Appendix

A. Implementation Details

A.1. $\mathbf{\Phi}_{\text{align}}$ Architecture

We use a small U-Net architecture [66] to represent $\mathbf{\Phi}_{\text{align}}$ consisting of four downscaling fully convolutional blocks and four upscaling fully convolutional blocks. Output of downscaling blocks is concatenated to the upscaling blocks, as is typical in U-Nets. The size and parameter details of each of the blocks is provided in Table 6.

Output of the final layer has two channels predicting the x and y canonical space coordinates for each pixel. The canonical grid is a learned embedding of dimension $256 \times 256 \times 4$. We fixed the learning rate for the network to 0.001 and train the entire network end to end for $20,000$ iterations with a batch size of 20 on a single GPU.

Table 6: U-Net architecture for $\mathbf{\Phi}_{\text{align}}$ - It is a fully convolutional architecture, consisting of an Input block, Output block, four Up blocks, and four Down blocks. Each block consists of Up, Down, and DoubleConv Layers as shown below. Each DoubleConv, Up, and Down blocks is parameterized by number of input and output channels *i.e.*, (C_{in}, C_{out}). Each Conv2D is represented by $(C_{in}, C_{out}, \text{kernel}, \text{stride}, \text{pad})$. BN stands for batch normalization layer and has C_{out} parameters. ReLU are Rectified Linear Units without any parameters.

Blocks	Layers	Output Size
Input	DoubleConv $(3, 32)$	$32 \times 128 \times 128$
Down-1	Down $(32, 64)$	$64 \times 64 \times 64$
Down-2	Down $(64, 128)$	$128 \times 32 \times 32$
Down-3	Down $(128, 256)$	$256 \times 16 \times 16$
Down-4	Down $(256, 512)$	$512 \times 8 \times 8$
Up-1	Up $(512, 128)$	$256 \times 16 \times 16$
Up-2	Up $(256, 64)$	$128 \times 32 \times 32$
Up-3	Up $(128, 32)$	$64 \times 64 \times 64$
Up-4	Up $(64, 32)$	$32 \times 128 \times 128$
Output	DoubleConv $(32, 4)$	$4 \times 256 \times 256$
DoubleConv	Conv2D$(C_{in}, C_{out}, 3, 1, 1)$ BN(C_{out}) ReLU Conv2D$(C_{in}, C_{out}, 3, 1, 1)$ BN(C_{out}) ReLU	
Down	MaxPool2D(2) DoubleConv(C_{in}, C_{out})	
Up	BilinearUpsample(2) DoubleConv(C_{in}, C_{out})	

A.2. Canonical grid $\mathbf{G}$

The canonical grid $\mathbf{G}$ consists of a simple $256 \times 256 \times 4$ feature grid which is learned during the training with the same learning rate and optimizer as the alignment network $\mathbf{\Phi}_{\text{align}}$. Each location in $\mathbf{G}$ stores an (r, g, b, α) value which corresponds to colors (r, g, b) and a probability α that this location corresponds to a foreground pixel in the image.

A.3. Loss terms

Recall that our overall objective function comprises 5 different loss terms of which $\mathcal{L}_{\text{KP}}$, $\mathcal{L}_{\text{Equi}}$, and $\mathcal{L}_{\text{TV}}$ are applied to canonical space coordinates $\mathbf{C}$ and update only the parameters of alignment network $\mathbf{\Phi}_{\text{align}}$ (and not the canonical grid $\mathbf{G}$). $\mathcal{L}_{\text{Recon}}$ and $\mathcal{L}_{\text{Parts}}$ can backpropagate gradients to both the alignment network and the canonical grid. In all our experiments, on all 4 datasets and their respective categories, we use the same set of weight coefficients (except for the ablation study in Section 4, where we make the coefficients zero one at a time). We set of the coefficients for different loss terms as following: $\lambda_{\text{KP}} = 10$, $\lambda_{\text{Equi}} = 1$, $\lambda_{\text{TV}} = 9000$, $\lambda_{\text{Recon}} = 1$, and $\lambda_{\text{Parts}} = 10$. We observed that our framework is robust to the choice of hyperparameters. We can further increase the PCK performance by setting per-category hyperparameters, however, per-category (or per-collection) tuning is not ideal for scaling the model to a large number of image collections. Hence, we choose to report all our numbers with a fixed set of hyperparameters.

