# 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.** We present a 3D capsule module for processing point clouds that is equivariant to 3D rotations and translations, as well as invariant to permutations of the input points. The operator receives a sparse set of local reference frames, computed from an input point cloud and establishes end-to-end transformation equivariance through a novel dynamic routing procedure on quaternions. Further, we theoretically connect dynamic routing between capsules to the well-known Weiszfeld algorithm, a scheme for solving *iterative re-weighted least squares* (IRLS) problems with provable convergence properties. It is shown that such group dynamic routing can be interpreted as robust IRLS rotation averaging on capsule votes, where information is routed based on the final inlier scores. Based on our operator, we build a capsule network that disentangles geometry from pose, paving the way for more informative descriptors and a structured latent space. Our architecture allows joint object classification and orientation estimation without explicit supervision of rotations. We validate our algorithm empirically on common benchmark datasets. We release our sources under: https://tolgabirdal.github.io/qecnetworks.*

**Keywords:** 3D, equivariance, disentanglement, rotation, quaternion

## 1 Introduction

It is now well understood that in order to learn a compact and informative representation of the input data, one needs to respect the symmetries in the problem domain [17,73]. Arguably, one of the primary reasons for the success of 2D convolutional neural networks (CNN) is the *translation-invariance* of the 2D convolution acting on the image grid [29,36]. Recent trends aim to transfer this success into the 3D domain in order to support many applications such as shape retrieval, shape manipulation, pose estimation, 3D object modeling and detection, etc. There, the data is naturally represented as sets of 3D points [55,57]. Unfortunately, an extension of CNN architectures to 3D point clouds is non-trivial due to two reasons: 1) point clouds are irregular and unorganized, 2) the group of transformations that we are interested in is more complex as 3D data is often observed under arbitrary non-commutative $SO(3)$ rotations. As a result, learning appropriate embeddings requires 3D point-networks to be *equivariant* to these transformations, while also being invariant to point permutations.

*First two authors contributed equally to this work.

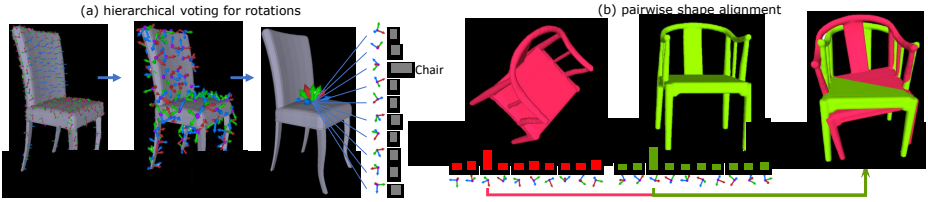



**Fig. 1.** (**a**) Our network operates on local reference frames (LRF) of an input point cloud (**i**). A hierarchy of quaternion equivariant capsule modules (QEC) then pools the LRFs to a set of latent capsules (**ii**, **iii**) disentangling the activations from poses. We can use activations in classification and the capsule (quaternion) with the highest activation in absolute (canonical) pose estimation without needing the supervision of rotations. (**b**) Our siamese variant can also solve for the relative object pose by aligning the capsules of two shapes with different point samplings. Our network directly consumes point sets and LRFs. Meshes are included only to ease understanding.

In order to fill this gap, we present a quaternion equivariant point capsule network that is suitable for processing point clouds and is equivariant to $SO(3)$ rotations, compactly parameterized by quaternions, while also preserving translation and permutation invariance. Inspired by the local group equivariance [40,17], we efficiently cover $SO(3)$ by restricting ourselves to a sparse set of local reference frames (LRFs) that collectively determine the object orientation. The proposed *quaternion equivariant capsule (QEC) module* deduces equivariant latent representations by robustly combining those LRFs using the proposed *Weiszfeld dynamic routing* with inlier scores as activations, so as to route information from one layer to the next. Hence, our latent features specify to local orientations and activations, disentangling orientation from evidence of object existence. Such explicit and factored storage of 3D information is unique to our work and allows us to perform rotation estimation jointly with object classification. Our final architecture is a hierarchy of QEC modules, where LRFs are routed from lower level to higher level capsules as shown in Fig. 1. We use classification error as the only training cue and adapt a Siamese version for regression of the relative rotations. We neither explicitly supervise the network with pose annotations nor train by augmenting rotations. In summary, our contributions are:

1. We propose a novel, fully $SO(3)$-equivariant capsule module that produces invariant latent representations while explicitly decoupling the orientation into capsules. Notably, equivariance results have not been previously achieved for $SO(3)$ capsule networks.
2. We connect dynamic routing between capsules [60] and generalized Weiszfeld iterations [4]. Based on this connection, we theoretically argue for the convergence of the included rotation estimation on votes and extend our understanding of dynamic routing approaches.
3. We propose a capsule network that is tailored for simultaneous classification and orientation estimation of 3D point clouds. We experimentally demonstrate the capabilities of our network on classification and orientation estimation on ModelNet10 and ModelNet40 3D shape data.

## 2 Related Work

***Deep learning on point sets.*** The capability to process raw, unordered point clouds within a neural network is introduced by the prosperous PointNet [55] thanks to the point-wise convolutions and the permutation invariant pooling functions. Many works have extended PointNet primarily to increase the local receptive field size [57,42,62,71]. Point-clouds are generally thought of as sets. This makes any permutation-invariant network that can operate on sets an amenable choice for processing points [81,58]. Unfortunately, common neural network operators in this category are solely equivariant to permutations and translations but to no other groups.

***Equivariance in neural networks.*** Early attempts to achieve invariant data representations usually involved data augmentation techniques to accomplish tolerance to input transformations [49,56,55]. Motivated by the difficulty associated with augmentation efforts and acknowledging the importance of theoretically equivariant or invariant representations, the recent years have witnessed a leap in theory and practice of equivariant neural networks [6,37].

