Supplemental Material for *Implicit-PDF: Non-Parametric Representation of Probability Distributions on the Rotation Manifold*

1. Additional IPDF predictions for objects from Pascal3D+

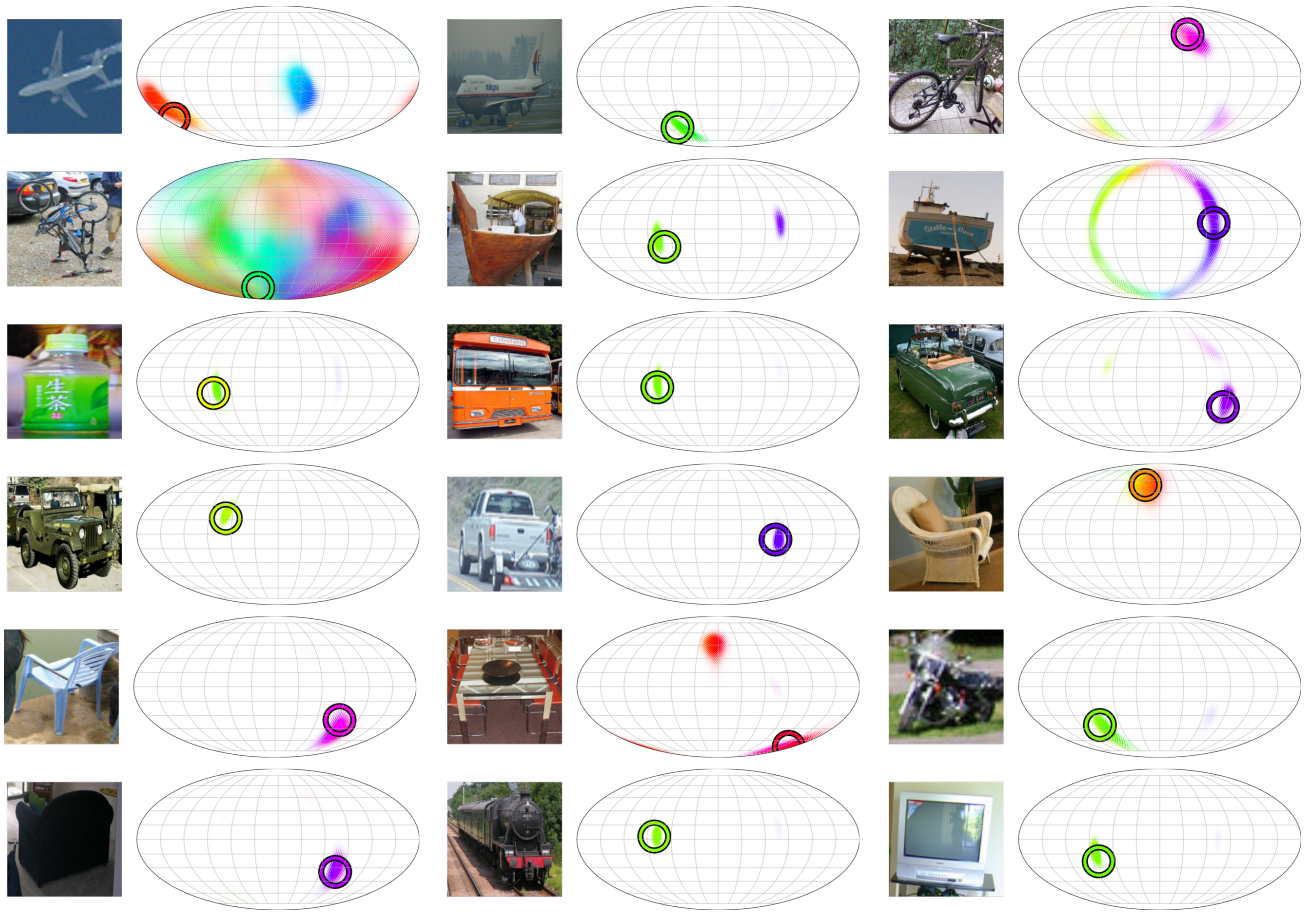

Figure S1. **Sample IPDF outputs on Pascal3D+ objects.** We visualize predictions by the IPDF model, trained on all twelve object categories, which yielded the results in Table 3 of the main text. The ground truth rotations are displayed as the colored open circles.

In Figure S1 we show sample predictions from IPDF trained on the objects in Pascal3D+. The network outputs much more information about the pose of the object in the image than can be expressed in a single estimate. Even in the examples where the distribution is unimodal, and the pose is relatively unambiguous, IPDF provides rich information about the uncertainty around the most likely pose. The expressivity of IPDF allows it to express category-level symmetries, which appear as multiple modes in the distributions above. The most stand-out example in Figure S1 is of the bicycle in the second row:

the pose estimate of IPDF is incredibly uncertain, yet still there is information in the exclusion of certain regions of $\mathbf{SO}(3)$ which have been 'ruled out'. The expressivity of IPDF allows an unprecedented level of information to be contained in the predicted pose distributions.