A.4. Choice of SSL for pseudo-correspondences.

In our experiments, we obtain initial set of pseudo-correspondences by finding mutual nearest neighbors from frozen DINO (ViT-S/8) network. Note that DINO is not trained or fine-tuned in our experiments. Our alignment network $\mathbf{\Phi}_{\text{align}}$, which is much smaller than DINO, is trained from scratch. This is also in contrast with other weakly supervised techniques such as PMD which uses ResNet-101 / VGG-16 ($>$ 40M params). We observe that performance of our framework can be improved further by using better pseudo-correspondences. Tab. 7 shows ASIC results when obtaining pseudo-correspondences from 3 different ViT architectures.

Table 7: Pseudo-correspondences Ablation on CUB-001

Architecture	# params	ImageNet Top-1 (Accuracy)	DVD PCK@0.1	ASIC (ours) PCK@0.1
ViT-S/16	21M	77.0	59.8	**63.7**
ViT-B/8	85M	80.1	66.4	**74.9**
ViT-S/8 (paper)	21M	79.7	66.8	**71.8**

B. Visualizing the Canonical Space

Recall that in Section 4 of the paper, we showed the canonical space mapping for various datasets learned by our model. Here we provide further details of the canonical space mapping. Specifically we first show the region in 2D space where each point in the image is getting mapped in Appendix B.1. Next we show the RGB grid that is learned by our model.

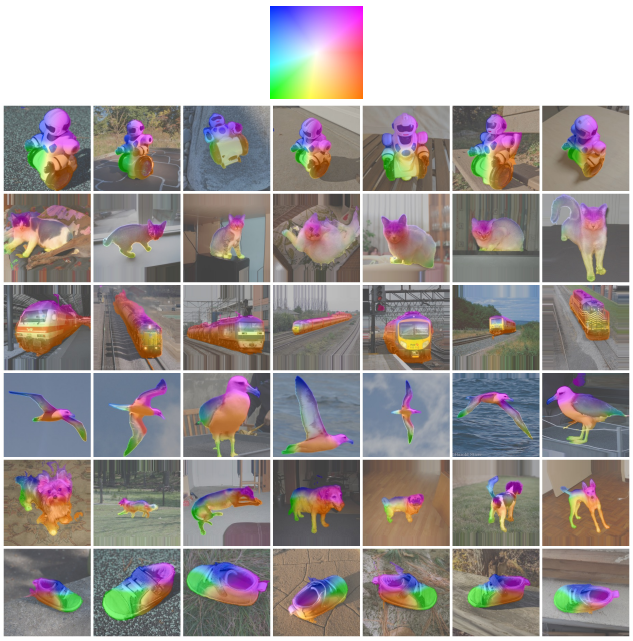

Figure 7: **Colormap for canonical space visualization.** We use the colormap shown in the top row to represent the canonical space. Based on the canonical space coordinates predicted by our model for each pixel, we copy (or more precisely splat) the colors from the canonical space colormap to the original image. Each row shows the mapping learned by the model for different datasets.

B.1. With colormap

First, we reproduce the results from Section 4 here, along with the colormap of canonical space used to visualize them in Figure 7. Note that we train a different model for each dataset. The figure shows that the semantically similar parts of objects get mapped to nearby location in the canonical space. Our model is able to learn a smooth mapping for each object.

B.2. With learned RGB Grid

Our method also learns an RGB grid. Figure 8 shows the grid learned for 4 different datasets. We observe that while our grid is not interpretable, there are distinct patterns that emerge for each dataset. Specifically, one can observe wheel-like shapes in the bicycle grid, and a cube in the train canonical grid. We attribute the weak interpretability of the learned grid to the large variability in the challenging in-the-wild images, where images may consist of different instances of an object category in very different poses, articulations, shapes, textures, background, and lighting. The collections we used are also very small (5-25 images). Further while our alignment network ensures that the pseudo-correspondences across images land at the same location in the canonical space, nearby points within the same image can still map to far away locations in the canonical space. Making the grid more interpretable could be useful for better understanding of the model's capabilities and limitations. For instance, in the case of GANgealing [61], training a GAN on a large dataset of cats (∼1.5M images), they are able to learn a canonical atlas which looks like face of a cat. This allows them to use canonical atlas as the template for image editing and edit propagation templates (although, a limitation of this approach is that it doesn't allow editing any parts other than the face of a cat).

