Quaternion Equivariant Capsule Networks for 3D Point Clouds

Yongheng Zhao[1,3,*], Tolga Birdal[2,*], Jan Eric Lenssen[4], Emanuele Menegatti[1], Leonidas Guibas[2], and Federico Tombari[3,5]

[1] University of Padova [2] Stanford University [3] TU Munich
[4] TU Dortmund [5] Google

Abstract. This document supplements our paper **Quaternion Equivariant Capsule Networks for 3D Point Clouds** by providing proofs regarding the dynamic routing, architectural details, discussions on the technical fronts as well as computational aspects, additional experiments and visual insights.

1 Proof of Proposition 1

Before presenting the proof we recall the three individual statements contained in Prop. 1:

1. $\mathcal{A}(\mathbf{g} \circ \mathbf{S}, \mathbf{w})$ is left-equivariant: $\mathcal{A}(\mathbf{g} \circ \mathbf{S}, \mathbf{w}) = \mathbf{g} \circ \mathcal{A}(\mathbf{S}, \mathbf{w})$.
2. Operator $\mathcal{A}$ is invariant under permutations: $\mathcal{A}(\{\mathbf{q}_{\sigma(1)}, \dots, \mathbf{q}_{\sigma(Q)}\}, \mathbf{w}_\sigma) = \mathcal{A}(\{\mathbf{q}_1, \dots, \mathbf{q}_Q\}, \mathbf{w})$.
3. The transformations $\mathbf{g} \in \mathbb{H}_1$ preserve the geodesic distance $\delta(\cdot)$.

Proof. We will prove the propositions in order.

1. We start by transforming each element and replace $\mathbf{q}_i$ by $(\mathbf{g} \circ \mathbf{q}_i)$ of the cost defined in Eq. 4 of the main paper:

$$\mathbf{q}^\top \mathbf{M} \mathbf{q} = \mathbf{q}^\top \Big(\sum_{i=1}^{Q} w_i \mathbf{q}_i \mathbf{q}_i^\top \Big) \mathbf{q} \tag{S1}$$

$$= \mathbf{q}^\top \Big(\sum_{i=1}^{Q} w_i (\mathbf{g} \circ \mathbf{q}_i)(\mathbf{g} \circ \mathbf{q}_i)^\top \Big) \mathbf{q} \tag{S2}$$

$$= \mathbf{q}^\top \Big(\sum_{i=1}^{Q} w_i \mathbf{G} \mathbf{q}_i \mathbf{q}_i^\top \mathbf{G}^\top \Big) \mathbf{q} \tag{S3}$$

$$= \mathbf{q}^\top \Big(\mathbf{G}\mathbf{M}_1\mathbf{G}^\top + \cdots + \mathbf{G}\mathbf{M}_Q\mathbf{G}^\top \Big) \mathbf{q}$$

$$= \mathbf{q}^\top \mathbf{G} \Big(\mathbf{M}_1\mathbf{G}^\top + \cdots + \mathbf{M}_Q\mathbf{G}^\top \Big) \mathbf{q} \tag{S4}$$

$$= \mathbf{q}^\top \mathbf{G} \Big(\mathbf{M}_1 + \cdots + \mathbf{M}_Q \Big) \mathbf{G}^\top \mathbf{q} \tag{S5}$$

$$= \mathbf{q}^\top \mathbf{G} \mathbf{M} \mathbf{G}^\top \mathbf{q} \tag{S6}$$

$$= \mathbf{p}^\top \mathbf{M} \mathbf{p}, \tag{S7}$$

where $\mathbf{M}_i = w_i \mathbf{q}_i \mathbf{q}_i^\top$ and $\mathbf{p} = \mathbf{G}^\top \mathbf{q}$. From orthogonallity of $\mathbf{G}$ it follows $\mathbf{p} = \mathbf{G}^{-1}\mathbf{q} \implies \mathbf{g} \circ \mathbf{p} = \mathbf{q}$ and hence $\mathbf{g} \circ \mathcal{A}(\mathbf{S}, \mathbf{w}) = \mathcal{A}(\mathbf{g} \circ \mathbf{S}, \mathbf{w})$.