2. Extension of IPDF beyond $\mathrm{SO}(3)$

IPDF is not limited to probability distributions on $\mathbf{SO}(3)$, which nevertheless served as a challenging and practical testing ground for the method. With minor modifications, IPDF can be extended to the problem of pose with six degrees of freedom (6DOF): we append translation coordinates to the rotation query, and use $10\times$ more samples during training to adequately cover the full joint space. Normalizing the distributions is similarly straightforward, by querying over a product of Cartesian and HealPix-derived grids. Predicted distributions on modified images of SYMSOL are shown in Figure S2. For two renderings of a cone from identical orientation but different translations, only the predicted distribution over translation differs between the two images.

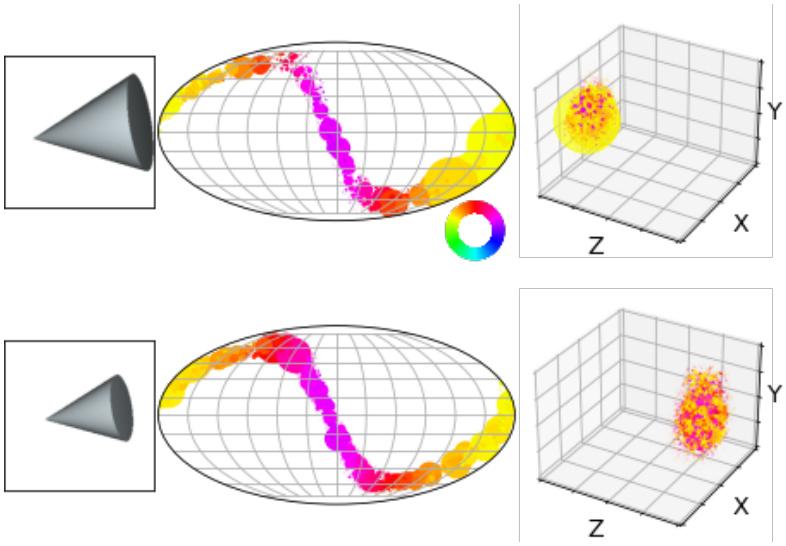

Figure S2. **Extension to 6DOF rotation+translation estimation.** We train IPDF on a modified SYMSOL I dataset, where the objects are also translated in space. Shown above are two images of a cone with the same orientation but shifted in space. We query the network over the full joint space of translations and rotations, and visualize the marginal distributions. Each point in rotation space has a corresponding point in translation space, and we color them the same to indicate as such. While uninformative in the above plots, this scheme of coloring allows nontrivial joint distributions to be expressed.

3. SYMSOL spread evaluation, compared to multimodal Bingham

Table S1. Spread estimation on SYMSOL. This metric evaluates how closely the probability mass is centered on any of the equivalent ground truths. For this reason, we can only evaluate it on SYMSOL I, where all ground truths are known at test time. Values are in degrees.

	cone	cyl.	tet.	cube	ico.
Deng et al.	10.1	15.2	16.7	40.7	29.5
Ours	1.4	1.4	4.6	4.0	8.4

We evaluate the spread metric on the SYMSOL I dataset, where the full set of ground truths is known at test time, for IPDF and the method of Deng et al. (2020). The results are shown in Table S1.

The metric values, in degrees, show how well the implicit method is able to home in on the ground truths. For the cone and cylinder, the spread of probability mass away from the continuous rotational symmetry has a typical scale of just over one degree.

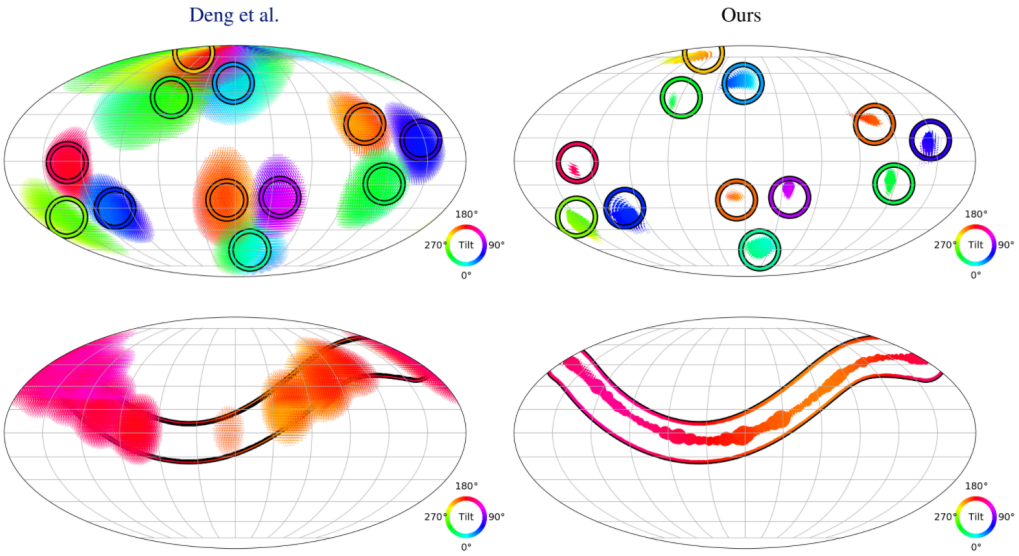

Figure S3. **Comparison of predicted distributions: tetrahedron and cone.** We show predicted pose distributions for a tetrahedron (top) and cone (bottom). Displayed on the left is the method of Deng et al. (2020), which outputs parameters for a mixture of Bingham distributions. The right side shows IPDF. The predicted distributions from the implicit method are much more densely concentrated around the ground truth, providing a visual grounding for the significant difference in the spread values of Table S1.