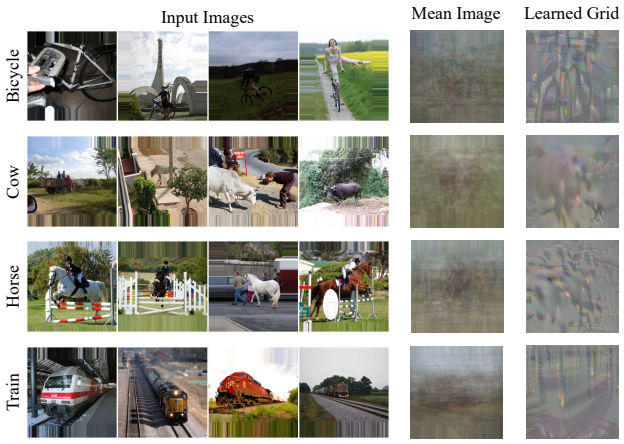

Figure 8: Sample images from the dataset in the left four columns, followed by the mean image of the dataset. The last column shows the joint canonical grid learned by the model.

C. Results on different k-values and all datasets for k-CyPCK

We share the results for of $k \in \{2, 3, 4\}$ and plot k-CyPCK for all the datasets (with groundtruth keypoint annotations) we considered in our experiments. Figures 9 to 11 show the comparison between our method and DVD. Note that DVD is also referred to as DINO + NN (where NN stands for nearest neighbors) in the main paper to clarify the strategy used to find the correspondences. Our method consistently outperforms the baseline, at both small and large values of α_{bbox} (which corresponds to the coarse and fine precision or accuracy of the transfer).

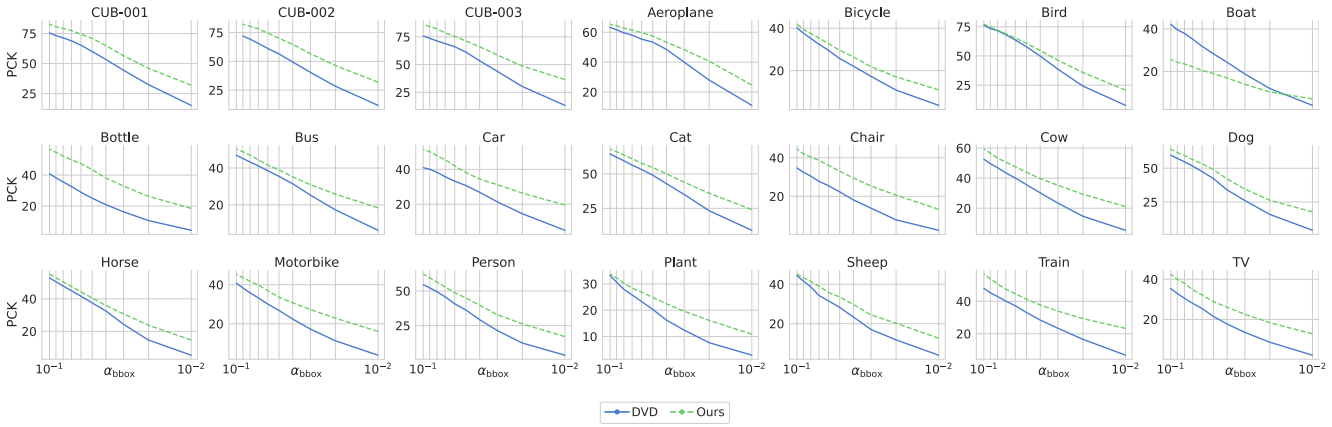

Figure 9: **2**-CyPCK for three CUB-200 categories and 18 SPair-71k categories (test split)