2. The proof follows trivially from the permutation invariance of the symmetric summation operator over the outer products in Eq (S4).
3. It is sufficient to show that $|\mathbf{q}_1^\top \mathbf{q}_2| = |(\mathbf{g} \circ \mathbf{q}_1)^\top (\mathbf{g} \circ \mathbf{q}_2)|$ for any $\mathbf{g} \in \mathbb{H}_1$:

$$|(\mathbf{g} \circ \mathbf{q}_1)^\top (\mathbf{g} \circ \mathbf{q}_2)| = |\mathbf{q}_1^\top \mathbf{G}^\top \mathbf{G} \mathbf{q}_2| \tag{S8}$$

$$= |\mathbf{q}_1^\top \mathbf{I} \mathbf{q}_2| \tag{S9}$$

$$= |\mathbf{q}_1^\top \mathbf{q}_2|, \tag{S10}$$

where $\mathbf{g} \circ \mathbf{q} \equiv \mathbf{G}\mathbf{q}$. The result is a direct consequence of the orthonormality of $\mathbf{G}$.

2 Proof of Lemma 1

We will begin by recalling some preliminary definitions and results that aid us to construct the connection between the dynamic routing and the Weiszfeld algorithm.

Definition 1 (Affine Subspace) *A d-dimensional affine subspace of R^N is obtained by a translation of a d-dimensional linear subspace $V \subset \mathbb{R}^N$ such that the origin is included in S:*

$$S = \Big\{ \sum_{i=1}^{d+1} \alpha_i \mathbf{x}_i \mid \sum_{i=1}^{d+1} \alpha_i = 1 \Big\}. \tag{S11}$$

Simplest choices for S involve points, lines and planes of the Euclidean space.

Definition 2 (Orthogonal Projection onto an Affine Subspace) *An orthogonal projection of a point $\mathbf{x} \in \mathbb{R}^N$ onto an affine subspace explained by the pair $(\mathbf{A}, \mathbf{c})$ is defined as:*

$$\mathit{\Pi}_i(\mathbf{x}) \triangleq proj_S(\mathbf{x}) = \mathbf{c} + \mathbf{A}(\mathbf{x} - \mathbf{c}). \tag{S12}$$

c *denotes the translation to make origin inclusive and* **A** *is a projection matrix typically defined via the orthonormal bases of the subspace.*

Definition 3 (Distance to Affine Subspaces) *Distance from a given point* **x** *to a set of affine subspaces $\{S_1, S_2 \dots S_k\}$ can be written as [3]:*

$$C(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, S_i) = \sum_{i=1}^{k} \|\mathbf{x} - proj_{S_i}(\mathbf{x})\|^2. \tag{S13}$$

Lemma S1. *Given that all the antipodal counterparts are mapped to the northern hemisphere, we will now think of the unit quaternion or versor as the unit normal of a four dimensional hyperplane h, passing through the origin:*

$$h_i(\mathbf{x}) = \mathbf{q}_i^\top \mathbf{x} + q_d := 0. \tag{S14}$$

q_d is an added term to compensate for the shift. When $q_d = 0$ the origin is incident to the hyperplane. With this perspective, quaternion $\mathbf{q}_i$ forms an affine subspace with $d = 4$, for which the projection operator takes the form:

$$proj_{S_i}(\mathbf{p}) = (\mathbf{I} - \mathbf{q}_i\mathbf{q}_i^\top)\mathbf{p} \tag{S15}$$

Proof. We consider Eq (S15) for the case where $\mathbf{c} = \mathbf{0}$ and $\mathbf{A} = (\mathbf{I} - \mathbf{q}\mathbf{q}^\top)$. The former follows from the fact that our subspaces by construction pass through the origin. Thus, we only need to show that the matrix $\mathbf{A} = \mathbf{I} - \mathbf{q}\mathbf{q}^\top$ is an orthogonal projection matrix onto the affine subspace spanned by $\mathbf{q}$. To this end, it is sufficient to validate that $\mathbf{A}$ is symmetric and idempotent: $\mathbf{A}^\top\mathbf{A} = \mathbf{A}\mathbf{A} = \mathbf{A}^2 = \mathbf{A}$. Note that by construction $\mathbf{q}^\top\mathbf{q}$ is a symmetric matrix and hence $\mathbf{A}$ itself. Using this property and the unit-ness of the quaternion, we arrive at the proof:

