Supplementary material: Cross-Domain 3D Equivariant Image Embeddings

Carlos Esteves [*1] **Avneesh Sud** [2] **Zhengyi Luo** [1] **Kostas Daniilidis** [1] **Ameesh Makadia** [2]

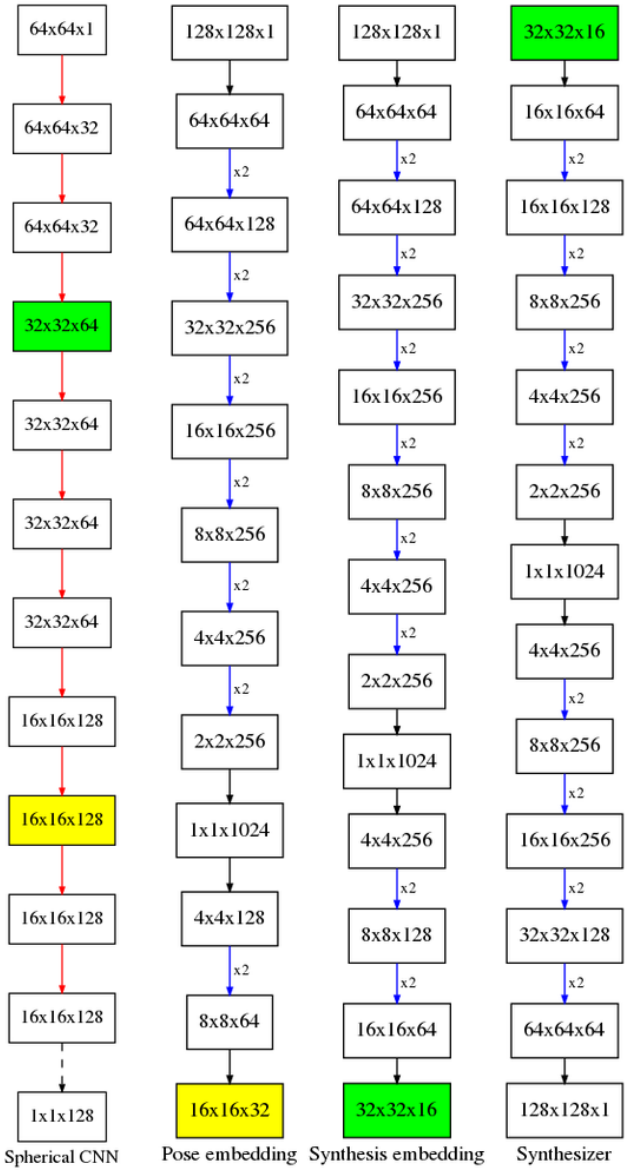

Figure 1. **Network architectures used in this work.** Rectangles indicate data dimensions (width x height x channels). *Red arrow:* spherical convolutional residual bottleneck layer; *dashed arrow:* global average pooling; *blue arrow:* residual bottleneck layer; *black arrow:* convolutional layer. Nodes with yellow and green backgrounds are the target embeddings for pose and synthesis, respectively.

1. Architecture details

1.1. Spherical CNN

We train a single 10-layer spherical CNN for object classification on ModelNet40 and use it to obtain the target embeddings for all experiments in this paper. The basic block is the spherical convolutional residual layer. The architecture is shown in figure 1, where the a cross-entropy loss over 40 classes is optimized. The network s is trained for 15 epochs with a batch size of 16, and ADAM optimizer with initial learning rate of 10^{-3} which is reduced to $2 * 10^{-4}$ and $4 * 10^{-5}$ at steps 5000 and 8500, respectively. Random anisotropic scaling is used as augmentation. It achieves 84.2% accuracy. Esteves et al. (2018) achieve 86.9% on the same task, but with a different architecture containing an extra branch to process surface normals; our inputs are only the ray lengths from the ray casting procedure.

1.2. Embedding network

Our embeddings are obtained with encoder-decoder residual networks. Given an input with dimensions $N \times N$, the encoding step contains one 7×7 convolutional layer followed by $\log_2 N - 1$ blocks of 2 residual layers, followed by a final convolutional layer that produce a 1D latent vector. The number of channels double at each residual block, starting at 64 and capped at 256. Downsampling is through strided convolutions.