The predicted distributions in Figure S3 for a tetrahedron and cone visually ground the values of Table S1. Many of the individual unimodal Bingham components can be identified for the output distributions of Deng et al. (2020), highlighting the difficulty covering the great circle of ground truth rotations for the cone with only a limited number of unimodal distributions (bottom). The spread around the ground truths for both shapes is significantly larger and more diffuse than for IPDF, shown on the right.

4. Computational cost

We evaluate the computational cost of our method by measuring the time it takes to obtain the pose distribution for a single image, which corresponds to the frequency it could run on real time. The fair baseline here is the direct regression method of Liao et al. (2019), using the same ResNet-50 backbone and the same size of MLP. The only difference is that while Liao et al. (2019) only feeds the image descriptor to the MLP, our model concatenates the descriptor to a number of query poses from a grid.

Table S2 shows the results. When using the coarser grid, the performance overhead is negligible with respect to the baseline. This grid has approximately 5° between nearest neighbors, which might be enough for some applications. When increased accuracy is required, our model can use more samples, trading speed for accuracy. Note that the MLP operations are highly parallelizable on GPUs so the processing time grows slower than linear with the grid size.

Table S2. Inference time evaluation. For our method, we measure the time needed to generate the normalized distribution over SO(3) given a single 224 × 224 image. The number of samples correspond to the HEALPix-SO(3) grids of levels 3, 4, and 5, respectively. The coarser grid has an average distance of approximately 5° between nearest neighbors. The processing time growth is slower than linear.

Method	Number of samples	frames/s ↓	Acc@15°↑	Acc@30°↑	Med. (°) ↓
Liao et al.	-	18.2	0.522	0.652	38.2
Ours	37 k	18.3	0.717	0.735	25.1
Ours	295 k	9.1	0.723	0.738	17.6
Ours	2359 k	2.4	0.723	0.738	18.7

5. Ablations

In Figure S4, we show the average log likelihood on the five shapes of SYMSOL I through ablations to various aspects of the method. The top row shows the dependence on the size of the dataset. Performance levels off after 50,000 images per shape, but is greatly diminished for only 10,000 examples. Note almost all of the values for 10,000 images are less than the log likelihood of a uniform distribution over SO(3), $-\log A = -2\log\pi = -2.29$, the 'safest' distribution to output if training is unsuccessful. This indicates overfitting: with only one rotation for each training example, a minimal number of examples is needed to connect all the ground truths with each view of a shape. The network becomes confident about the rotations it has seen paired with a particular view, and assigns small probability to the unseen ground truths, resulting in large negative log likelihood values.

The second row varies the positional encoding applied to the rotations when querying the MLP. 0 positional encoding terms corresponds to no positional encoding at all: the flattened rotation matrix is used as the query rotation. The positional encoding benefits the three shapes with discrete symmetries and is neutral or even slightly negative for the cone and cylinder. Intended to facilitate the representation of high frequency features (Mildenhall et al., 2020), positional encoding helps capture the twelve modes of tetrahedral symmetry with two terms, whereas four are necessary for peak performance on the cube and icosahedron. For all shapes, including more positional encoding terms eventually degrades the performance.

In the third row, we compare different formats for the query rotation, pre-positional encoding. For all shapes, representing rotations as matrices is optimal, with axis-angle and quaternion formats comparable to each other and a fair amount worse. Representing rotations via Euler angles averages out near the log likelihood of a uniform distribution (-2.29), though with a large spread which indicates most but not all runs fail to train.

Finally, the fourth row examines the effect of normalization in the likelihood loss during training. Randomly sampling queries from $\mathbf{SO}(3)$ offers simplicity and freedom over the exact number of queries, but results in inexact normalization of the probability distribution. During training, this leads to roughly equivalent performance as when an equivolumetric grid of queries is used, which can be exactly normalized.

In Figure S5 we show the efficacy of performing gradient ascent to extract the most likely pose from IPDF, given an image. The first way to find the rotation with maximal probability is by sampling from $\mathbf{SO}(3)$ and taking the `argmax` over the unnormalized outputs of IPDF. Predictably, finer resolution of the samples yields more accurate predictions, indicated by shrinking median angular error (left) and growing accuracy at 30° (right) averaged over the categories of Pascal3D+. The second way to produce an estimate leverages the fact that IPDF is fully differentiable. We use the best guess from a sampling of queries as a starting value for gradient ascent on the output of IPDF. The space of valid rotations is embedded in a much larger query space, so we project the updated query back to $\mathbf{SO}(3)$ after every step of gradient ascent, and run it for 100 steps. The estimates returned by gradient ascent yield optimal performance for anything more than 10,000 queries, whereas `argmax` requires more than 500,000 queries for similar results. The difference between the `argmax` and gradient ascent is primarily in the median angular error (left): improvements of an estimate on the order of a degree would benefit this statistic more than the accuracy at 30°.

6. Metrics for evaluation: extended discussion

6.1. Prediction as a distribution: spread and average log likelihood

Here we compare the metrics used in the main text on a simplified example in one dimension, where the ground truth consists of two values: $\{x_{GT}\} = \pm 1$. We evaluate the four distributions (P_1, P_2, P_3, P_4) shown in Figure S6 which model the ground truth to varying degree.