$$\begin{aligned}
\mathbf{A}^\top\mathbf{A} &= (\mathbf{I} - \mathbf{q}\mathbf{q}^\top)^\top(\mathbf{I} - \mathbf{q}\mathbf{q}^\top) && \text{(S16)}\\
&= (\mathbf{I} - \mathbf{q}\mathbf{q}^\top)(\mathbf{I} - \mathbf{q}\mathbf{q}^\top) && \text{(S17)}\\
&= \mathbf{I} - 2\mathbf{q}\mathbf{q}^\top + \mathbf{q}\mathbf{q}^\top\mathbf{q}\mathbf{q}^\top && \text{(S18)}\\
&= \mathbf{I} - 2\mathbf{q}\mathbf{q}^\top + \mathbf{q}\mathbf{q}^\top && \text{(S19)}\\
&= \mathbf{I} - \mathbf{q}\mathbf{q}^\top \triangleq \mathbf{A} && \text{(S20)}
\end{aligned}$$

It is easy to verify that the projections are orthogonal to the quaternion that defines the subspace by showing $\mathrm{proj}_S(\mathbf{q})^\top\mathbf{q} = 0$:

$$\mathbf{q}^\top\mathrm{proj}_S(\mathbf{q}) = \mathbf{q}^\top\mathbf{A}\mathbf{q} = \mathbf{q}^\top(\mathbf{I} - \mathbf{q}\mathbf{q}^\top)\mathbf{q} = \mathbf{q}^\top(\mathbf{q} - \mathbf{q}\mathbf{q}^\top\mathbf{q}) = \mathbf{q}^\top(\mathbf{q} - \mathbf{q}) = 0. \tag{S21}$$

Also note that this choice corresponds to $\mathrm{tr}(\mathbf{q}\mathbf{q}^\top) = \sum_{i=1}^{d+1}\alpha_i = 1$.

Lemma S2. *The quaternion mean we suggest to use in the main paper [4] is equivalent to the Euclidean Weiszfeld mean on the affine quaternion subspaces.*

Proof. We now recall and summarize the L_q-Weiszfeld Algorithm on affine subspaces [3], which minimizes a q-norm variant of the cost defined in Eq (S13):

$$C_q(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, S_i) = \sum_{i=1}^{k} \|\mathbf{x} - \mathrm{proj}_{S_i}(\mathbf{x})\|^q. \tag{S22}$$

Defining $\mathbf{M}_i = \mathbf{I} - \mathbf{A}_i$, Alg. 1 summarizes the iterative procedure.

Note that when $q = 2$, the algorithm reduces to the computation of a non-weighted mean ($w_i = 1\,\forall i$), and a closed form solution exists for Eq (S24) and is given by the normal equations:

$$\mathbf{x} = \Big(\sum_{i=1}^{k} w_i\mathbf{M}_i\Big)^{-1}\Big(\sum_{i=1}^{k} w_i\mathbf{M}_i\mathbf{c}_i\Big) \tag{S25}$$

Algorithm 1: L_q Weiszfeld Algorithm on Affine Subspaces [3].

input: An initial guess $\mathbf{x}_0$ that does not lie any of the subspaces $\{S_i\}$, Projection operators Π_i, the norm parameter q

$\mathbf{x}^t \leftarrow \mathbf{x}_0$

while *not converged* **do**

Compute the weights $\mathbf{w}^t = \{w_i^t\}$:

$$w_i^t = \|\mathbf{M}_i(\mathbf{x}^t - \mathbf{c}_i)\|^{q-2} \quad \forall i = 1 \dots k \tag{S23}$$

Solve:

$$\mathbf{x}^{t+1} = \underset{\mathbf{x}\in\mathbb{R}^N}{\arg\min} \sum_{i=1}^{k} w_i^t \|\mathbf{M}_i(\mathbf{x} - \mathbf{c}_i)\|^2 \tag{S24}$$

For the case of our quaternionic subspaces $\mathbf{c} = \mathbf{0}$ and we seek the solution that satisfies:

$$\Big(\sum_{i=1}^{k} \mathbf{M}_i\Big)\mathbf{x} = \Big(\frac{1}{k}\sum_{i=1}^{k} \mathbf{M}_i\Big)\mathbf{x} = \mathbf{0}. \tag{S26}$$

It is well known that the solution to this equation under the constraint $\|\mathbf{x}\| = 1$ lies in nullspace of $\mathbf{M} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{M}_i$ and can be obtained by taking the singular vector of $\mathbf{M}$ that corresponds to the largest singular value. Since $\mathbf{M}_i$ is idempotent, the same result can also be obtained through the eigendecomposition:

$$\mathbf{q}^\star = \underset{\mathbf{q}\in\mathcal{S}^3}{\arg\max}\, \mathbf{q}\mathbf{M}\mathbf{q} \tag{S27}$$

which gives us the unweighted Quaternion mean [4].

