Supplementary Material for *Delving into Discrete Normalizing Flows on SO(3) Manifold for Probabilistic Rotation Modeling*

Yulin Liu* Haoran Liu* Yingda Yin* Yang Wang Baoquan Chen† He Wang†
Peking University

1. Why perform affine transformation on quaternion?

It seems straightforward to perform affine transformation on the rotation matrix, however, we find that these methods concentrate the distribution in an undesired way. The simplest way is to left or right multiply a 3×3 invertible matrix W to rotation matrix R, and then use SVD decomposition or the Gram-Schmidt orthogonalization to project the result WR or RW into a rotation matrix, i.e. SVD(WR), SVD(RW), GS(WR), GS(RW).

However, if we use SVD decomposition (SVD(WR) or SVD(RW)) or right Gram-Schmith orthogonalization (GS(RW), it can be proved that it only functions as a rotation and thus has poor expressivity. If we use left Gram-Schmidt orthogonalization (GS(WR)), we concentrate the distribution to 4 different modes where the relative angle between each pair of them is fixed to be 180° as shown in Figure 1, which is usually undesired.

The 4-modal distribution can be understood as multiplying W to the first column and then normalizing it can be interpreted as scaling S^2 to an ellipsoid and then squeezing it to make it concentrated similar to quaternion affine transformation. Multiplying W to the second column and using the Gram-Schmidt orthogonalization can be viewed as scaling a circle S^1 to an oval and then squeezing it to make it concentrated. Those two operations both have the property of antipodal symmetry similar to quaternion affine transformation. However, keeping antipodal symmetry is needed in quaternion affine transformation as q and $-q$ represent the same rotation, while in rotation matrix affine transformation, if (c_1, c_2, c_3) is one mode where c_i is the i-th column of the rotation matrix, then $(-c_1, -c_2, c_3)$, $(-c_1, c_2, -c_3)$ and $(c_1, -c_2, -c_3)$ will also be the mode due to this symmetry.

One can also treat the first two or all three columns of the rotation matrix as a vector and then perform affine transformation on it, but these methods usually lead to even more modes and the inverse of those transformations are much harder to solve.

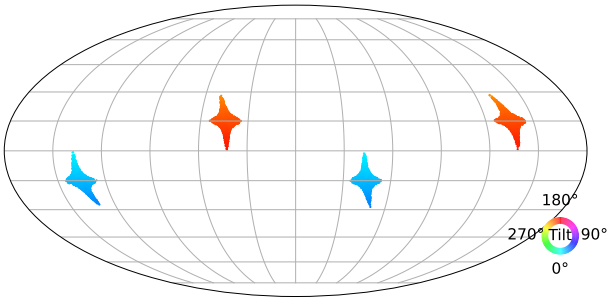

Figure 1. **Illustration of rotation matrix affine transformation.** It can be shown that when the rotation matrix affine transformation is performed by left multiplying 3×3 invertible matrix W to the rotation matrix and then using the Gram-Schmidt orthogonalization. The probability is concentrated to 4 different modes, which usually leads to unsatisfactory performance.

On the contrary, as shown in Figure 5, the affine transformation on quaternion has elegant geometric interpretations and it can concentrate the distribution to only one mode, so we choose to perform affine transformation on quaternion.

Table 1 reports results of abaltion study of quaternion affine transformation and rotation matrix affine transformation on ModelNet10-SO3 datasets. It shows that the overall performance of quaternion affine transformation is better than rotation matrix affine transformation. Note that our average median error is higher because the result is highly affected by *bathtub* category, whereas *bathtub* exhibits symmetry while there is only one ground truth annotation for each image.

2. How to obtain the 4×4 Invertible Matrix?

We present two methods to parameterize the invertible matrix W in quaternion affine transformation. One is to output an unconstrained matrix, and the other is to obtain the matrix by LU decomposition $W = PL(U + S)$, as in [3]. For this method, we follow [3] to construct P as a fixed permutation matrix, L as a lower triangular matrix with ones

*Equal contribution

†He Wang and Baoquan Chen are the corresponding authors ({hewang, baoquan}@pku.edu.cn).