The 1D encoding is upsampled using a convolutional layer followed by a sequence of residual blocks and a final 7×7 convolutional layer up to the desired resolution and number of channels, which is $16 \times 16 \times 32$ for the pose experiments, and $32 \times 32 \times 16$ for novel view synthesis. Upsampling is through transposed convolutions.

Note that our targets are spherical CNN features inside the residual bottlenecks, so the embeddings have 4 times fewer channels than the actual spherical CNN layer outputs. The image inputs are 128×128 and 1024 units are used in the 1D encoding. See figure 1 for the resolutions and number of channels per layer.

The embedding network f is trained to minimize a Huber loss for 200k steps with a batch size of 16, and ADAM optimizer with initial learning rate of $2 * 10^{-4}$ which is reduced

to $4 * 10^{-5}$ and 10^{-5} at steps 80k and 180k, respectively. Random anisotropic scaling of meshes prior rendering is used as augmentation.

1.3. Synthesis network

The synthesizer network g follows the same structure as the embedding, the difference being that the inputs are $32 \times 32 \times 16$ and the outputs 128×128. One question that arises is if the synthesizer should be trained with the target spherical CNN embeddings as inputs ($s(x)$) or from the embeddings obtained from single views by our network ($f(y)$). We found that the latter is slightly better.

The synthesis network is trained to minimize an L_2 loss for 200k steps with a batch size of 8, and ADAM optimizer with initial learning rate of $2 * 10^{-4}$ which is reduced to $4 * 10^{-5}$ and 10^{-5} at steps 80k and 180k, respectively. Random anisotropic scaling of meshes prior to rendering is used as augmentation.

2. Evaluation details

In this section, we include details of the training setup for competing approaches. Note that we still outperform these methods even when allowing more information in the form of oriented meshes, pose supervision and warm starting from pre-trained networks.

2.1. Regression

We utilize the same architecture as in the middle columns of figure 1 up to the 1024 dimensional bottleneck, followed by the pose network from Mahendran et al. (2017). We train for 200k steps, with a batch size of 16, and ADAM optimizer, with initial learning rate of $1 * 10^{-4}$, which is reduced to $5 * 10^{-5}$, $2 * 10{-}5$, and $8 * 10^{-6}$ at steps 40k, 75k and 125k respectively. The RMSE and geodesic loss scheduling similar to Mahendran et al. (2017) is used - RMSE loss is used for 100k steps, followed by geodesic loss. When training the 3DOF model, we found that the performance improves when warm starting from a network pre-trained on the 2DOF training set.

The original model is trained to regress a canonical pose. In our setting, where we render views from a mesh dataset, the meshes need to be aligned or have annotated poses. Since our method does not require aligned meshes, a more fair comparison would be to train the regression model on pairs of views where the regression target is the relative pose. We experimented with several variations of this approach and the performance was worse than the regression to a canonical orientation, so we only report results computing the relative pose from the regressed canonical orientations.

2.2. KeyPointNet

We utilize the authors' publicly available code and default parameters with minor modifications. The required changes are because Suwajanakorn et al. (2018) distribute the training, which allows a larger batch size of 256, while we train only on a single GPU with a batch size of 24. We found that with a smaller batch size the default orientation prediction annealing steps (30k-60k) prevents convergence; we changed it to 120k-150k and increased the number of steps from 200k to 300k to be able to reproduce (and slightly improve) the numbers reported in Suwajanakorn et al. (2018) (see table 1). We also modify their rendering code to generate the 2DOF and 3DOF datasets, as the original paper only considers a 2DOF hemisphere.

	airplane	car	chair
Our parameters	6.06	3.31	4.94
Original parameters	5.72	3.37	5.42

Table 1. Median angular error in degrees for instance based 2DOF hemisphere alignment on ShapeNet. Our parameter selection slightly outperforms the original results from Suwajanakorn et al. (2018).

3. Extra experiments

We evaluate image to mesh alignment on ShapeNet and relative pose estimation on ModelNet40. For completeness, we also include regression results to estimate error to a canonical pose. Table 3 shows the results for 3DOF alignment and Table 4 for 2DOF alignment for ModelNet40.