The results for the spread and average log likelihood, defined in the main text, are shown in Table S3. There are several takeaways from this simplified example. The spread, being the average over the ground truths of the minimum error, captures how well *any* of the ground truths are represented. By this metric, P_1 and P_3 are equivalent. When the full set of ground truths is not known at evaluation, the spread ceases to be meaningful.

The average log likelihood measures how well *all* ground truths are represented and is invariant to whether the full set of GTs is provided with each test example, or only a subset. The latter is the predominant scenario for pose estimation datasets, where annotations are not provided for near or exact symmetries. This means only one ground truth is provided for each test example, out of possibly several equivalent values. In Table S3, the average log likelihood ranks the distributions in the order one would expect, with the 'ignorant' uniform distribution (P_4) performing slightly worse than P_1 and P_2, and with

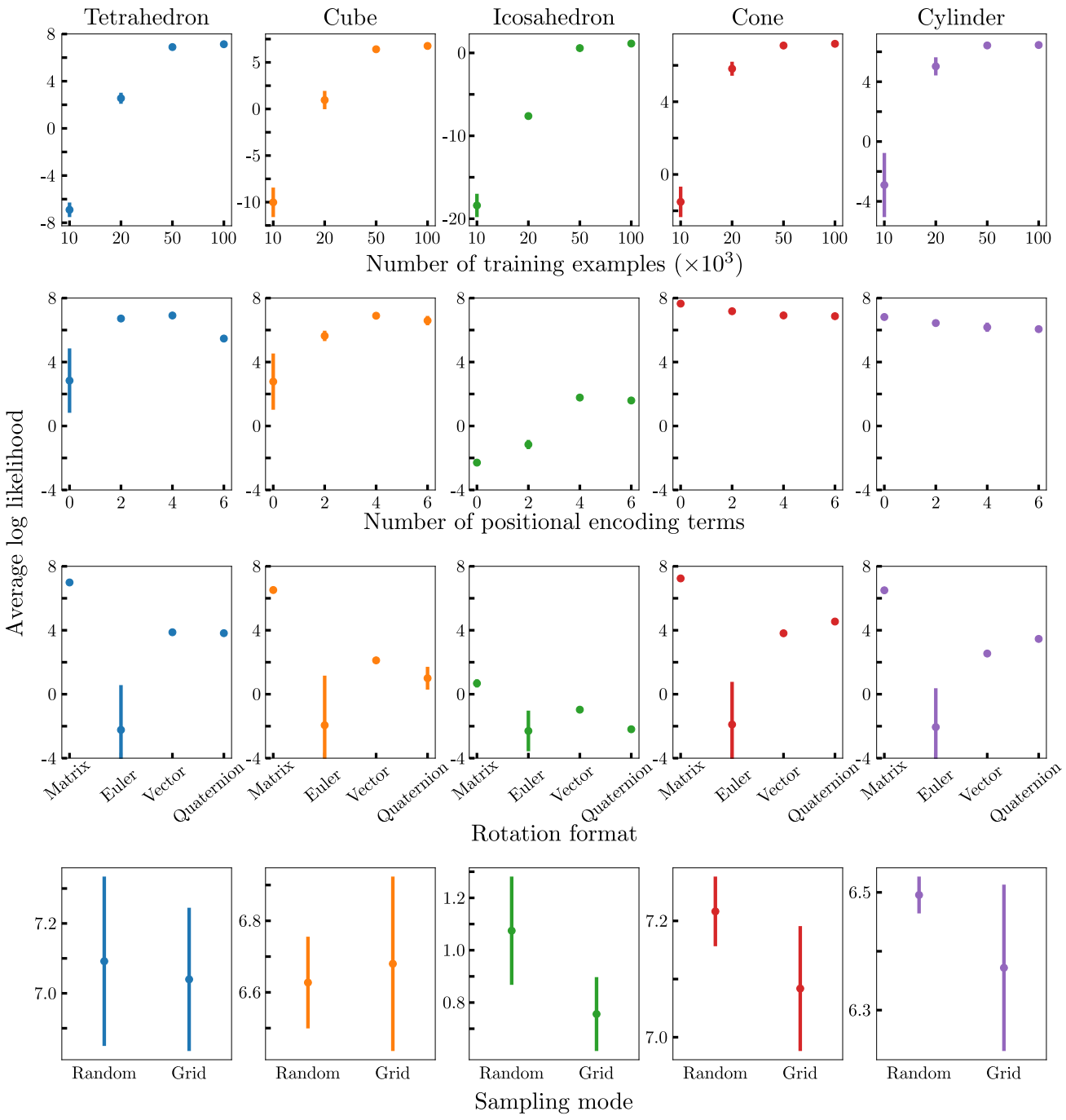

Figure S4. **Ablative studies.** We report the average log likelihood for the shapes of SYMSOL I with various aspects of the method ablated. Error bars are the standard deviation over five networks trained with different random seeds. In the top row, we show the dependence on the size of the dataset, with performance leveling off after 50,000 images per shape. The subsequent row varies the positional encoding, with 0 positional encoding terms corresponding to no positional encoding at all: the flattened rotation matrix is the query rotation. The third row examines the role of the rotation format when querying the MLP (before positional encoding is applied). The final row shows that, during training, inexact normalization arising from the queries being randomly sampled over $\mathbf{SO}(3)$ leads to roughly equivalent performance as the proper normalization from using the equivolumetric grid as the query points. Note that evaluation makes use of an equivolumetric grid in both cases, to calculate the log likelihood.