		avg.	bathtub	bed	chair	desk	dresser	tv	n. stand	sofa	table	toilet
Acc@15°	RSVD	0.724	**0.430**	0.861	0.888	0.613	0.716	0.790	0.577	0.888	0.522	0.952
	LSVD	0.734	0.353	0.872	0.899	0.660	0.730	0.804	0.601	0.927	0.526	0.965
	RSmith	0.717	0.400	0.841	0.866	0.622	0.710	0.792	0.588	0.882	0.524	0.951
	LSmith	0.725	0.400	0.873	0.875	0.613	0.722	0.799	0.597	0.913	0.506	0.950
	Ours	**0.760**	0.402	**0.896**	**0.927**	**0.704**	**0.753**	**0.843**	**0.602**	**0.939**	**0.561**	**0.975**
Acc@30°	RSVD	0.731	**0.439**	0.863	0.907	0.618	0.719	0.817	0.580	0.894	0.525	0.954
	LSVD	0.750	0.371	0.880	0.926	0.678	0.742	0.841	0.615	0.934	0.535	0.972
	RSmith	0.726	0.407	0.843	0.887	0.626	0.715	0.820	0.593	0.884	0.526	0.956
	LSmith	0.738	0.411	0.880	0.900	0.632	0.729	0.826	0.605	0.920	0.516	0.959
	Ours	**0.774**	0.419	**0.904**	**0.946**	**0.722**	**0.766**	**0.868**	**0.617**	**0.948**	**0.567**	**0.982**
Median Error (°)	RSVD	**11.3**	**91.4**	**1.4**	**2.8**	2.6	**1.3**	2.8	1.9	**1.4**	5.6	**1.9**
	LSVD	12.2	93.0	1.7	3.1	3.2	1.8	3.0	2.8	1.7	8.9	2.3
	RSmith	16.9	146.8	1.5	2.9	**2.5**	1.4	2.8	**1.8**	**1.4**	5.9	2.0
	LSmith	13.5	106.5	1.5	3.1	2.9	1.5	2.8	2.2	1.5	11.3	2.2
	Ours	14.6	124.8	1.5	**2.8**	2.7	1.5	**2.6**	2.4	1.5	**3.9**	2.0

Table 1. **Ablation of different affine transformation on ModelNet10-SO3 dataset.** We adopt 15° accuaracy, 30° accuaracy and median error as the evaluation metrics. We use uniform distribution in $\mathrm{SO}(3)$ as base distribution. The best performance is shown in **bold** and the second best is with underlined.

on the diagonal, U as an upper triangular matrix with zeros on the diagonal, and S as a diagonal matrix with positive entries.

We conduct an ablation study on different strategies in both unconditional and conditional experiments, and the results are shown in Table 2. We find that 4×4 unconstrained matrix outperforms LU decomposition as shown in the table. This phenomenon may result from the less expressivity of the construction by LU decomposition. Given $W = PL(U + S)$, the sign of the diagonal elements of S is always positive, so it can only represent a subspace of 4×4 invertible matrix, which limits the expressivity. For example, if P is fixed to be the identity matrix, the rotation matrix $diag(-1, -1, 1, 1)$ can't be parameterized via this strategy.

Table 2. **Abalations on parameterization strategies of the** 4×4 **invertible matrix**. We report results on synthetic datasets and SYMSOL-I dataset with log-likelihood (↑) as the evaluation metric. We also report results on ModelNet10-SO3 dataset and adpot acc@15, acc@30 and error median as the evaluation metrics.

Synthetic datasets	avg.	peak	cone	cube	line
Ours (unconstrained M)	**7.23**	**13.93**	**8.99**	**4.81**	**1.38**
Ours (LU)	**7.23**	13.92	**8.99**	**4.81**	**1.38**

SYMSOL I	avg.	cone	cube	cyl.	ico.	tet.
Ours (unconstrained M)	**10.38**	**10.05**	**11.64**	**9.54**	**8.26**	**12.43**
Ours (LU)	8.57	9.95	8.60	9.38	3.94	10.99

ModelNet10-SO3	acc@15↑	acc@30↑	Median Error↓
Ours (unconstrained M)	**0.760**	**0.774**	**14.6**
Ours (LU)	0.727	0.738	16.7

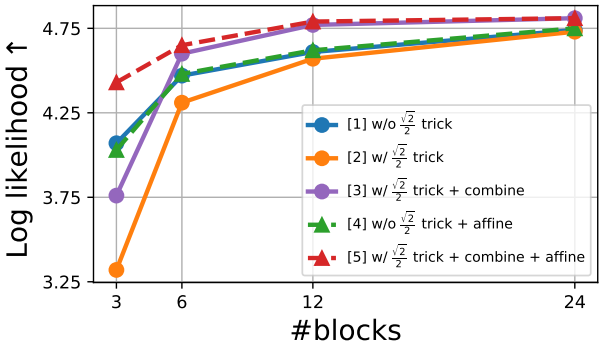

Figure 2. **Abalations on $\frac{\sqrt{2}}{2}$ trick on Cube dataset.** We plot results of log likelihood with different blocks and choice of structure. Curve[1] stands for single Mobius transformation without trick, [2] for single Mobius transformation with trick, [3] for 64-combination Mobius transformation with trick, [4] for single Mobius transformation without trick with affine transformation and [5] for 64-combination Mobius transformation with trick with affine transformation.