While laying out the fundamentals of the group convolution, G-CNNs [18] guaranteed equivariance with respect to finite symmetry groups. Similarly, Steerable CNNs [21] and its extension to 3D voxels [75] considered discrete symmetries only. Other works opted for designing filters as a linear combination of harmonic basis functions, leading to frequency domain filters [76,74]. Apart from suffering from the dense coverage of the group using group convolution, filters living in the frequency space are less interpretable and less expressive than their spatial counterparts, as the basis does not span the full space of spatial filters.

Achieving equivariance in 3D is possible by simply generalizing the ideas of the 2D domain to 3D by voxelizing 3D data. However, methods using dense grids [16,21] suffer from increased storage costs, eventually rendering the implementations infeasible. An extensive line of work generalizes the harmonic basis filters to $SO(3)$ by using *e.g.* , a spherical harmonic basis instead of circular harmonics [19,25,22]. In addition to the same downsides as their 2D counterparts, these approaches have in common that they require their input to be projected to the unit sphere [33], which poses additional problems for unstructured point clouds. A related line of research are methods which define a regular structure on the sphere to propose equivariant convolution operators [44,13].

To learn a rotation equivariant representation of a 3D shape, one can either act on the input data or on the network. In the former case, one either presents augmented data to the network [55,49] or ensures rotation-invariance in the input [23,24,34]. In the latter case one can enforce equivariance in the bottleneck so as to achieve an invariant latent representation of the input [50,66,63]. Further, equivariant networks for discrete sets of views [27] and cross-domain views [26] have been proposed. Here, we aim for a different way of embedding equivariance in the network by means of an explicit latent rotation parametrization in addition to the invariant feature.

***Vector field networks*** [47] followed by the 3D *Tensor Field Networks* (TFN) [66] are closest to our work. Based upon a geometric algebra framework, the authors

did achieve localized filters that are equivariant to rotations, translations and permutations. Moreover, they are able to cover the continuous groups. However, TFN are designed for physics applications, are memory consuming and a typical implementation is neither likely to handle the datasets we consider nor can provide orientations in an explicit manner.

***Capsule networks.*** The idea of capsule networks was first mentioned by Hinton *et al.* [30], before Sabour *et al.* [60] proposed the *dynamic routing by agreement*, which started the recent line of work investigating the topic. Since then, routing by agreement has been connected to several well-known concepts, e.g. the EM algorithm [59], clustering with KL divergence regularization [68] and equivariance [40]. They have been extended to autoencoders [38] and GANs [32]. Further, capsule networks have been applied for specific kinds of input data, e.g. graphs [78], 3D point clouds [83,64] or medical images [1].

## 3 Preliminaries and Technical Background

We now provide the necessary background required for the grasp of the equivariance of point clouds under the action of quaternions.

### 3.1 Equivariance

**Definition 1 (Equivariant Map)** *For a $\mathcal{G}$-space acting on $\mathcal{X}$, the map $\Phi : \mathcal{G} \times \mathcal{X} \mapsto \mathcal{X}$ is said to be equivariant if its domain and co-domain are acted on by the same symmetry group [18,20]:*

$$\Phi(\mathbf{g}_1 \circ \mathbf{x}) = \mathbf{g}_2 \circ \Phi(\mathbf{x}) \tag{1}$$

*where $\mathbf{g}_1 \in \mathcal{G}$ and $\mathbf{g}_2 \in \mathcal{G}$. Equivalently $\Phi(T(\mathbf{g}_1)\,\mathbf{x}) = T(\mathbf{g}_2)\,\Phi(\mathbf{x})$, where $T(\cdot)$ is a linear representation of the group $\mathcal{G}$. Note that $T(\cdot)$ does not have to commute. It suffices for $T(\cdot)$ to be a homomorphism: $T(\mathbf{g}_1 \circ \mathbf{g}_2) = T(\mathbf{g}_1) \circ T(\mathbf{g}_2)$. In this paper we use a stricter form of equivariance and consider $\mathbf{g}_2 = \mathbf{g}_1$.*

**Definition 2 (Equivariant Network)** *An architecture or network is said to be equivariant if all of its layers are equivariant maps. Due to the transitivity of the equivariance, stacking up equivariant layers will result in globally equivariant networks e.g. , rotating the input will produce output vectors which are transformed by the same rotation [40,37].*

### 3.2 The Quaternion Group $\mathbb{H}_1$

The choice of 4-vector quaternions as representation for $SO(3)$ has multiple motivations: (1) All 3-vector formulations suffer from infinitely many singularities as angle goes to 0, whereas quaternions avoid those, (2) 3-vectors also suffer from infinitely many redundancies (the norm can grow indefinitely). Quaternions have a single redundancy: $q = -q$ that is in practice easy to enforce [9], (3) Computing the actual 'manifold mean' on the Lie algebra requires iterative techniques with subsequent updates on the tangent space. Such iterations are computationally and numerically harmful for a differentiable GPU implementation.

**Definition 3 (Quaternion)** *A* quaternion $\mathbf{q}$ *is an element of Hamilton algebra* $\mathbb{H}_1$*, extending the complex numbers with three imaginary units* $\boldsymbol{i}$*,* $\boldsymbol{j}$*,* $\boldsymbol{k}$ *in the form:* $\mathbf{q} = q_1\boldsymbol{1} + q_2\boldsymbol{i} + q_3\boldsymbol{j} + q_4\boldsymbol{k} = (q_1, q_2, q_3, q_4)^T$*, with* $(q_1, q_2, q_3, q_4)^T \in \mathbb{R}^4$ *and* $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -\boldsymbol{1}$*.* $q_1 \in \mathbb{R}$ *denotes the scalar part and* $\boldsymbol{v} = (q_2, q_3, q_4)^T \in \mathbb{R}^3$*, the vector part. The* conjugate $\bar{\mathbf{q}}$ *of the quaternion* $\mathbf{q}$ *is given by* $\bar{\mathbf{q}} := q_1 - q_2\boldsymbol{i} - q_3\boldsymbol{j} - q_4\boldsymbol{k}$*. A* unit quaternion $\mathbf{q} \in \mathbb{H}_1$ *with* $1 \stackrel{!}{=} \|\mathbf{q}\| := \mathbf{q} \cdot \bar{\mathbf{q}}$ *and* $\mathbf{q}^{-1} = \bar{\mathbf{q}}$*, gives a compact and numerically stable parametrization to represent orientation of objects on the unit sphere* $\mathcal{S}^3$*, avoiding gimbal lock and singularities [15]. Identifying antipodal points* $\mathbf{q}$ *and* $-\mathbf{q}$ *with the same element, the unit quaternions form a double covering group of* $SO(3)$*.* $\mathbb{H}_1$ *is closed under the non-commutative multiplication or the Hamilton product:*