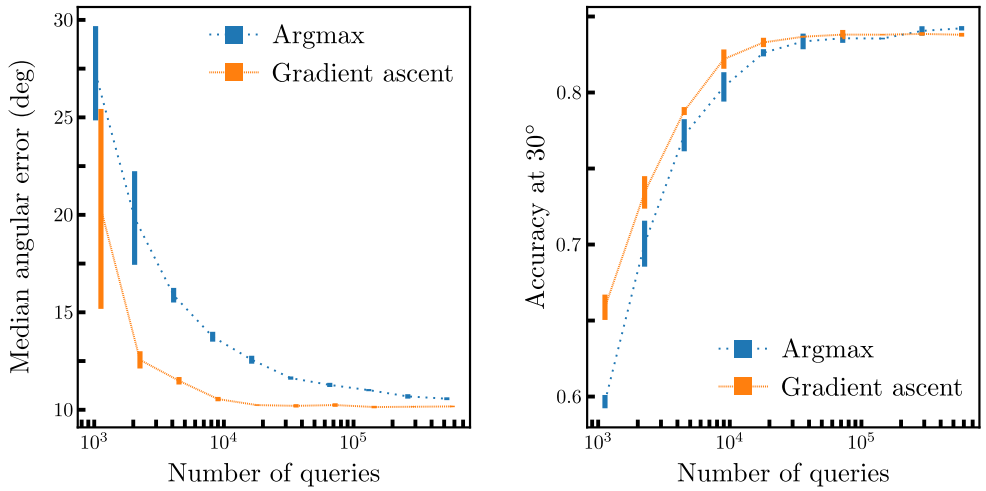

Figure S5. **The efficacy of gradient ascent on Pascal3D+.** We report the average performance across classes on Pascal3D+, for the same IPDF model, using different means to extract a single-valued pose estimate. The error bars are the standard deviation among random sampling attempts, and the curves are slightly offset horizontally for clarity.

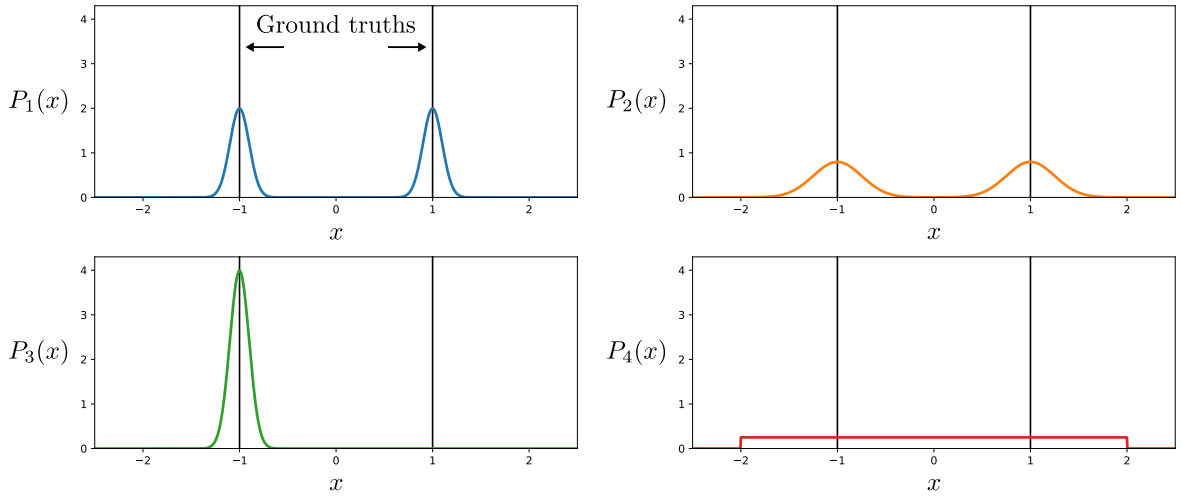

Figure S6. **Distributions modelling a scenario with multiple ground truths.** P_1 and P_2 are mixtures of two normal distributions, with the components centered on the ground truths at $x = \pm 1$. P_3 is a normal distribution centered on only one of the two ground truths. P_4 is a uniform distribution over the interval $[-2, 2]$.

Table S3. Distribution-based evaluation metrics from the main text.

	Full GT at evaluation		Partial GT at evaluation	
Distribution	Spread ↓	Average log likelihood ↑	Spread ↓	Average log likelihood ↑
$P_1 = \frac{1}{2}(\mathcal{N}(-1, 0.1^2) + \mathcal{N}(1, 0.1^2))$	0.08	0.69	1.04	0.69
$P_2 = \frac{1}{2}(\mathcal{N}(-1, 0.25^2) + \mathcal{N}(1, 0.25^2))$	0.20	−0.23	1.10	−0.23
$P_3 = \mathcal{N}(-1, 0.1^2)$	0.08	−98.62	1.04	−98.62
$P_4 = \mathcal{U}(-2, 2)$	0.50	−1.39	1.25	−1.39

P_3 severely penalized for failing to cover both of the ground truths.

6.2. Prediction as a finite set and unknown symmetries: top-k

For the case where only a single ground truth is available, despite potential symmetries, the log-likelihood metric is the only one that is still meaningful unchanged.