3. Ablation $\frac{\sqrt{2}}{2}$ trick

We present a $\frac{\sqrt{2}}{2}$ trick to alleviate discontinuity encountered in *linear combination* of Mobius transformations. We have to confess that the expressivity of single Mobius transformation are reduced due to restriction of $\|\omega\|$, however, as it enables *linear combination* of Mobius transformations, the general expressivity are gained.

Without $\frac{\sqrt{2}}{2}$ trick, we can not compute the inverse process of *linear combination* of Mobius transformations due

to the ambiguity of Mobius combination. So we have to compare the gain from using linear combination and the cost of this $\frac{\sqrt{2}}{2}$ trick.

We conduct ablation study of $\frac{\sqrt{2}}{2}$ trick on Cube dataset in Fig. 2. Seen from curves [1-2], the introduction of $\frac{\sqrt{2}}{2}$ trick indeed limits the performance; however, from [3] vs. [1] and [5] vs. [4], the accommodation of linear combination in return provides larger expressivity.

4. More Discussions on using other Normalizing Flows on SO(3)

4.1. ReLie

ReLie [2] performs normalizing flows on the Lie algebra of Lie groups, where for rotations in $\mathrm{SO}(3)$, the Lie algebra is the axis-angle representation ($\theta\mathbf{e}$) in $\mathbb{R}^3$. ReLie applies normalizing flows in Euclidean space and then maps the Euclidean samples back to $\mathrm{SO}(3)$ space through the exponential map. Noticing that the axis-angle representation periodically covers $\mathrm{SO}(3)$ space, i.e., $(\theta + 2i\pi)\mathbf{e}$, $i \in \mathbb{Z}$ represent the same rotation, ReLie restricts the output of the Euclidean normalizing flows in a sphere with the radius r by an $r \cdot \tanh(\cdot)$ operation to resolve the infinity-to-one issue. However, ReLie still exhibits several drawbacks. Firstly, even if limited in a sphere with the radius of π, axis-angle representation is not a diffeomorphism to $\mathrm{SO}(3)$ [11], thus the normalizing flows cannot build diffeomorphic mappings between Lie algebra and Lie group and suffer from discontinuous rotation representation. Secondly, with the non-linear $\tanh(\cdot)$ operation, at the inverse stage, $\tanh^{-1}(\cdot)$ can yields infinitely large values, resulting in numerical instability (see Figure 3). Note that this issue can not be solved by replacing $\tanh(\cdot)$ by other functions or varying the value r, since a non-linear mapping function is always needed to restrict $\mathbb{R}^3$ into a sphere.

In our experiment, ReLie fails to fit the *peak* distribution (see Table 1 in the main paper). This is because the normalizing flows learn to push mostly all the points to the peak, and in the inverse process, points that are not close to the peak are pushed back near to the surface of the r-sphere in Lie algebra, which yields `NAN` in training and breaks the process.

4.2. ProHMR

ProHMR [4] leverages 6D rotation representation and considers the first two columns of the rotation matrix as a 6D Euclidean vector. Thus, it applies Euclidean normalizing flows on the 6D vectors and maps the samples back to $\mathrm{SO}(3)$ by Gram-Schmidt orthogonalization. Without any constraint, the infinity-to-one mapping clearly makes the probability density of a given rotation intractable. This is because a rotation in $\mathrm{SO}(3)$ corresponds to infinity points in Euclidean space and the PDF of the rotation should be the integration of all the corresponding Euclidean points. Due to the unavailability of the probability densities, we do not incorporate it as our baseline in experiments.

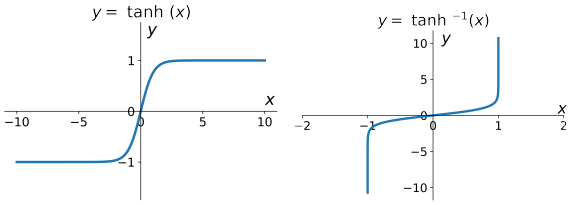

Figure 3. **Illustration of $\tanh(x)$ function (left) and $\tanh^{-1}(x)$ function (right).** When $|x|$ is close to 1, $|\tanh^{-1}(x)|$ approaches infinity.