3.1. Image to mesh alignment

Although we focus on tasks where the inputs are 2D images, our method produces a common equivariant representation for images and meshes that can be used to align images to meshes. Table 2 shows the results.

		airplane	car	chair	sofa
2DOF	im-mesh	5.65	4.95	13.28	12.34
	im-im	6.24	4.73	12.10	10.80
3DOF	im-mesh	5.98	4.24	13.21	11.43
	im-im	7.27	4.59	12.30	9.66

Table 2. Image to mesh alignment experiment on ShapeNet. We show the category based median relative pose error in deg for image to image (*im-im*) and image to mesh (*im-mesh*).

3.2. ModelNet40 relative pose

Tables 3 and 4 show alignment results for ModelNet40.

		airplane	bed	chair	car	sofa	toilet
IB	Regr.	11.8	26.0	43.7	16.5	25.3	17.8
	Ours	**7.23**	**4.93**	**7.79**	**3.95**	**6.51**	**5.17**
CB	Regr.	12.9	29.9	52.5	15.2	34.5	17.8
	Ours	**8.81**	**8.55**	**15.3**	**5.12**	**11.0**	**10.9**

Table 3. Median angular error in degrees for instance (IB) and category-based (CB) 3DOF alignment on ModelNet40.

		airplane	bed	chair	car	sofa	toilet
IB	Regr.	6.29	12.7	25.5	6.84	12.5	9.76
	Ours	**3.33**	**4.46**	**7.07**	**4.12**	**4.52**	**4.88**
CB	Regr.	7.13	15.8	32.2	7.00	13.3	**10.4**
	Ours	**4.80**	**6.60**	**10.2**	**4.82**	**9.56**	10.8

Table 4. Median angular error in degrees for instance (IB) and category-based (CB) 2DOF alignment on ModelNet40.

3.3. Novel view synthesis

We show novel view synthesis results for other classes, including a failure case in 2.

3.4. Visualization

Figure 3 shows inputs, embedding channels, rotated embedding channels and outputs from synthesis. Figure 4 shows inputs, embedding channels, and alignment visualization. See animations in the supplementary material (`synthesis1.gif`, `synthesis2.gif`, `pose.gif`).

References

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. Learning SO(3) equivariant representations with spherical cnns. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Mahendran, S., Ali, H., and Vidal, R. 3d pose regression using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.

Suwajanakorn, S., Snavely, N., Tompson, J. J., and Norouzi, M. Discovery of latent 3d keypoints via end-to-end geometric reasoning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2063–2074, 2018.