3 Proof of Theorem 1

Once the Lemma 1 is proven, we only need to apply the direct convergence results from the literature. Consider a set of points $\mathbf{Y} = \{\mathbf{y}_1 \dots \mathbf{y}_K\}$ where $K > 2$ and $\mathbf{y}_i \in \mathbb{H}_1$. Due to the compactness, we can speak of a ball $\mathcal{B}(\mathbf{o}, \rho)$ encapsulating all $\mathbf{y}_i$. We also define the $\mathcal{D} = \{\mathbf{x} \in \mathbb{H}_1 \,|\, C_q(\mathbf{x}) < C_q(\mathbf{o})\}$, the region where the loss decreases.

We first state the assumptions that permit our theoretical result. These assumptions are required by the works that establish the convergence of such Weiszfeld algorithms [1,2] :

H1. $\mathbf{y}_1 \dots \mathbf{y}_K$ should not lie on a single geodesic of the quaternion manifold.
H2. $\mathcal{D}$ is bounded and compact. The topological structure of $SO(3)$ imposes a

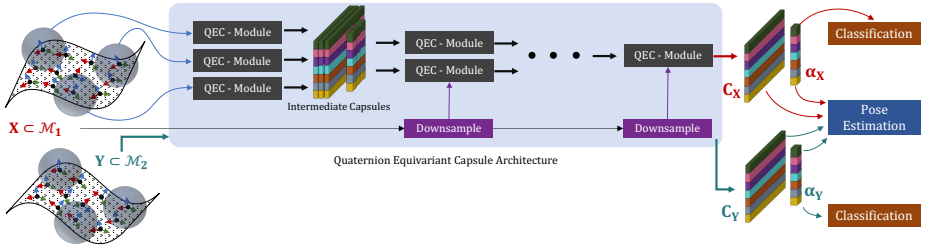


Fig. S1. Our siamese architecture used in the estimation of relative poses. We use a shared network to process two distinct point clouds $(\mathbf{X}, \mathbf{Y})$ to arrive at the latent representations $(\mathbf{C}_X, \boldsymbol{\alpha}_X)$ and $(\mathbf{C}_Y, \boldsymbol{\alpha}_Y)$ respectively. We then look for the highest activated capsules in both point sets and compute the rotation from the corresponding capsules. Thanks to the rotations disentangled into capsules, this final step simplifies to a relative quaternion calculation.

bounded convexity radius of $\rho < \pi/2$.
H3. The minimizer in Eq (S24) is continuous.
H4. The weighting function $\sigma(\cdot)$ is concave and differentiable.
H5. Initial quaternion (in our network chosen randomly) does not belong to any of the subspaces.

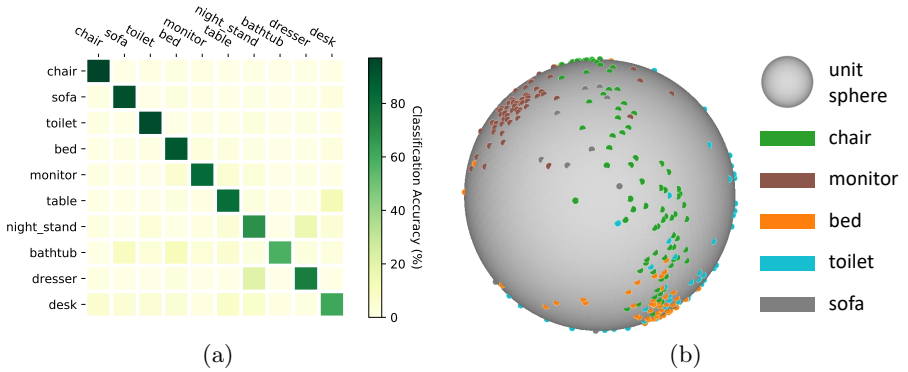
Note that **H5** is not a strict requirement as there are multiple ways to circumvent (simplest being a re-initialization). Under these assumptions, the sequence produced by Eq (S24) will converge to a critical point unless $\mathbf{x}^t = \mathbf{y}_i$ for any t and i [2]. For $q = 1$, this critical point is on one of the subspaces specified in Eq (S14) and thus is a geometric median. □