5. Application: Entropy Estimation of Arbitrary Distribution

One outstanding feature of our Normalizing Flows compared to other probability inference methods on $\mathrm{SO}(3)$ (like [8]) is its ability for efficient samples. [8] can only sample by querying a large number of rotations and calculating the probability density function. For highly peaked distribution, this method may fail as it is hard to have queries that are enough close to the peak such that the probability is not close to zero. However, we can sample by transforming z sampled from a base distribution through our flows. Efficient sampling makes it possible to estimate properties of data x, for example, *entropy* can be estimated via Monte Carlo:

$$S = E[\log p(x)]. \tag{1}$$

In this experiment, we compare our rotation NFs with Implicit-PDF in approximating the entropy of the target distributions. In order to obtain the ground truth entropy for evaluation, we adopt multiple matrix Fisher distributions (whose entropy can be analytically computed) with different parameters as the target distributions. We sample 600k points from each target distribution as the training data and evaluate both our method and Implicit-PDF by randomly sampling N (N=5, 10, 100, 1k, 10k) points from the learned distributions. The results are shown in Figure 4. We can see that even when the sampling size is small, our rotation normalizing flows still achieve accurate estimation of entropy for different target distributions, while Implicit PDF fails to do so.

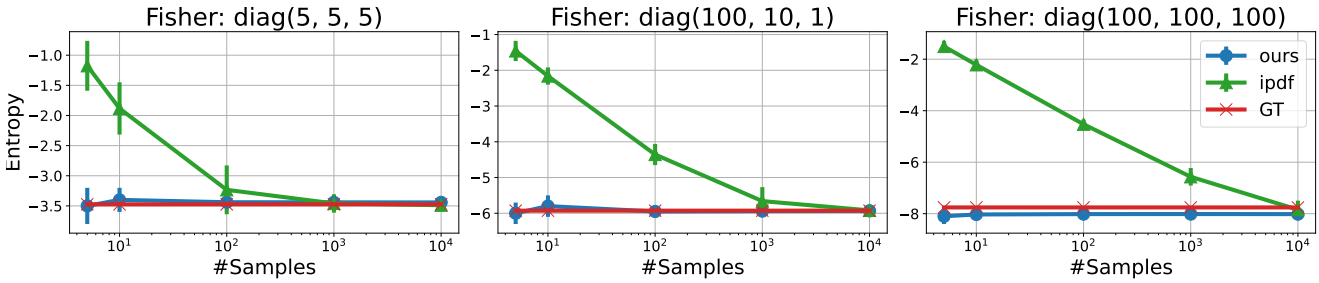

Figure 4. **Results of entropy estimation for three different target distributions.** We compare the mean and variance of estimated entropy after 100 times of sampling. We use uniform distribution as base distribution. Note that the horizontal axis is in log scale.

6. Calculations of Our Normalizing Flows

6.1. Determinate of Jacobian

Mobius Transformation The forward pass of Mobius transformation is defined as follows:

$$c' = f_\omega(c) = \frac{1 - \|\omega\|^2}{\|c - \omega\|^2}(c - \omega) - \omega. \tag{2}$$

Where c, c' are vectors $\in \mathcal{S}^2$, ω is a point $\in \mathbb{R}^3$ that satisfies $\|\omega\| < \frac{\sqrt{2}}{2}$.The Jacobian matrix is given by partial derivatives of c' with respect to c,

$$J = \frac{\mathrm{d}c'}{\mathrm{d}c} = \frac{1 - \|\omega\|^2}{\|c - \omega\|^2}(I - 2\frac{(c - \omega)^T(c - \omega)}{\|c - \omega\|^2}) \tag{3}$$

where I is the 3×3 identity matrix.

In our implementation, the Mobius transformation is a one-degree-of-freedom mapping, which maps unit vectors lying in the plane vertical to the unchanged condition column to unit vectors in the same plane. As any three-dimensional vector c in the plane can be parameterized by an angle θ to reference vector c_2 where c_2 and c_3 are one pair of the orthogonal basis of the plane.

$$c = \cos\theta c_2 + \sin\theta c_3 \tag{4}$$

$$c' = \cos\theta' c_2 + \sin\theta' c_3 \tag{5}$$

Via a change of variable formula, the determinant of Jacobian can be calculated as:

$$\frac{\mathrm{d}\theta'}{\mathrm{d}\theta} = \frac{\mathrm{d}\theta'}{\mathrm{d}c'}\frac{\mathrm{d}c'}{\mathrm{d}\theta} \tag{6}$$

Given that c, c', c_2, c_3 are unit vectors, the absolute value of determinant of Jacobian is thus calculated as:

$$|\det J| = |\frac{\mathrm{d}\theta'}{\mathrm{d}\theta}| = \|\frac{\mathrm{d}c'}{\mathrm{d}\theta}\| \tag{7}$$