Precision and spread metrics are misleading because they penalize correct predictions that don't have a corresponding annotation. Our solution is to drop the precision metric and split the distribution into different modes to compute the spreads, by finding connected components in probability distribution predicted.

The recall metrics are problematic when viewed independently of precision, since they can be easily optimized for by returning a large number of candidate poses covering the whole space. Our solution here is to limit the number of output pose candidates to k, yielding metrics that we denote the top-k accuracy@15°, top-k accuracy@30°, and top-k error. For example, the metrics reported by Liao et al. (2019); Mohlin et al. (2020) on ModelNet10-SO(3) are equivalent to our top-1.

One issue with the top-k evaluation is that we cannot disentangle if errors are due to the dataset (lack of symmetry annotations), or due to the model. Since there is no way around it without expensive annotation, we find it useful to report the top-k for different k, including $k = 1$, where no model errors are forgiven.

Now, for each entry in the dataset, R_{GT} is the single annotated ground truth, the top-k pose predictions are $\{\hat{R}_i\}_{1 \leq i \leq k}$, and we have k normalized probability distributions corresponding to each of the top-k modes, $\{\hat{p}_i\}_{1 \leq i \leq k}$. The following equations describe the metrics,

$$\text{top-}k \text{ accuracy@}\alpha = \left[\min_{1 \leq j \leq k} \left\{ d(R_{GT}, \hat{R}_j) \right\} < \alpha \right], \tag{1}$$

$$\text{top-}k \text{ error} = \min_{1 \leq j \leq k} d(R_{GT}, \hat{R}_j), \tag{2}$$

$$\text{top-}k \text{ spread} = \min_{1 \leq j \leq k} \left\{ \int_{\mathbf{SO}(3)} \hat{p}_j(R) d(R, R_{GT}) \, dR \right\}. \tag{3}$$

Typically, accuracy and spread are averaged over the whole dataset, while the median error over all entries is reported.

7. ModelNet10-SO(3) detailed results

Table S4 extends the ModelNet10-SO(3) table in the main paper and shows per-category metrics.

Since our model predicts a full distribution of rotations, we find the modes of this distribution, by first thresholding by density and then assigning to the same mode any two points that are closer than a second threshold. This method outputs a variable number of modes for each input, as opposed to methods based on mixtures of unimodal distributions (Gilitschenski et al., 2019; Deng et al., 2020), where the number of modes is a fixed hyperparameter.

We then rank the modes by their total probability mass, assign their most likely pose as the mode center, and return the top-k centers for a given k. The evaluation takes the minimum error over the list of candidates, as described in Section 6.2. This kind of top-k evaluation is common practice for image classification tasks like ImageNet (Russakovsky et al., 2015).

As expected, all metrics improve by increasing k, but the symmetric categories, where the single ground-truth evaluation is inappropriate, improve dramatically, suggesting that the lower top-1 performance can indeed be attributed to the lack of symmetry annotations for evaluation and is not a limitation of our model.

8. Implementation specifics

We train with the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with a linear warm up to the base learning rate of 10^{-4} over 1000 steps, and then cosine decay to zero over the remainder of training.

Efficient implementation The input to the MLP is a concatenation of the image descriptor produced by a CNN and a query pose. During both training and inference, we evaluate densities for a large number of poses per image. A naive

		avg.	bathtub	bed	chair	desk	dresser	tv	n. stand	sofa	table	toilet
Acc@15°	Deng et al. (2020)	0.562	0.140	0.788	0.800	0.345	0.563	0.708	0.279	0.733	0.440	0.832
	Prokudin et al. (2018)	0.456	0.114	0.822	0.662	0.023	0.406	0.704	0.187	0.590	0.108	0.946
	Mohlin et al. (2020)	0.693	0.322	0.882	0.881	0.536	0.682	0.790	0.516	0.919	0.446	0.957
	IPDF (ours)	0.719	0.392	0.877	0.874	0.615	0.687	0.799	0.567	0.914	0.523	0.945
	IPDF (ours), top-2	0.868	0.735	0.946	0.900	0.803	0.810	0.883	0.756	0.959	0.932	0.960
	IPDF (ours), top-4	0.904	0.806	0.966	0.905	0.862	0.870	0.899	0.842	0.966	0.956	0.963
Acc@30°	Deng et al. (2020)	0.694	0.325	0.880	0.908	0.556	0.649	0.807	0.466	0.902	0.485	0.958
	Prokudin et al. (2018)	0.528	0.175	0.847	0.777	0.061	0.500	0.788	0.306	0.673	0.183	0.972
	Mohlin et al. (2020)	0.757	0.403	0.908	0.935	0.674	0.739	0.863	0.614	0.944	0.511	0.981
	IPDF (ours)	0.735	0.410	0.883	0.917	0.629	0.688	0.832	0.570	0.921	0.531	0.967
	IPDF (ours), top-2	0.888	0.770	0.953	0.946	0.825	0.812	0.918	0.762	0.968	0.945	0.982
	IPDF (ours), top-4	0.926	0.846	0.973	0.953	0.889	0.874	0.939	0.851	0.975	0.972	0.988
Median Error (°)	Deng et al. (2020)	32.6	147.8	9.2	8.3	25.0	11.9	9.8	36.9	10.0	58.6	8.5
	Prokudin et al. (2018)	49.3	122.8	3.6	9.6	117.2	29.9	6.7	73.0	10.4	115.5	4.1
	Mohlin et al. (2020)	17.1	89.1	4.4	5.2	13.0	6.3	5.8	13.5	4.0	25.8	4.0
	IPDF (ours)	21.5	161.0	4.4	5.5	7.1	5.5	5.7	7.5	4.1	9.0	4.8
	IPDF (ours), top-2	4.9	6.8	4.1	5.5	5.3	4.9	5.3	5.1	3.9	3.7	4.8
	IPDF (ours), top-4	4.8	6.0	4.1	5.4	5.1	4.7	5.2	4.8	3.9	3.7	4.8