Note that due to the assumption **H2**, we cannot converge from any given point. For randomly initialized networks this is indeed a problem and does not guarantee practical convergence. Yet, in our experiments we have not observed any issue with the convergence of our dynamic routing. As our result is one of the few ones related to the analysis of DR, we still find this to be an important first step.

For different choices of $q : 1 \leq q \leq 2$, the weights take different forms. In fact, this IRLS type of algorithm is shown to converge for a larger class of weighting choices as long as the aforementioned conditions are met. That is why in practice we use a simple sigmoid function.

4 Further Discussions

On convergence, runtime and complexity. Note that while the convergence basin is known, to the best of our knowledge, a convergence rate for a Weiszfeld algorithm in affine subspaces is not established. From the literature of robust minimization via Riemannian gradient descent (this is essentially the corresponding particle optimizer), we conjecture that such a rate depends upon the choice of the convex regime (in this case $1 \leq q \leq 2$) and is at best linear – though we did not prove this conjecture. In practice we run the Weiszfeld



(a) (b)

Fig. S2. (a) Confusion matrix on ModelNet10 for classification. **(b)** Distribution of initial poses per class.

iteration only 3 times, similar to the original dynamic routing. This is at least sufficient to converge to a point good enough for the network to explain the data at hand.

QEC module summarized in the Alg. 2 of the main paper can be dissected into three main steps: (i) canonicalization of the local oriented point set, (ii) the t-kernel and (iii) dynamic routing. Overall the total computational complexity reads $O(L+KC_{MLP}+C_{DR})$ where C_{MLP} and C_{DR} are the computational costs of the MLP and the DR respectively:

$$\begin{aligned} C_{DR} &= LM + M(K + k(2L) + L) = M(K + 2(k+1)L) \\ C_{MLP} &= 64N_c + 4MN_c^2. \end{aligned} \tag{S28}$$

Note that Eq (S28) depicts the complexity of a single QEC module. In our architecture we use a stack of those each of which cause an added increase in the complexity proportional to the number of points downsampled.

Our weighted quaternion average relies upon a differentiable SVD. While not increasing the theoretical computational complexity, when done naively, this operation can cause significant increase in runtime. Hence, we compute the SVD using CUDA kernels in a batch-wise manner. This batch-wise SVD makes it possible to average a large amount of quaternions with high efficiency. Note that we omit the computational aspects of LRF calculation as we consider it to be an input to our system and different LRFs exhibit different costs.

We have further conducted a runtime analysis in the *3D Shape Classification* experiment on an Nvidia GeForce RTX 2080 Ti with the network configuration mentioned in Sec. 5.2 of the main paper. During training, each batch (where batch size $b = 8$) takes 0.226s and 1939M of GPU memory. During inference, processing each instance takes 0.036s and consumes 1107M of GPU memory.

Note that the use of LRFs helps us to restrict the rotation group to certain elements and thus we can use networks with significantly less parameters (as low as $0.44M$) compared to others as shown in Tab. 1 of the main paper. Num-

ber of parameters in our network depends upon the number of classes, e.g. for ModelNet10 we have $0.047M$ parameters.

Quaternion ambiguity. Quaternions of the northern and southern hemispheres represent the same exact rotation, hence one of them is *redundant*. By mapping one hemisphere to the other, we sacrifice the closeness of the manifold. This could slightly distort the behavior of the linearization operator around the Ecuador. However, the rest of the operations such as geodesic distances respect such antipodality, as we consider the Quaternionic manifold and not the sphere. When the subset of operations we develop and the nature of local reference frames are concerned, we did not find this transformation to cause serious shortcomings.