Where $\frac{\mathrm{d}c'}{\mathrm{d}\theta}$ are given by:

$$\frac{\mathrm{d}c'}{\mathrm{d}\theta} = \frac{\mathrm{d}c'}{\mathrm{d}c}\frac{\mathrm{d}c}{\mathrm{d}\theta} \tag{8}$$

$$\frac{\mathrm{d}c}{\mathrm{d}\theta} = -\sin\theta c_2 + \cos\theta c_3 \tag{9}$$

Affine Transformation The forward of quaternion affine transformation is given by:

$$\mathbf{q}' = g(\mathbf{q}) = \frac{W\mathbf{q}}{\|W\mathbf{q}\|} \tag{10}$$

The determinate of Jacobian of affine transformation is very straightforward and is given by:

$$\det J(\mathbf{q}) = \frac{\det W}{\|W\mathbf{q}\|^4} \tag{11}$$

6.2. Inverse

Mobius Transformation The inverse of Mobius transformation is implemented by connecting $-c'$ and parameters ω with a straight line that intersects with S^D at c. The explicit expression for the inverse of Mobius transformation is given as follows:

$$f_\omega^{-1}(c') = \frac{1 - \|\omega\|^2}{\|c' + \omega\|^2}(c' + \omega) + \omega \tag{12}$$

The forward process and the inverse process have the same computational complexity.

However, as we utilize a linear combination, i.e. the weighted sum of angles to improve the expressivity of Mobius transformation, there is no analytical inverse for combined Mobius transformation, so we use binary search algorithms to find its inverse, as the combined angle θ' are constrained within$(-\pi/2, \pi/2)$ and there's no discontinuity around the boundary $\pm\pi/2$. The computational complexity for inversing the Mobius transformation is $O(\log(1/\epsilon)$, where ϵ is the computational error.

Affine Transformation The inverse of affine transformation is implemented as:

$$g^{-1}(\mathbf{q}') = \frac{W^{-1}\mathbf{q}'}{\|W^{-1}\mathbf{q}'\|} \tag{13}$$

Where W is a 4×4 invertible matrix and $\mathbf{q}'$ is a quaternion. We find that the inverse and forward processes of

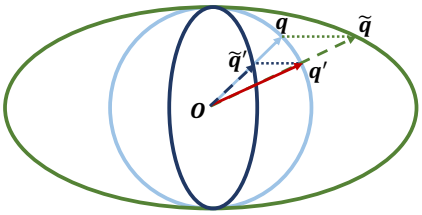

Figure 5. Forward and inverse of scaling and normalization operation in quaternion affine transformation. $\mathbf{q}$ are scaled to $\tilde{\mathbf{q}}$ by multiplying s_1, s_2, s_3, s_4 on its coordinates and then normalized to the unit sphere $\mathbf{q}'$. Inversely, $\mathbf{q}'$ are scaled to $\tilde{\mathbf{q}}'$ by $1/s_1, 1/s_2, 1/s_3, 1/s_4$ and normalized to $\mathbf{q}$. $\Delta O\tilde{\mathbf{q}}'\mathbf{q}'$ and $\Delta O\mathbf{q}\tilde{\mathbf{q}}$ are similar triangles.

affine transformation have the same expression form while the invertible matrix W in the forward path is replaced by its inverse matrix W^{-1} in the inverse path. We attribute this feature to the geometry of affine transformation.

Via SVD decomposition, the affine transformation is divided into two types of operations: 1)rotation, 2) scaling and normalization. The inverse of rotation operation R takes place by multiplying R^{-1} which is straightforward, whereas the inverse of scaling and normalization is also implemented by taking the inverse of S which sounds nontrivial.

The feature is caused by a geometry coincidence, as illustrated in Figure 5. $\mathbf{q}$ are scaled to $\tilde{\mathbf{q}}$ by multiplying s_1, s_2, s_3, s_4 on its coordinates and then normalized to the unit sphere $\mathbf{q}'$. Inversely, $\mathbf{q}'$ are scaled to $\tilde{\mathbf{q}}'$ by $1/s_1, 1/s_2, 1/s_3, 1/s_4$ and normalized to points intersects with S^3. $\Delta O\tilde{\mathbf{q}}'\mathbf{q}'$ and $\Delta O\mathbf{q}\tilde{\mathbf{q}}$ are similar triangles as:

$$\frac{\overline{O\tilde{\mathbf{q}}'}}{\overline{O\mathbf{q}}} = \frac{\overline{O\mathbf{q}'}}{\overline{O\tilde{\mathbf{q}}}} = \frac{\overline{\tilde{\mathbf{q}}'\mathbf{q}'}}{\overline{\mathbf{q}\tilde{\mathbf{q}}}} \tag{14}$$

As $O, \mathbf{q}', \tilde{\mathbf{q}}$ are on the same line and $\angle\tilde{\mathbf{q}}'O\mathbf{q}' = \angle\mathbf{q}O\tilde{\mathbf{q}}$ due to the property of similar triangles, $O, \tilde{\mathbf{q}}', \mathbf{q}$ are on the same line, thus normalized points $\tilde{\mathbf{q}}'$ are $\mathbf{q}$, which is the inverse of affine transformation.