Table S4. ModelNet10-SO(3) per-category results.

implementation would replicate and tile image descriptors $\{d_i\}_{0\leq i<N_B}$ and pose queries $\{q_j\}_{0\leq j<N_Q}$, where N_B is the mini-batch size and N_Q is the number of pose queries, and evaluate the first fully connected operation with weights W (before applying bias and nonlinearity) in a batched fashion, as follows,

$$W \begin{bmatrix} d_1 & d_1 & d_1 & \cdots & d_2 & d_2 & d_2 & \cdots \\ q_1 & q_2 & q_3 & \cdots & q_1 & q_2 & q_3 & \cdots \end{bmatrix}. \tag{4}$$

When computed this way, this single step is the computational bottleneck. An alternative, much more efficient method is to observe that

$$W \begin{bmatrix} d_i \\ q_j \end{bmatrix} = W \begin{bmatrix} d_i \\ 0 \end{bmatrix} + W \begin{bmatrix} 0 \\ q_j \end{bmatrix} = W_d d_i + W_q q_j, \tag{5}$$

where $W = [W_d \quad W_q]$. In this manner, W_d can be applied batchwise to image descriptors, yielding a $N_O \times N_B$ output, and W_q can be applied to all query poses independently, yielding a $N_O \times N_Q$ output, where N_O is the number of output channels (number of rows in W). An $N_O \times N_Q \times N_B$ tensor equivalent to Eq. (4) is then obtained via a broadcasting sum, drastically reducing the number of operations.

SYMSOL For the SYMSOL experiments, three positional encoding terms were used for the query, and four fully connected layers of 256 units with ReLU activation for the MLP. One network was trained for all five shapes of SYMSOL I with a batch size of 128 images for 100,000 steps (28 epochs). A different network was trained for each of the three textured shapes of SYMSOL II; these trained with a batch size of 64 images for 50,000 steps (36 epochs). The loss calculation requires evaluating a coverage of points on SO(3) along with the ground truth in order to find the approximate normalization rescaling of the likelihoods. We found that this coverage did not need to be particularly dense, and used 4096 points for training.

T-LESS For T-LESS, only one positional encoding term was used, and the MLP consisted of a single layer of 256 units with ReLU activation. The images were color-normalized and tight-cropped as in Gilitschenski et al. (2019). Training was with a batch size of 64 images for 50,000 steps (119 epochs).

Layer	Activation	Output	
Vision Description Input	-	2048	
Rotation Input	-	[3, 3]	
Flatten	-	9	
Positional Encoding	-	$[2m \times 9]$	
Concatenate	-	$[2048 + 2m \times 9]$	
Dense	ReLU	256	
		...	$\times n$
Dense	None	1	

Table S5. **IPDF architecture.** m is the number of positional encoding frequencies and n is the number of fully connected layers in the MLP. The factor of 2 comes from using both sines and cosines in the positional encoding. The vision description is the result of applying global average pooling to the output of an ImageNet pre-trained ResNet to obtain a 2048-dimensional vector. We use an ImageNet pre-trained Resnet50 for SYMSOL, T-LESS, and ModelNet10-SO(3), and Resenet101 for Pascal3D+.

ModelNet10-SO(3) For ModelNet10-SO(3) (Liao et al., 2019), we use four fully connected layers of 256 units with ReLU activation as in SYMSOL. We train a single model for the whole dataset, for 100,000 steps with batch size of 64. Following Liao et al. (2019) and Mohlin et al. (2020), we concatenate a one-hot encoding of the class label to the image descriptor before feeding it to the MLP.

8.1. Baseline methods

[Deng et al. (2020)] We trained the multi-modal Bingham distribution model from Deng et al. (2020) using their PyTorch code.[1] Note, this is a follow-up work of an earlier paper which references the same implementation (Deng et al., 2020). Our only modification was a minor one to remove the translation component from the model as only the rotation representation needs to be learned. We found the model performed best with the same general settings as used in the reference paper (rWTA loss with two stage training – first stage trains rotations only, the second stage trains both rotations and mixture coefficients).

For the ModelNet10-SO(3) and SYMSOL datasets we trained a single model per shape category, and we found no benefit with increasing the number of components (we used 10 for ModelNet10 and 16 for SYMSOL).

[Gilitschenski et al. (2019)] We trained the multi-modal Bingham distribution model from Gilitschenski et al. (2019) using their PyTorch code.[2] For this baseline we again trained a single model per shape for ModelNet10-SO(3) and SYMSOL. We followed the published approach and trained the model in two stages – first stage with fixed dispersion and second stage updates all distribution parameters. For a batch size of 32, a single training step for a 4-component distribution takes almost 2 seconds on a NVIDIA TESLA P100 GPU. The time is dominated by the lookup table interpolation to calculate the distribution's normalizing term (and gradient), and is linear in the number of mixture components (training with 12 mixture components took over 7 seconds per step). This limited our ability to tune hyperparameters effectively or train with a large number of mixture components.