Performance on different shapes with same orientation. The NR/NR scenario in Tab. 1 of the main paper involves classification on different shapes within a category without rotation, *e.g.* chairs with different shapes. In addendum, we now provide in Fig. 2(b) an additional insight into the pose distribution for all canonicalized objects within a class. To do so, we rotate the horizontal standard basis vector $\mathbf{e}_x = [1, 0, 0]$ using the predict quaternion (the most activated output capsule) and plot the resulting point on a unit sphere as shown in Fig. 2(b). A qualitative observation reveals that for all five non-symmetric classes, the poses of all the instances within a class would form a cluster. This roughly holds across all classes and indicates that the relative pose information is consistent within the classes. On the other hand, objects with symmetries form multiple clusters.

5 Our Siamese Architecture

For estimation of the relative pose with supervision, we benefit from a Siamese variation of our network. In this case, latent capsule representations of two point sets $\mathbf{X}$ and $\mathbf{Y}$ jointly contribute to the pose regression as shown in Fig. S1.

We show additional results from the computation of local reference frames and the multi-channel capsules deduced from our network in Fig. S3.

6 Additional Details on Evaluations

Details on the evaluation protocol. For Modelnet40 dataset used in Tab. 1, we stick to the official split with 9,843 shapes for training and 2,468 different shapes for testing. For rotation estimation in Tab. 2, we again used the official Modelenet10 dataset split with 3991 for training and 908 shapes for testing. 3D point clouds (10K points) are randomly sampled from the mesh surfaces of each shape [5,6]. The objects in training and testing dataset are different, but they are from the same categories so that they can be oriented meaningfully. During training, we did not augment the dataset with random rotations. All the shapes are trained with single orientation (well-aligned). We call this *trained with NR*. During testing, we randomly generate multiple arbitrary $SO(3)$ rotations

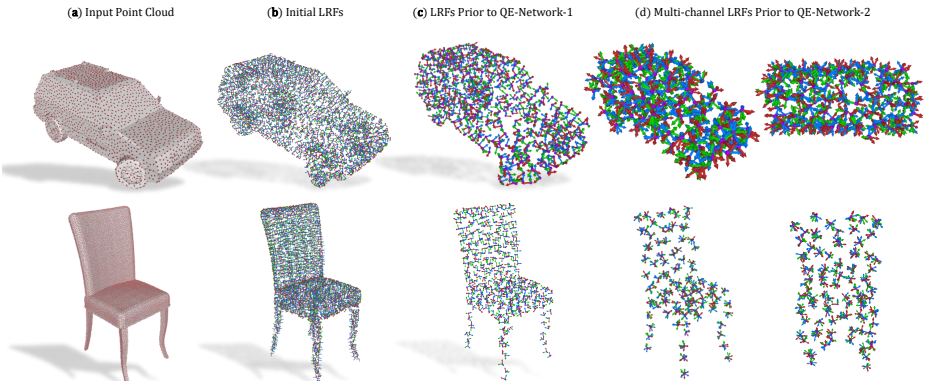


Fig. S3. Additional intermediate results on car (first row) and chair (second row) objects. This figure supplements Fig. 1(a) of the main paper.

for each shape and evaluate the average performance for all the rotations. This is called *test with AR*. This protocol is used in both our algorithms and the baselines.

Confusion of classification in ModelNet. To provide additional insight into how our activation features perform, we now report the confusion matrix in the task of classification on the all the objects of ModelNet10. Unique to our algorithm, the classification and rotation estimation reinforces one another. As seen from Fig. 2(a) on the right, the first five categories that exhibit less rotational symmetry has the higher classification accuracy than their rotationally symmetric counterparts.

Distribution of errors reported in Tab. 2. We now provide more details on the errors attained by our algorithm as well as the state of the art. To this end, we report, in Fig. S5 the histogram of errors that fall within quantized ranges of orientation errors. It is noticeable that our Siamese architecture behaves best in terms of estimating the objects rotation. For completeness, we also included the results of the variants presented in our ablation studies: Ours-2kLRF, Ours-1kLRF. They evaluate the model on the re-calculated LRFs in order to show the robustness towards to various point densities. We have also modified IT-Net and PointNetLK only to predict rotation because the original works predict both rotations and translations. Finally, note here that we do not use data augmentation for training our networks (see AR), while both for PointNetLK and for IT-Net we do use augmentation.