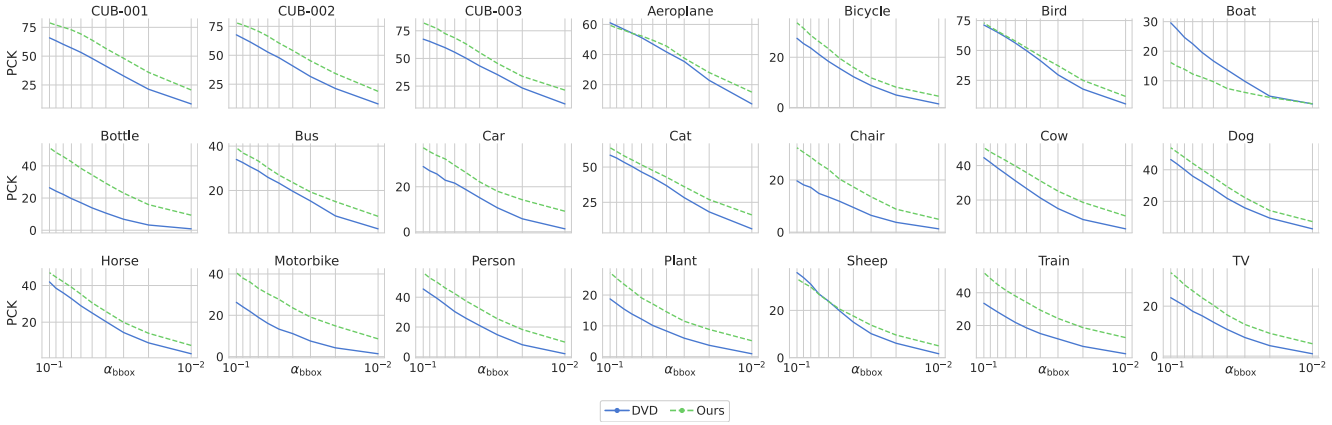

Figure 10: **3**-CyPCK for three CUB-200 categories and 18 SPair-71k categories (test split)

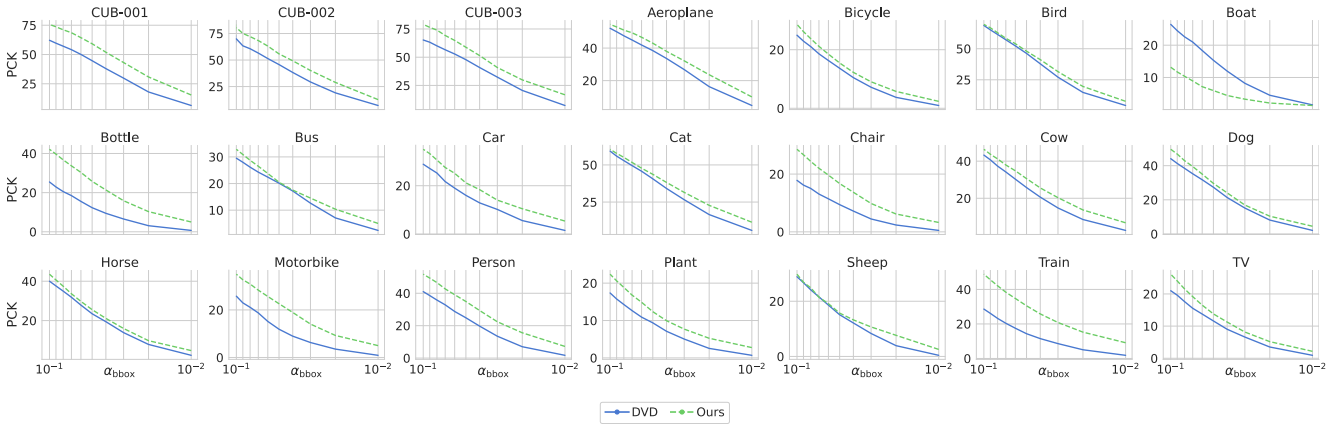

Figure 11: **4**-CyPCK for three CUB-200 categories and 18 SPair-71k categories (test split)