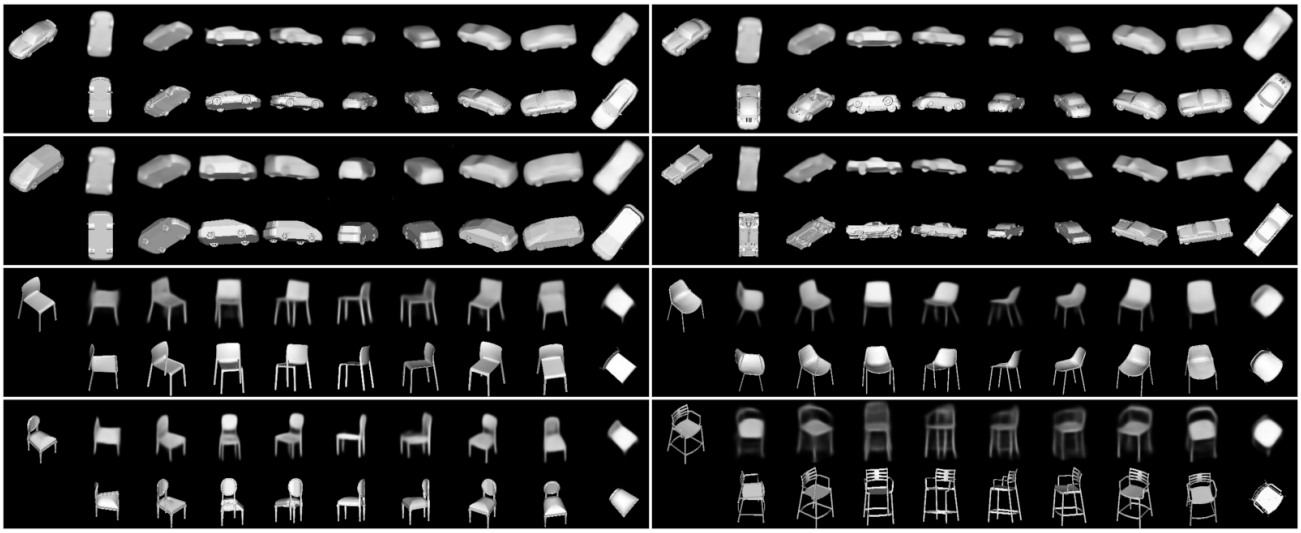

Figure 2. **More novel view synthesis results.** *Top-left:* inputs, which are 2D images from the test set. *Top row:* novel views generated using our method. *Bottom row:* ground truth views rendered from the original mesh. The bottom right shows a failure case due to a chair with uncommon appearance.

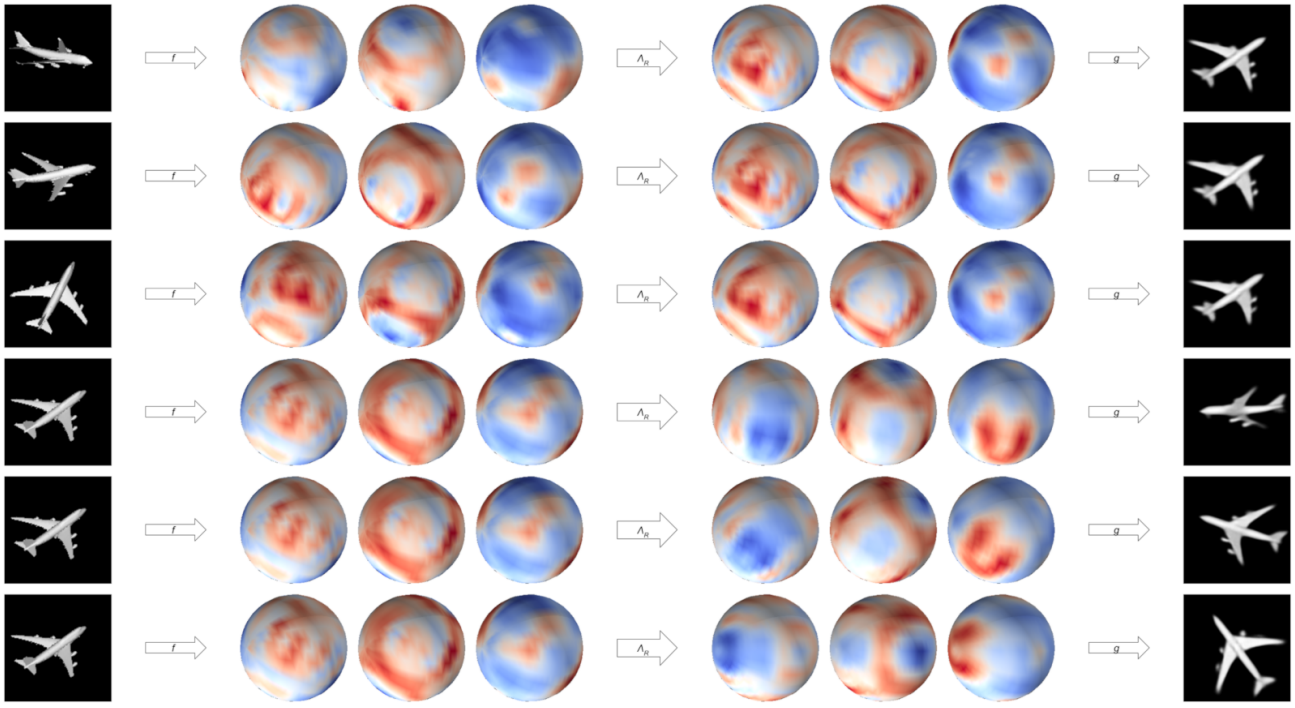

Figure 3. **Novel view synthesis visualization.** *Each row:* inputs, 3 embedding channels, rotated embedding channels, outputs. Top 3 rows show generation of a canonical view from arbitrary views and correspond to `synthesis1.gif`. Bottom 3 rows show generation of arbitrary views from a canonical view and correspond to `synthesis2.gif`.