References

1. Aftab, K., Hartley, R.: Convergence of iteratively re-weighted least squares to robust m-estimators. In: Winter Conference on Applications of Computer Vision. IEEE (2015)

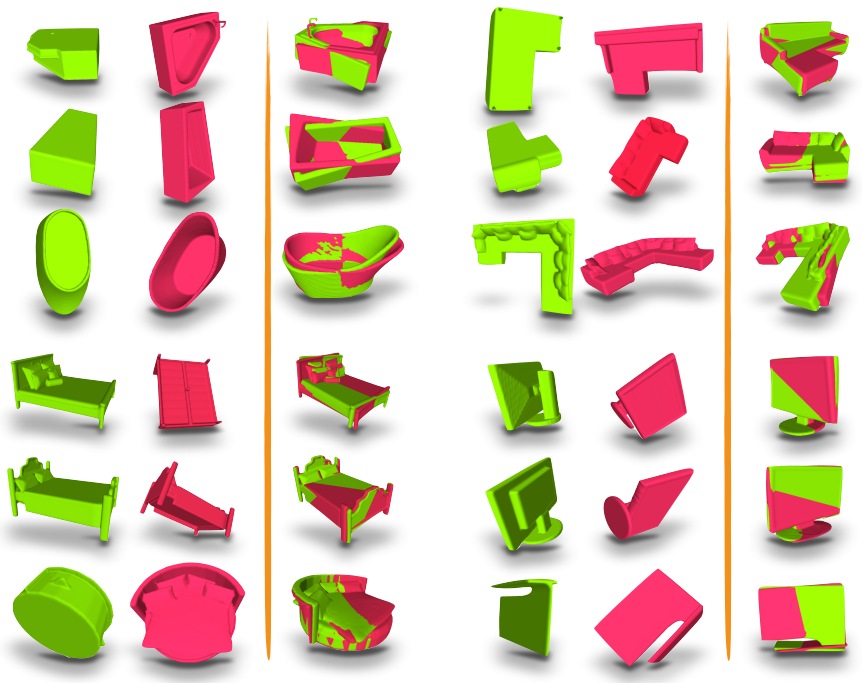
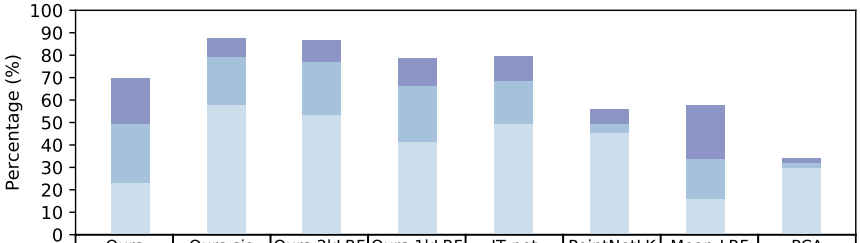


Fig. S4. Additional pairwise shape alignment on more categories in Modelnet10 dataset. We do not perform any ICP and the transformations that align the two point clouds are direct results of the forward pass of our Siamese network.

2. Aftab, K., Hartley, R., Trumpf, J.: Generalized weiszfeld algorithms for lq optimization. IEEE transactions on pattern analysis and machine intelligence **37**(4) (2014)
3. Aftab, K., Hartley, R., Trumpf, J.: l_q closest-point to affine subspaces using the generalized weiszfeld algorithm. International Journal of Computer Vision (2015)
4. Markley, F.L., Cheng, Y., Crassidis, J.L., Oshman, Y.: Averaging quaternions. Journal of Guidance, Control, and Dynamics **30**(4), 1193–1197 (2007)
5. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)
6. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)



	Ours	Ours-sia	Ours-2kLRF	Ours-1kLRF	IT-net	PointNetLK	Mean-LRF	PCA
< 15°	23.18%	58.16%	53.64%	41.36%	49.68%	45.40%	16.24%	29.80%
< 30°	49.53%	79.12%	76.88%	66.40%	68.40%	49.28%	33.84%	32.00%
< 60°	69.87%	87.48%	86.68%	78.76%	79.48%	55.84%	57.64%	33.80%

Fig. S5. Cumulative error histograms of rotation estimation on ModelNet10. Each row ($< \theta^\circ$) of this extended table shows the percentage of shapes that have rotation error less than θ. The colors of the bars correspond to the rows they reside in. The higher the errors are contained in the first bins (light blue) the better. Vice versa, the more the errors are clustered toward the 60° the worse the performance of the method.