$$(\mathbf{p} \in \mathbb{H}_1) \circ (\boldsymbol{r} \in \mathbb{H}_1) = [p_1 r_1 - \mathbf{v}_p \cdot \mathbf{v}_r \,;\, p_1\mathbf{v}_r + r_1\mathbf{v}_p + \mathbf{v}_p \times \mathbf{v}_r]. \tag{2}$$

**Definition 4 (Linear Representation of $\mathbb{H}_1$)** *We follow [12] and use the parallelizable nature of unit quaternions (*$d \in \{1, 2, 4, 8\}$ *where* $d$ *is the dimension of the ambient space) to define* $T : \mathbb{H}_1 \mapsto \mathbb{R}^{4\times4}$ *as:*

$$\mathbf{T}(\mathbf{q}) \triangleq \begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \\ q_2 & q_1 & -q_4 & q_3 \\ q_3 & q_4 & q_1 & -q_2 \\ q_4 & -q_3 & q_2 & q_1 \end{bmatrix}.$$

*To be concise we will use capital letters to refer to the matrix representation of quaternions e.g.* $\mathbf{Q} \equiv T(\mathbf{q})$*,* $\mathbf{G} \equiv T(\mathbf{g})$*. Note that* $T(\cdot)$*, the injective homomorphism to the orthonormal matrix ring, by construction satisfies the condition in Dfn. 1 [65]:* $\det(\mathbf{Q}) = 1, \mathbf{Q}^\top = \mathbf{Q}^{-1}, \|\mathbf{Q}\| = \|\mathbf{Q}_{i,:}\| = \|\mathbf{Q}_{:,i}\| = 1$ *and* $\mathbf{Q} - q_1\mathbf{I}$ *is skew symmetric:* $\mathbf{Q} + \mathbf{Q}^\top = 2q_1\mathbf{I}$*. It is easy to verify these properties.* $T$ *linearizes the Hamilton product or the group composition:* $\mathbf{g} \circ \mathbf{q} \triangleq T(\mathbf{g})\mathbf{q} \triangleq \mathbf{G}\mathbf{q}$*.*

### 3.3 3D Point Clouds

**Definition 5 (Point Cloud)** *We define a 3D surface to be a differentiable 2-manifold embedded in the ambient 3D Euclidean space:* $\mathcal{M}^2 \in \mathbb{R}^3$ *and a point cloud to be a discrete subset sampled on* $\mathcal{M}^2$*:* $\mathbf{X} \in \{\mathbf{x}_i \in \mathcal{M}^2 \cap \mathbb{R}^3\}$*.*

**Definition 6 (Local Geometry)** *For a smooth point cloud* $\{\mathbf{x}_i\} \in \mathcal{M}^2 \subset \mathbb{R}^{N\times3}$*, a* local reference frame (LRF) *is defined as an ordered basis of the tangent space at* $\mathbf{x}$*,* $\mathcal{T}_\mathbf{x}\mathcal{M}$*, consisting of orthonormal vectors:* $\mathcal{L}(\mathbf{x}) = [\boldsymbol{\partial}_1, \boldsymbol{\partial}_2, \boldsymbol{\partial}_3 \equiv \boldsymbol{\partial}_1 \times \boldsymbol{\partial}_2]$*. Usually the first component is defined to be the surface normal* $\boldsymbol{\partial}_1 \triangleq \mathbf{n} \in \mathcal{S}^2 : \|\mathbf{n}\| = 1$ *and the second one is picked according to a heuristic.*

Note that recent trends, *e.g.* as in Cohen *et al.* [17], acknowledge the ambiguity and either employ a *gauge* (tangent frame) equivariant design or propagate the determination of a certain direction until the last layer [54]. Here, we will assume that $\boldsymbol{\partial}_2$ can be uniquely and repeatably computed, a reasonable assumption for the point sets we consider [52]. For the cases where this does not hold, we will rely on the robustness of the iterative routing procedures in our network. We will explain our method of choice in Sec. 6 and visualize LRFs of an airplane object in Fig. 1.

## 4 $SO(3)$-Equivariant Dynamic Routing

Disentangling orientation from representations requires guaranteed equivariances and invariances. Yet, the original capsule networks of Sabour *et al.* [60] cannot achieve equivariance to general groups. To this end, Lenssen *et al.* [40] proposed a dynamic routing procedure that guarantees equivariance and invariance under $SO(2)$ actions, by applying a manifold-mean and the geodesic distance as routing operators. We will extend this idea to the non-abelian $SO(3)$ and design capsule networks that sparsely operate on a set of LRFs computed via [53] on local neighborhoods of points. The $SO(3)$ elements are paremeterized by quaternions similar to [82]. In the following, we begin by introducing our novel equivariant dynamic routing procedure, the main building block of our architecture. We show the connection to the well known Weiszfeld algorithm, broadening the understanding of dynamic routing by embedding it into traditional computer vision methodology. Then, we present an example of how to stack those layers via a simple aggregation, resulting in an $SO(3)$-equivariant 3D capsule network that yields invariant representations (or activations) as well as equivariant orientations (latent capsules).