7. ModelNet10-SO(3) and Pascal3D+ detailed results

Table 3 and Table 4 show per-category metrics for ModelNet10-SO(3) and Pascal3D+ dataset respectively. Note that our method achieves significantly lower median errors in experiments on ModelNet10-SO3 in all of the categories except the *batutub* category, where *batutub* images are well-known to exhibit severe symmetry and all the methods have unreasonably poor performance.

8. Analysis of Evaluation Methods

8.1. Log Likelihood Evaluation

In case of main experiments (rotation distribution fitting) and SYMSOL I/II datasets, we exploit **log likelihood** averaged over test data or test set annotations as evaluation metrics. It is implemented by transforming data through the flow to its base distribution and predicting log-likelihood.

8.2. Sample Generation

Sometimes pose estimation tasks require single-value prediction. Estimated rotations are generated via flowing samples from base distribution inversely and obtained a set of samples following target distribution. We pick the sample with the highest probability likelihood as our single prediction.

In SYMSOL I datasets, we present ***spread*** following IPDF [8]. Given a complete set of equivalent ground truth rotations, *spread* is defined as expectation of angular deviation to any of the ground truth values: $\mathbb{E}_{R\sim p(R|x)}[\min_{R'\in\{R_{GT}\}} d(R, R')]$, where x is the given image and $d(R, R')$ is the relative angles between R and R'. This measures how close the samples are to the ground truth poses. It is calculated as the mean of the relative angle between each of the generated samples and the ground truth poses closest to it.

In ModelNet10-SO(3) and Pascal3D+ datasets, we calculate the angular error of our single-value rotation prediction with single ground truth value, and report **error median**, accuracy at threshold 15° (for ModelNet-SO(3) only), 30°, i.e. **Acc@15, Acc@30**.

8.3. Ablations of other sample generation methods

We compare our inference method to those used in IPDF [8], which is implemented via first evaluating a grid $\{R_i\}$ or samples randomly sampled on $\mathrm{SO}(3)$, and solving

$$R_x^* = \arg\max_{R\in\mathrm{SO}(3)} p(R|x) \tag{15}$$

to pick the single pose prediction (We called the method *PDF* for short). The prediction can be made more accurate with gradient ascent of Equation (15) (*Grad* for short). The equivolumetric grids are generated first generating equal area grids on the 2-sphere with HEALPix method, and cover $\mathrm{SO}(3)$ with Hopf fibration by sampling equivolumetric points on a circle with each point on the 2-sphere.

Table 5 shows that with few numbers of samples ($\sim$ 5), our method is capable of capturing accurate results with little computational cost ($\sim$ 1-2 minutes for evaluating the whole test sets). While for PDF and Grad, they require high-resolution grids to obtain high accuracy and the computation cost is not affordable when evaluating a large dataset.

		avg.	bathtub	bed	chair	desk	dresser	tv	n. stand	sofa	table	toilet
Acc@15°	Deng et al. [1]	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. [9]	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. [7]	0.693	0.322	0.882	0.881	0.536	0.682	0.790	0.516	0.919	0.446	0.957
	IPDF [8]	0.719	0.392	0.877	0.874	0.615	0.687	0.799	0.567	0.914	0.523	0.945
	Ours(Uni.)	**0.760**	0.402	**0.896**	**0.927**	**0.704**	**0.753**	**0.843**	**0.602**	**0.939**	**0.561**	0.975
	Ours(Fisher)	0.744	**0.439**	0.890	0.909	0.638	0.715	0.810	0.585	0.938	0.535	**0.978**
Acc@30°	Deng et al. [1]	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. [9]	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. [7]	0.757	0.403	**0.908**	0.935	0.674	0.739	0.863	0.614	0.944	0.511	0.981
	IPDF [8]	0.735	0.410	0.883	0.917	0.629	0.688	0.832	0.570	0.921	0.531	0.967
	Ours(Uni.)	**0.774**	0.419	0.904	**0.946**	**0.722**	**0.766**	**0.868**	**0.617**	**0.948**	**0.567**	0.982
	Ours(Fisher)	0.768	**0.460**	0.898	0.934	0.694	0.738	0.859	0.615	**0.948**	0.544	**0.987**
Median Error (°)	Deng et al. [1]	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. [9]	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. [7]	17.1	**89.1**	4.4	5.2	13.0	6.3	5.8	13.5	4.0	25.8	4.0
	IPDF [8]	21.5	161.0	4.4	5.5	7.1	5.5	5.7	7.5	4.1	9.0	4.8
	Ours(Uni.)	**14.6**	124.8	**1.5**	**2.8**	**2.7**	**1.5**	**2.6**	**2.4**	**1.5**	**3.9**	**2.0**
	Ours(Fisher)	**12.2**	91.6	1.8	3.0	5.5	2.0	3.2	4.3	1.6	6.7	2.1

