enable computations with a different focal length in the x and y directions, which models the case of non-square pixels, the above notation was performed individually for the horizontal and vertical directions. This, however, can lead to stretched crops if different scales are predicted for $\mathbf{s}$ in horizontal and vertical directions. To maintain the original aspect ratio, we set the focal length to the minimum of the axis-specific lengths. This leads to a crop that is the same or larger than the original one with mismatching aspect ratio, thereby strictly containing the object of interest.