### 4.1 Equivariant Quaternion Mean

To construct equivariant layers on the group of rotations, we are required to define a left-equivariant averaging operator $\mathcal{A}$ that is invariant under permutations of the group elements, as well as a distance metric $\delta$ that remains unchanged under the action of the group [40]. For these, we make the following choices:

**Definition 7 (Geodesic Distance)** *The Riemannian (geodesic) distance on the manifold of rotations lead to the following geodesic distance* $\delta(\cdot) \equiv d_{quat}(\cdot)$*:*

$$d(\mathbf{q}_1, \mathbf{q}_2) \equiv d_{quat}(\mathbf{q}_1, \mathbf{q}_2) = 2\cos^{-1}(|\langle \mathbf{q}_1, \mathbf{q}_2 \rangle|) \tag{3}$$

**Definition 8 (Quaternion Mean $\boldsymbol{\mu}(\cdot)$)** *For a set of* $Q$ *rotations* $\mathbf{S} = \{\mathbf{q}_i\}$ *and associated weights* $\mathbf{w} = \{w_i\}$*, the weighted mean operator* $\mathcal{A}(\mathbf{S}, \mathbf{w}) : {\mathbb{H}_1}^n \times \mathbb{R}^n \mapsto {\mathbb{H}_1}^n$ *is defined through the following maximization procedure [48]:*

$$\bar{\mathbf{q}} = \underset{\mathbf{q} \in \mathbb{S}^3}{\arg\max}\, \mathbf{q}^\top \mathbf{M} \mathbf{q} \tag{4}$$

*where* $\mathbf{M} \in \mathbb{R}^{4\times 4}$ *is defined as:* $\mathbf{M} \triangleq \sum_{i=1}^{Q} w_i \mathbf{q}_i \mathbf{q}_i^\top$.

The average quaternion $\bar{\mathbf{q}}$ is the eigenvector of $\mathbf{M}$ corresponding to the maximum eigenvalue. This operation lends itself to both analytic [46] and automatic differentiation [39]. The following properties allow $\mathcal{A}(\mathbf{S}, \mathbf{w})$ to be used to build an equivariant dynamic routing:

**Theorem 1** *Quaternions, the employed mean* $\mathcal{A}(\mathbf{S}, \mathbf{w})$ *and geodesic distance* $\delta(\cdot)$ *enjoy the following properties:*

**Algorithm 1:** Quaternion Equivariant Dynamic Routing

**input :** Input points $\{\mathbf{x}_1, ..., \mathbf{x}_K\} \in \mathbb{R}^{K\times 3}$, input capsules (LRFs) $\mathcal{Q} = \{\mathbf{q}_1, \dots, \mathbf{q}_L\} \in {\mathbb{H}_1}^L$, with $L = N^c \cdot K$, $N^c$ is the number of capsules per point, activations $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_L)^T$, trainable transformations $\mathcal{T} = \{\mathbf{t}_{i,j}\}_{i,j} \in {\mathbb{H}_1}^{L\times M}$

**output:** Updated frames $\hat{\mathcal{Q}} = \{\hat{\mathbf{q}}_1, \dots, \hat{\mathbf{q}}_M\} \in {\mathbb{H}_1}^M$, updated activations $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \dots, \hat{\alpha}_M)^T$

**for** *All primary (input) capsules* $i$ **do**
  **for** *All latent (output) capsules* $j$ **do**
    $\mathbf{v}_{i,j} \leftarrow \mathbf{q}_i \circ \mathbf{t}_{i,j}$ `// compute votes`
**for** *All latent (output) capsules* $j$ **do**
  $\hat{\mathbf{q}}_j \leftarrow \mathcal{A}\big(\{\mathbf{v}_{1,j} \dots \mathbf{v}_{K,j}\}, \boldsymbol{\alpha}\big)$ `// initialize output capsules`
  **for** $k$ *iterations* **do**
    **for** *All primary (input) capsules* $i$ **do**
      $w_{i,j} \leftarrow \alpha_i \cdot \text{sigmoid}\big(-\delta(\hat{\mathbf{q}}_j, \mathbf{v}_{i,j})\big)$ `// the current weight`
    $\hat{\mathbf{q}}_j \leftarrow \mathcal{A}\big(\{\mathbf{v}_{1,j} \dots \mathbf{v}_{L,j}\}, \mathbf{w}_{:,j}\big)$ `// see Eq (4)`
  $\hat{\alpha}_j \leftarrow \text{sigmoid}\big(-\frac{1}{K}\sum_1^L \delta(\hat{\mathbf{q}}_j, \mathbf{v}_{i,j})\big)$ `// recompute activations`

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:*

$$\mathcal{A}(\{\mathbf{q}_1, \dots, \mathbf{q}_Q\}, \mathbf{w}) = \mathcal{A}(\{\mathbf{q}_{\sigma(1)}, \dots, \mathbf{q}_{\sigma(Q)}\}, \mathbf{w}_\sigma). \tag{5}$$

3. *The transformations* $\mathbf{g} \in \mathbb{H}_1$ *preserve the geodesic distance* $\delta(\cdot)$ *given in Dfn. 7.*

*Proof.* The proofs are given in the supplementary material.

We also note that the above mean is closed form, differentiable and can be computed in a batch-wise fashion. We are now ready to construct the *dynamic routing* (DR) by agreement that is equivariant to $SO(3)$ actions, thanks to Thm. 1.

### 4.2 Equivariant Weiszfeld Dynamic Routing

Our routing procedure extends previous work [60,40] for quaternion valued input. The core idea is to *route* from the *primary capsules* that constitute the input LRF set to the *latent capsules* by an iterative clustering of votes $\mathbf{v}_{i,j}$. At each step, we assign the weighted group mean of votes to the respective output capsules. The weights $w \leftarrow \sigma(\mathbf{x}, \mathbf{y})$ are inversely propotional to the distance between the vote quaternions and the new quaternion (cluster center). See Alg. 1 for details. In the following, we analyze our variant of routing as an interesting case of the affine, Riemannian Weiszfeld algorithm [4,3].