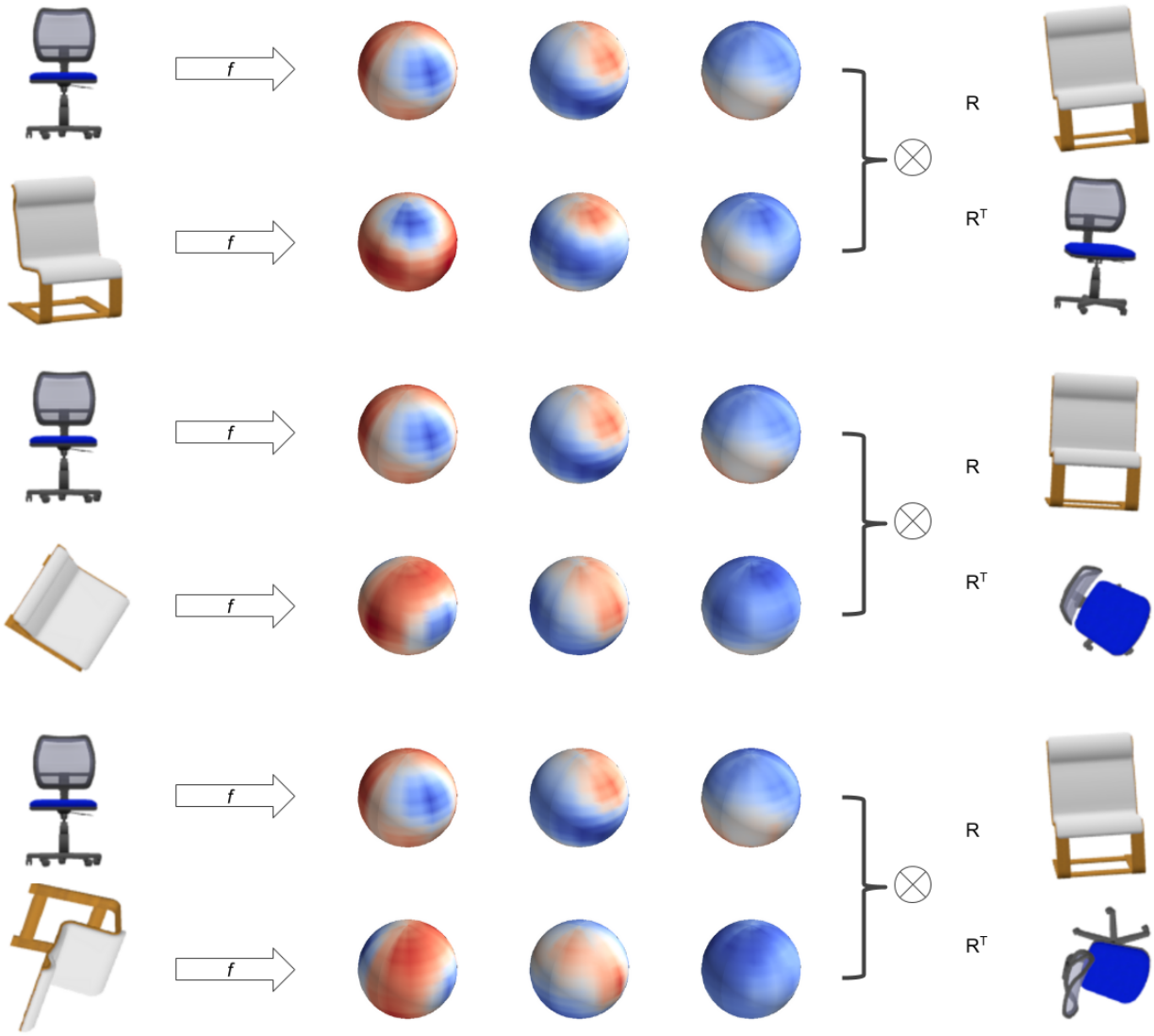

Figure 4. **Relative pose estimation visualization.** *Each block of two rows:* pair of inputs, 3 embedding channels per input, mesh 2 rotated into pose 1, and mesh 1 rotated into pose 2. We render from the ground truth meshes for visualization purposes only; our inputs are solely the 2D views and output is the relative pose. See `pose.gif` for an animation.