Table 3. **Numerical results of rotation regression on ModelNet10-SO3 dataset.** We adopt 15° accuaracy, 30° accuaracy and median error as the evaluation metrics. The best performance is shown in **bold** and the second best is with underlined.

		avg.	aero	bike	boat	bottle	bus	car	chair	table	mbike	sofa	train	tv
Acc@30°	Liao et al. [5]	0.819	0.82	0.77	0.55	0.93	0.95	0.94	0.85	0.61	0.80	0.95	**0.83**	0.82
	Mohlin et al. [7]	0.825	0.90	0.85	0.57	0.94	0.95	**0.96**	0.78	0.62	0.87	0.85	0.77	0.84
	Prokudin et al. [9]	0.838	**0.89**	0.83	0.46	**0.96**	0.93	0.90	0.80	0.76	0.90	0.90	0.82	**0.91**
	Tulsiani & Malik [10]	0.808	0.81	0.77	0.59	0.93	**0.98**	0.89	0.80	0.62	0.88	0.82	0.80	0.80
	Mahendran et al. [6]	0.859	0.87	0.81	**0.64**	**0.96**	0.97	0.95	0.92	0.67	0.85	**0.97**	0.82	0.88
	IPDF [8]	0.837	0.81	0.85	0.56	0.93	0.95	0.94	0.87	**0.78**	0.85	0.88	0.78	0.86
	Ours(Uni.)	0.827	0.83	0.78	0.56	0.95	0.96	0.93	0.87	0.62	0.85	0.90	0.81	0.86
	Ours(Fisher)	**0.863**	**0.89**	**0.89**	0.55	**0.96**	**0.98**	0.95	**0.94**	0.67	**0.91**	0.95	0.82	0.85
Median Error (°)	Liao et al. [5]	13.0	13.0	16.4	29.1	10.3	4.8	6.8	11.6	12.0	17.1	12.3	8.6	14.3
	Mohlin et al. [7]	11.5	10.1	15.6	24.3	7.8	3.3	5.3	13.5	12.5	12.9	13.8	7.4	11.7
	Prokudin et al. [9]	12.2	9.7	15.5	45.6	**5.4**	2.9	4.5	13.1	12.6	**11.8**	9.1	**4.3**	12.0
	Tulsiani & Malik [10]	13.6	13.8	17.7	21.3	12.9	5.8	9.1	14.8	15.2	14.7	13.7	8.7	15.4
	Mahendran et al. [6]	10.1	**8.5**	14.8	**20.5**	7.0	3.1	5.1	9.3	11.3	14.2	10.2	5.6	11.7
	IPDF [8]	10.3	10.8	12.9	23.4	8.8	3.4	5.3	10.0	**7.3**	13.6	9.5	6.4	12.3
	Ours(Uni.)	10.2	8.9	15.2	24.9	6.9	**2.9**	**4.3**	**8.7**	10.7	12.8	9.3	6.3	**11.3**
	Ours(Fisher)	**9.9**	9.6	**12.4**	22.7	7.5	3.1	4.8	9.2	8.6	13.5	**8.6**	6.7	11.6

Table 4. **Numerical results of rotation regression on Pascal3D+ dataset.** We adopt 30° accuaracy and median error as the evaluation metrics. The best performance is shown in **bold** and the second best is with underlined.

9. Implementation Details

We use Adam as our optimizer with a learning rate of 1e-4.

Detailes of $\frac{\sqrt{2}}{2}$ trick In Mobius coupling layers, each ω is predicted by first predicting ω' by a [64, 64, 64, 64] multi-layer perceptron (MLP), ReLU as activation function and a residual connection between the first and the last layer with the condition part as input, and then projecting ω' to the vertical plane of condition column via Gram-Smith process. It is then reparameterized by $\omega = \frac{0.7}{1+\|\omega\|}\omega$ to constrain it within $\frac{\sqrt{2}}{2}$ sphere.

Unconditional Experiments we use 24 layers of blocks (24 Mobius + 24 Affine) with the combination of 64 Mobius transformations at a batch size of 64 for 50k steps.