**Lemma 1.** *For* $\sigma(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x}, \mathbf{y})^{q-2}$ *the equivariant routing procedure given in Alg. 1 is a variant of the affine subspace Wieszfeld algorithm [4,3] that is a robust algorithm for computing the* $L_q$ *geometric median.*

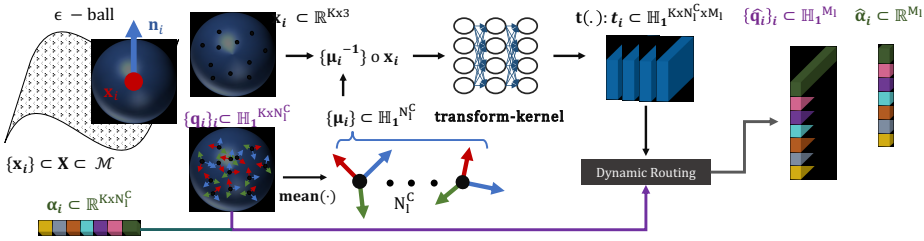



**Fig. 2.** Our **q**uaternion **e**quivariant **c**apsule (QEC) layer for processing local patches: Our input is a 3D point set $\mathbf{X}$ on which we query local neighborhoods $\{\mathbf{x}_i\}$ with precomputed LRFs $\{\mathbf{q}_i\}$. Essentially, we learn the parameters of a fully connected network that continuously maps the canonicalized local point set to transformations $\mathbf{t}_i$, which are used to compute hypotheses (votes) from input capsules. By a special dynamic routing procedure that uses the activations determined in a previous layer, we arrive at latent capsules that are composed of a set of orientations $\hat{\mathbf{q}}_i$ and new activations $\hat{\boldsymbol{\alpha}}_i$. Thanks to the decoupling of local reference frames, $\hat{\boldsymbol{\alpha}}_i$ is invariant and orientations $\hat{\mathbf{q}}_i$ are equivariant to input rotations. All the operations and hence the entire QE-network are equivariant achieving a guaranteed disentanglement of the rotation parameters. *Hat symbol ($\hat{\mathbf{q}}$) refers to 'estimated'.*

*Proof (Proof Sketch).* The proof follows from the definition of Weiszfeld iteration [3] and the mean and distance operators defined in Sec. 4.1. We first show that computing the weighted mean is equivalent to solving the normal equations in the iteratively reweighted least squares (IRLS) scheme [14]. Then, the inner-most loop corresponds to the IRLS or Weiszfeld iterations. We provide the detailed proof in supplementary material.

Note that, in practice one is quite free to choose the weighting function $\sigma(\cdot)$ as long as it is inversely proportional to the geodesic distance and concave [2]. The original dynamic routing can also be formulated as a clustering procedure with a KL divergence regularization. This holistic view paves the way to better routing algorithms [68]. Our perspective is akin yet more geometric due to the group structure of the parameter space. Thanks to the connection to Weiszfeld algorithm, the convergence behavior of our dynamic routing can be directly analyzed within the theoretical framework presented by [3,4].

**Theorem 2** *Under mild assumptions provided in the appendix, the sequence of the DR-iterates generated by the inner-most loop almost surely converges to a critical point.*

*Proof (Proof Sketch).* Proof, given in the appendix, is a direct consequence of Lemma 1 and directly exploits the connection to the Weiszfeld algorithm.

In summary, the provided theorems show that our dynamic routing by agreement is in fact a variant of robust IRLS rotation averaging on the predicted votes, where refined inlier scores for combinations of input/output capsules are used to route information from one layer to the next.

**Algorithm 2:** Quaternion Equivariant Capsule Module

**input :** Input points of one patch $\{\mathbf{x}_1, ..., \mathbf{x}_K\} \in \mathbb{R}^{K\times 3}$, input capsules (LRFs) $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_L\} \in {\mathbb{H}_1}^L$, with $L = N^c \cdot K$, $N^c$ is the number of capsules per point, activations $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_L)^T$

**output:** Updated frames $\hat{\mathcal{Q}} = \{\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_M\} \in {\mathbb{H}_1}^M$, updated activations $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_M)^T$

**for** *Each input channel* $n^c$ *of all the primary capsules channels* $N^c$ **do**
  $\mu(n^c) \leftarrow \mathcal{A}(\mathcal{Q}(n^c))$ // Input quaternion average, see Eq (4)
  **for** *Each point* $\mathbf{x}_i$ *of this patch* **do**
    $\mathbf{x}'_i \leftarrow \mu(n^c)^{-1} \circ \mathbf{x}_i$ // Rotate to a canonical orientation

$\{\mathbf{x}'_i\} \in \mathbb{R}^{K\times N^c\times 3}$ // Points in multiple($N^c$) canonical frames

**for** *Each point* $\mathbf{x}'_i$ *of this patch* **do**
  $\mathbf{t} \leftarrow t(\mathbf{x}'_i)$ // Transform kernel, $t(\cdot) : \mathbb{R}^{N^c\times 3} \rightarrow \mathbb{R}^{N^c\times M\times 4}$

$\mathcal{T} \equiv \{\mathbf{t}_i\} \in {\mathbb{H}_1}^{K\times N_i^c\times M} \leftarrow \{\mathbf{t}\} \in {\mathbb{H}_1}^{L\times M}$

$(\hat{\mathcal{Q}}, \hat{\boldsymbol{\alpha}}) \leftarrow \text{DynamicRouting}(X, \mathcal{Q}, \boldsymbol{\alpha}, \mathcal{T})$ // See Alg. 1

## 5 Equivariant Capsule Network Architecture

In the following, we describe how we leverage the novel dynamic routing algorithm to build a capsule network for point cloud processing that is equivariant under $SO(3)$ actions on the input. The essential ingredient of our architecture, the *quaternion equivariant capsule (QEC) module* that implements a capsule layer with dynamic routing, is described in Sec. 5.1, before using it as building block in the full architecture, as described in Sec. 5.2.

### 5.1 QEC Module

The main module of our architecture, the QEC module, is outlined in Fig. 2. We also provide the corresponding pseudocode in Alg. 2.

***Input.*** The input to the module is a local patch of points with coordinates $\mathbf{x}_i \subset \mathbb{R}^{K\times 3}$, rotations (LRFs) attached to these points, parametrized as quaternions $\mathbf{q}_i \subset {\mathbb{H}_1}^{K\times N^c}$ and activations $\boldsymbol{\alpha}_i \subset \mathbb{R}^{K\times N^c}$. We also use $\mathbf{q}_i$ to denote the input capsules. $N^c$ is the number of input capsule channels per point and it is equal to the number of output capsules ($M$) from the last layer.