[Prokudin et al. (2018)] We trained the infinite mixture model from Prokudin et al. (2018) using their Tensorflow code.[3] The only modification was during evaluation: the log likelihood required our method of normalization via equivolumetric grid because representing a distribution over $\mathbf{SO}(3)$ as the product of three individually normalized von Mises distributions lacks the necessary Jacobian. We left the improperly normalized log likelihood in their loss, as it was originally formulated. A different model was trained per shape category of SYMSOL and ModelNet10-SO(3).

Note that our implicit pose distribution is trained as a single model for the whole of SYMSOL I and ModelNet10-SO(3) datasets, so the comparisons against Deng et al. (2020), Gilitschenski et al. (2019), and Prokudin et al. (2018) favor the baselines. Our method outperforms them nevertheless.

[1] https://github.com/Multimodal3DVision/torch_bingham.
[2] https://github.com/igilitschenski/deep_bingham.
[3] https://github.com/sergeyprokudin/deep_direct_stat.

8.2. A note on Pascal3D+ evaluations with respect to Liao et al. and Mohlin et al.

In the Pascal3D+ table in the main paper, and mentioned in that caption, we report numbers for Liao et al. (2019) and Mohlin et al. (2020) which differ from the numbers reported in their papers (these are the rows marked with ‡).

Liao et al. (2019) An error in the evaluation code, reported on github[4], incorrectly measured the angular error – reported numbers were incorrectly lower by a factor of $\sqrt{2}$. The authors corrected the evaluation code for ModelNet10-SO(3) and posted updated numbers, which we show in our paper. However, their evaluation code used for Pascal3D+ still contains the incorrect $\sqrt{2}$ factor: comparing the corrected ModelNet10-SO(3) geodesic distance function[5] and the Pascal3D+ geodesic distance function[6] the $\sqrt{2}$ difference is clear. We sanity checked this by running their Pascal3D+ code with the incorrect metric and were able to closely match the numbers in the paper. In the main paper, we report performance obtained using the corrected evaluation code.

Mohlin et al. (2020) We found that the code released by (Mohlin et al., 2020) uses different dataset splits for training and testing on Pascal3D+ than many of the other baselines we compared against. Annotated images in the Pascal3D+ dataset are selected from one of four source image sets: ImageNet_train, ImageNet_val, PASCALVOC_train, and PASCALVOC_val. Methods like Mahendran et al. and Liao et al. place all the ImageNet images (ImageNet_train, ImageNet_val) in the training partition (i.e. used for training and/or validation): *"We use the ImageNet-trainval and Pascal-train images as our training data and the Pascal-val images as our testing data."* Mahendran et al. (2018), Sec 4. However, in the code released by Mohlin et al. (2020), we observe the test set is sourced from the ImageNet data[7]. We reran the Mohlin et al. code as-is and were able to match their published numbers. After logging both evaluation loops, we confirmed the test data differs between Mohlin et al. and Liao et al.. The numbers we report in the main paper for Mohlin et al. are after modifying the data pipeline to match Liao et al., which is also what we follow for our IPDF experiments. We ran Mohlin et al. with and without augmentation and warping in the data pipeline and chose the best results (which was with warping and augmentation).

References

Deng, H., Bui, M., Navab, N., Guibas, L., Ilic, S., and Birdal, T. Deep Bingham Networks: Dealing with Uncertainty and Ambiguity in Pose Estimation. *arXiv preprint arXiv:2012.11002*, 2020.

Gilitschenski, I., Sahoo, R., Schwarting, W., Amini, A., Karaman, S., and Rus, D. Deep Orientation Uncertainty Learning Based on a Bingham Loss. In *International Conference on Learning Representations*, 2019.

Liao, S., Gavves, E., and Snoek, C. G. M. Spherical Regression: Learning Viewpoints, Surface Normals and 3D Rotations on *n*-Spheres. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Mahendran, S., Ali, H., and Vidal, R. A mixed classification-regression framework for 3d pose estimation from 2d images. *The British Machine Vision Conference (BMVC)*, 2018.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020.

Mohlin, D., Bianchi, G., and Sullivan, J. Probabilistic Orientation Estimation with Matrix Fisher Distributions. In *Advances in Neural Information Processing Systems 33*, 2020.

Prokudin, S., Gehler, P., and Nowozin, S. Deep Directional Statistics: Pose Estimation with Uncertainty Quantification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 534–551, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[4] https://github.com/leoshine/Spherical_Regression/issues/8

[5] https://github.com/leoshine/Spherical_Regression/blob/a941c732927237a2c7065695335ed949e0163922/S3.3D_Rotation/lib/eval/GTbox/eval_quat_multilevel.py#L45

[6] https://github.com/leoshine/Spherical_Regression/blob/a941c732927237a2c7065695335ed949e0163922/S1.Viewpoint/lib/eval/eval_aet_multilevel.py#L135

[7] https://github.com/Davmo049/Public_prob_orientation_estimation_with_matrix_fisher_distributions/blob/4baba6d06ca36db4d4cf8c905c5c3b70ab5fb54a/Pascal3D/Pascal3D.py#L558-L583