SYMSOL In experiments on SYMSOL, we add a conditional Affine transformation at the beginning of our flow (near the target distribution) and use 21 layers of blocks (64-combination conditional Mobius coupling layers + unconditional affine transformation) at a batch size of 128 for 900k steps. Conditional features with dimension 512 are captured by an ImageNet pre-trained ResNet50 following settings in IPDF [8]. We trained a single model for SYMSOL I for 5 categories and trained a single for each category in SYMSOL also following IPDF.

ModelNet10-SO3 For experiments on ModelNet10-SO3, 24 layers of blocks (64-combination conditional Mobius coupling layers + conditional affine transformation) at a

Table 5. **Abalations on evaluation time**. We report results on conditional rotation regression tasks of ModelNet10-SO3. We compare the time required to generate estimated rotation via inversing the flow (Ours), inferringthe probability of a grid(PDF), gradient descent of probility of a grid(Grad). The number of samples used in PDF and Grad corespond to the HEALPix-SO(3) grids of levels 2, 3 and 4 respectively.

Method	Number of samples	evaluation time(total)↓	evulation time(s)/iter↓	Acc@15° ↑	Acc@30° ↑	Med. (°)↓
Ours	1	1.2 min	0.508	0.740	0.760	14.4
Ours	5	1.3 min	0.525	0.759	0.773	14.0
Ours	50	1.4 min	0.555	0.759	0.773	14.1
Ours	500	4.8 min	1.97	**0.760**	0.773	14.8
Ours	5k	0.7 h	16.8	**0.760**	**0.774**	14.6
PDF	4.6k	0.6 h	0.122	0.480	0.611	31.8
PDF	37k	1.4 h	0.289	0.699	0.745	18.2
PDF	295k	9.5 h	1.88	0.755	0.771	18.0
Grad	4.6k	2.2h	0.446	0.496	0.612	31.6
Grad	37k	3.0h	0.610	0.704	0.748	16.6
Grad	295k	11.0h	2.20	0.758	0.772	**13.8**

batch size of 128 for 250k steps are used. We also use the strategy of learning rate decay and multiply the learning rate by 0.1 at 30 and 40 epochs. Conditional features with dimension 2048 are captured by an ImageNet pre-trained ResNet101 following settings in Mohlin et al. [7]. We train a single model for the whole dataset and use a 32-dimensional positional embedding for 10 categories.

Pascal3D+ For Pascal3D+, 24 layers of blocks (64-combination conditional Mobius coupling layers + conditional affine transformation) at a batch size of 128 for 350k steps are used. We also use the strategy of learning rate decay and multiply the learning rate by 0.1 at 200k and 250k iterations when using uniform distribution as base distribution, 12 and 15 epochs when using pre-trained fisher as base distribution respectively. Conditional features with dimension 2048 are captured by an ImageNet pre-trained ResNet101. We train a single model for the whole dataset and use a 32-dimensional positional embedding for 10 categories.

References

[1] Haowen Deng, Mai Bui, Nassir Navab, Leonidas Guibas, Slobodan Ilic, and Tolga Birdal. Deep bingham networks: Dealing with uncertainty and ambiguity in pose estimation. *International Journal of Computer Vision*, pages 1–28, 2022. 6

[2] Luca Falorsi, Pim de Haan, Tim R Davidson, and Patrick Forré. Reparameterizing distributions on lie groups. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3244–3253. PMLR, 2019. 3

[3] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018. 1

[4] Nikos Kolotouros, Georgios Pavlakos, Dinesh Jayaraman, and Kostas Daniilidis. Probabilistic modeling for human mesh recovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11605–11614, 2021. 3

[5] Shuai Liao, Efstratios Gavves, and Cees GM Snoek. Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-spheres. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9759–9767, 2019. 6

[6] Siddharth Mahendran, Haider Ali, and Rene Vidal. A mixed classification-regression framework for 3d pose estimation from 2d images. *arXiv preprint arXiv:1805.03225*, 2018. 6

[7] David Mohlin, Josephine Sullivan, and Gérald Bianchi. Probabilistic orientation estimation with matrix fisher distributions. *Advances in Neural Information Processing Systems*, 33:4884–4893, 2020. 6, 7

[8] Kieran Murphy, Carlos Esteves, Varun Jampani, Srikumar Ramalingam, and Ameesh Makadia. Implicit-pdf: Non-parametric representation of probability distributions on the rotation manifold. *arXiv preprint arXiv:2106.05965*, 2021. 3, 5, 6

[9] Sergey Prokudin, Peter Gehler, and Sebastian Nowozin. Deep directional statistics: Pose estimation with uncertainty quantification. In *Proceedings of the European conference on computer vision (ECCV)*, pages 534–551, 2018. 6

[10] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015. 6

[11] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5738–5746, 2019. 3