***Trainable transformations.*** Recalling the original capsule networks of Sabour *et al.* [60], the trainable transformations $\mathbf{t}$, which are applied to the input rotations to compute the votes, lie in a grid kernel in the 2D image domain. Therefore, the procedure can learn to produce well-aligned votes if and only if the learned patterns in $\mathbf{t}$ match those in input capsule sets (agreement on evidence of object existence). Since our input points in the local receptive field lie in continuous $\mathbb{R}^3$, training a discrete set of pose transformations $\mathbf{t}_{i,j}$ based on discrete local coordinates is not possible. Instead, we use a similar approach as

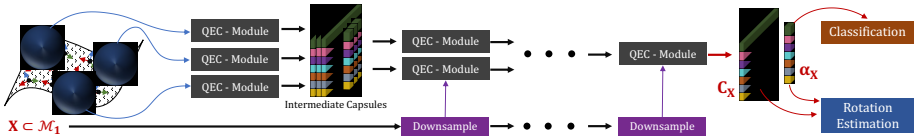



**Fig. 3.** Our entire capsule-network architecture. We hierarchically send all the local patches to our QEC-module as shown in Fig. 2. At each level the points are pooled in order to increase the receptive field, gradually reducing the LRFs into a single capsule per class. We use classification and orientation estimation (in the siamese case) as supervision cues to train the transform-kernels $\mathbf{t}(\cdot)$.

Lenssen *et al.* [40] and employ a continuous kernel $t(\cdot) : \mathbb{R}^{N^c \times 3} \rightarrow \mathbb{R}^{M \times N^c \times 4}$ that is defined on the continuous $\mathbb{R}^{N^c \times 3}$, instead of only a discrete set of positions. The network is shared over all points to compute the transformations $\mathbf{t}_{i,j} = (t(\mathbf{x}'_1), ..., t(\mathbf{x}'_K))_{i,j} \subset \mathbb{R}^{K \times M \times N^c \times 4}$, which are used to calculate the votes for dynamic routing with $\mathbf{v}_{i,j} = \mathbf{q}_i \circ \mathbf{t}_{i,j}$. The network $t(\cdot)$ consists of fully-connected layers that regresses the transformations, similar to common operators for continuous convolutions [61,70,28], just with quaternion output. The kernel is able to learn pose patterns in the 3D space, which align the resulting votes if certain pose sets are present. Note that $t(\cdot)$ predicts quaternions by unit-normalizing the regressed output: $\mathbf{t}_{i,j} \subset {\mathbb{H}_1}^{K \times M \times N^c}$. Although Riemannian layers [7] or spherical predictions [43] can improve the performance, the simple strategy works reasonably for our case.

In order for the kernel to be invariant, it needs to be aligned using an equivariant initial orientation candidate [40]. Given points $\mathbf{x}_i$ and rotations $\mathbf{q}_i$, we compute the mean $\boldsymbol{\mu}_i$ in a channel-wise manner like that of the initial candidates: $\boldsymbol{\mu}_i \subset {\mathbb{H}_1}^{N^c}$. These candidates are used to bring the kernels in canonical orientations by inversely rotating the input points: $\mathbf{x}'_i = ({\boldsymbol{\mu}_i}^{-1} \circ \mathbf{x}_i) \subset \mathbb{R}^{K \times N^c \times 3}$.

***Computing the output.*** After computing the votes, we utilize the input activation $\boldsymbol{\alpha}_i$ as initialization weights and iteratively refine the output capsule rotations (robust rotation estimation on votes) $\hat{\mathbf{q}}_i$ and activations $\hat{\boldsymbol{\alpha}}_i$ (final inlier scores) by our Weiszfeld routing by agreement as shown in Alg. 1.

### 5.2 Network Architecture

For processing point clouds, we use multiple QEC modules in a hierarchical architecture as shown in Fig. 3. In the first layer, the input primary capsules are represented by LRFs computed with FLARE algorithm [53]. Therefore, the number of input capsule channels $N^c$ in the first layer is equal to 1 and activations are uniform. The output of a former layer is propagated to the input of the latter, creating the hierarchy.

In order to gradually increase the receptive field, we stack QEC modules creating a deep hierarchy, where each layer reduces the number of points and increases the receptive field. In our experiments, we use a two level architecture,

**Table 1.** Classification accuracy on ModelNet40 dataset [77] for different methods as well as ours. We also report the number of parameters optimized for each method. **X**/**Y** means that we train with **X** and test with **Y**.

| | PN | PN++ | DGCNN | KDTreeNet | Point2Seq | Sph.CNNs | PRIN | PPF | Ours (Var.) | Ours |
|---|---|---|---|---|---|---|---|---|---|---|
| **NR/NR** | 88.45 | 89.82 | **92.90** | 86.20 | 92.60 | - | 80.13 | 70.16 | 85.27 | 74.43 |
| **NR/AR** | 12.47 | 21.35 | 29.74 | 8.49 | 10.53 | 43.92 | 68.85 | 70.16 | 11.75 | **74.07** |
| **#Params** | 3.5M | 1.5M | 2.8M | 3.6M | 1.8M | 0.5M | 1.5M | 3.5M | 0.4M | **0.4M** |

which receives $N = 64$ patches as input. We call the centers of these patches *pooling centers* and compute them via a uniform farthest point sampling as in [11]. Pooling centers serve as the positions of output capsules of the current layer. Each of those centers is linked to their immediate vicinity leading to $K = 9$-star local connectivity from which serve as input to the first QEC module to compute rotations and activations of $64 \times 64 \times 4$ intermediate capsules. The second module connects those intermediate capsules to the output capsules, whose number corresponds to the number of classes. Specifically, for layer 1, we use $K = 9, N_l{}^c = 1, M_l = 64$ and for layer 2, $K = 64, N_l{}^c = 64, M_l = C = 40$. This way, the last QEC module receives only one input patch and pools all capsules into a single point with an estimated LRF. For further details, we refer to our source code, which we will make available online before publication and provide in the supplemental materials.

## 6 Experimental Evaluations

***Implementation details.*** We implement our network in PyTorch and use the ADAM optimizer [35] with a learning rate of 0.001. Our point-transformation mapping network (transform-kernel) is implemented by two FC-layers composed of 64 hidden units. We set the initial activation of the input LRF to 1.0. In each layer, we use 3 iterations of DR. For classification we use the spread loss [59] and the rotation loss is identical to $\delta(\cdot)$.

The first axis of the LRF is the surface normal computed by local plane fits [31]. We compute the second axis, $\boldsymbol{\partial}_2$, by FLARE [53], that uses the normalized projection of the point with the largest distance within the periphery of the support, onto the tangent plane of the center: $\boldsymbol{\partial}_2 = \frac{\mathbf{p}_{\max} - \mathbf{p}}{\|\mathbf{p}_{\max} - \mathbf{p}\|}$. Using other choices such as SHOT [67] or GFrames [51] is possible. We found FLARE to be sufficient for our experiments. Prior to all operations, we flip all the LRF quaternions such that they lie on the northern hemisphere : $\{\mathbf{q}_i \in \mathbb{S}^3 : q_i^w > 0\}$.

***3D shape classification.*** We use ModelNet40 dataset of [77,57] to assess our classification performance where each shape is composed of $10K$ points randomly sampled from the mesh surfaces of each shape [55,57]. We use the official split with 9,843 shapes for training and 2,468 for testing. We assign the LRFs to a subset of the uniformly sampled points, $N = 512$ [11].

During training, we do not augment the dataset with random rotations. All the shapes are trained with single orientation (well-aligned). We call this *trained*

**Table 2.** Relative angular error (RAE) of rotation estimation in different categories of ModelNet10. Right side of the table denotes the objects with rotational symmetry, which we include for completeness. PCA-S refers to running PCA only on a resampled instance, while PCA-SR applies both rotations and resampling.

| Method | Avg. | No_Sym | Chair | Bed | Sofa | Toilet | Monitor | Table | Desk | Dresser | NS | Bathtub |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mean LRF | 0.41 | 0.35 | 0.32 | 0.36 | 0.34 | 0.41 | 0.34 | 0.45 | 0.60 | 0.50 | 0.46 | 0.32 |
| PCA-S | 0.40 | 0.42 | 0.60 | 0.53 | 0.46 | 0.32 | 0.12 | 0.47 | **0.23** | **0.33** | 0.43 | 0.55 |
| PCA-SR | 0.67 | 0.67 | 0.69 | 0.70 | 0.67 | 0.68 | 0.61 | 0.67 | 0.67 | 0.67 | 0.66 | 0.70 |
| PointNetLK [5] | 0.37 | 0.38 | 0.43 | 0.31 | 0.40 | 0.40 | 0.31 | 0.40 | 0.33 | 0.39 | 0.38 | 0.34 |
| IT-Net [80] | 0.27 | 0.19 | 0.10 | 0.22 | 0.17 | 0.20 | 0.28 | **0.31** | 0.41 | 0.44 | 0.40 | 0.39 |
| Ours | 0.27 | 0.17 | 0.11 | 0.20 | 0.16 | 0.18 | 0.19 | 0.43 | 0.40 | 0.48 | 0.33 | 0.31 |
| Ours (siamese) | **0.20** | **0.09** | **0.08** | **0.10** | **0.08** | **0.11** | **0.08** | 0.40 | 0.35 | 0.34 | **0.32** | **0.30** |

*with NR*. During testing, we randomly generate multiple arbitrary $SO(3)$ rotations for each shape and evaluate the average performance for all the rotations. This is called *test with AR*. This protocol is similar to [5]'s and is used both for our algorithms and for the baselines. Our results are shown in Tab. 1 along with that of PointNet (PN) [55], PointNet++ (PN++) [55], DGCNN [72], KD-treeNet [41], Point2Seq [45], Spherical CNNs [25], PRIN [79] and the theoretically invariant PPF-FoldNet (PPF) [23]. We also present a version of our algorithm (*Var*) that avoids the canonicalization within the QE-network. This is a non-equivariant network that we still train without data augmentation or orientation supervision. While this version gets comparable results to the state of the art for the NR/NR case, it cannot handle random $SO(3)$ variations (AR). Note that PPF uses the point-pair-feature [10] encoding and hence creates invariant input representations. For the scenario of NR/AR, our equivariant version outperforms all the other methods, including equivariant spherical CNNs [25] by a significant gap of at least 5% even when [25] exploits the 3D mesh. The object rotational symmetries in this dataset are responsible for a significant portion of the errors we make and we provide further details in supplementary material. It is worth mentioning that we also trained TFNs [66] for that task, but their memory demand made it infeasible to scale to this application.

***Computational aspects.*** As shown in Tab. 1 for ModelNet40 our network has $0.047M$ parameters. It incurs a computational cost in the order $O(MKL)$. The details are given in the supplementary material.

***Rotation estimation in 3D point clouds.*** Our network can estimate both the absolute and relative 3D object rotations without pose-supervision. To evaluate this desired property, we used the well classified shapes on ModelNet10 dataset, a sub-dataset of Modelnet40 [77]. This time, we use the official Modelenet10 dataset split with 3991 for training and 908 shapes for testing.

During testing, we generate multiple instances per shape by transforming the instance with five arbitrary $SO(3)$ rotations. As we are also affected by the sampling of the point cloud, we resample the mesh five times and generate different pooling graphs across all the instances of the same shape. Our QE-architecture can estimate the pose in two ways: 1) *canonical*: by directly using the

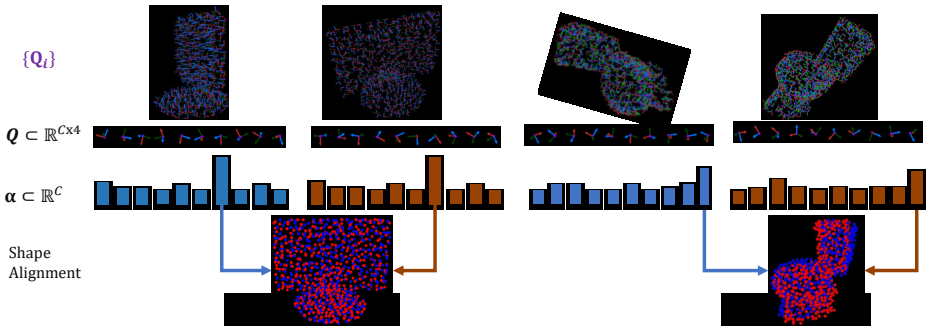



**Fig. 4.** Shape alignment on the **monitor** (left) and **toilet** (right) objects via our siamese equivariant capsule architecture. The shapes are assigned to the the maximally activated class. The corresponding pose capsule provides the rotation estimate.

output capsule with the highest activation, 2) *siamese*: by a siamese architecture that computes the relative quaternion between the capsules that are maximally activated as shown in Fig. 4. Both modes of operation are free of the data augmentation and we give further schematics of the latter in our appendix. It is worth mentioning that unlike regular pose estimation algorithms which utilize the same shape in both training and testing, our network never sees the test shapes during training. This is also known as *category-level* pose estimation [69].

Our results against the baselines including a naive averaging of the LRFs (Mean LRF) and principal axis alignment (PCA) are reported in Tab. 2 as the relative angular error (RAE). We further include results of PointNetLK [5] and IT-Net [80], two state of the art 3D networks that iteratively align two given point sets. These methods are in nature similar to iterative closest point (ICP) algorithm [8] but 1) do not require an initialization (*e.g.* first iteration estimates the pose), 2) learn data driven updates. Methods that use mesh inputs such as Spherical CNNs [25] cannot be included here as the random sampling of the same surface would not affect those. We also avoid methods that are just invariant to rotations (and hence cannot estimate the pose) such as Tensorfield Networks [66]. Finally, note that , IT-net [80] and PointNetLK [5] need to train for a lot of epochs (*e.g.* 500) with random $SO(3)$ rotation augmentation in order to obtain models with full coverage of $SO(3)$, whereas we train only for $\sim 100$ epochs. Finally, the recent geometric capsule networks [64] remains similar to PCA with an RAE of 0.42 on No_Sym when evaluated under identical settings. We include more details about the baselines in the appendix.

Relative Angle in Degrees (RAE) between the ground truth and the prediction is computed as: $d(\mathbf{q}_1, \mathbf{q}_2)/\pi$. Note that resampling and random rotations render the job of all methods difficult. However, both of our canonical and siamese versions which try to find a canonical and a relative alignment respectively, are better than the baselines. As pose estimation of objects with rotational symmetry is a challenging task due to inherent ambiguities, we also report results on the non-symmetric subset (No_Sym).

**Table 3.** Ablation study on point density.

| LRF Input | LRF-10K | | | | LRF-2K | LRF-1K |
|---|---|---|---|---|---|---|
| Dropout | 50% | 66% | 75% | 100% | 100% | 100% |
| Classification Accuracy | 77.8 | 83.3 | 83.4 | 87.8 | 85.46 | 79.74 |
| Angular Error | 0.34 | 0.27 | 0.25 | 0.09 | 0.10 | 0.12 |

***Robustness against point and LRF resampling.*** Density changes in the local neighborhoods of the shape are an important cause of error for our network. Hence, we ablate by applying random resampling (patch-wise dropout) objects in ModelNet10 dataset and repeating the classification and pose estimation as described above. While we use all the classes in classification accuracy, we only consider the well classified non-symmetric (No_Sym) objects for ablating on the pose estimation. The first part (LRF-10K) of Tab. 3 shows our findings against gradual increases of the number of patches. Here, we sample 2K LRFs from the 10K LRFs computed on an input point set of cardinality 10K. 100% dropout corresponds to 2K points in all columns. On second ablation, we reduce the amount of points on which we compute the LRFs, to 2K and 1K respectively. As we can see from the table, our network is robust towards the changes in the LRFs as well as the density of the points.

## 7 Conclusion and Discussion

We have presented a new framework for achieving permutation invariant and $SO(3)$ equivariant representations on 3D point clouds. Proposing a variant of the capsule networks, we operate on a sparse set of rotations specified by the input LRFs thereby circumventing the effort to cover the entire $SO(3)$. Our network natively consumes a compact representation of the group of 3D rotations - quaternions. We have theoretically shown its equivariance and established convergence results for our Weiszfeld dynamic routing by making connections to the literature of robust optimization. Our network by construction disentangles the object existence that is used as global features in classification. It is among the few for having an explicit group-valued latent space and thus naturally estimates the orientation of the input shape, even without a supervision signal.

***Limitations.*** In the current form our performance is severely affected by the shape symmetries. The length of the activation vector depends on the number of classes and for a sufficiently descriptive latent vector we need to have significant number of classes. On the other hand, this allows us to perform with merit on problems where the number of classes are large. The computation of LRFs are still sensitive to the point density changes and resampling. LRFs themselves can also be ambiguous and sometimes non-unique.

***Future work.*** Inspired by [17] and [54] our feature work will involve exploring the Lie algebra for equivariances, establishing invariance to the tangent directions, application of our network in the broader context of 6DoF object detection from point sets and looking for equivariances among point resampling.

## References


1. Afshar, P., Mohammadi, A., Plataniotis, K.N.: Brain tumor type classification via capsule networks. In: 2018 25th IEEE International Conference on Image Processing (ICIP) (2018)
2. 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)
3. Aftab, K., Hartley, R., Trumpf, J.: Generalized weiszfeld algorithms for lq optimization. IEEE transactions on pattern analysis and machine intelligence **37**(4) (2014)
4. Aftab, K., Hartley, R., Trumpf, J.: $l_q$ closest-point to affine subspaces using the generalized weiszfeld algorithm. International Journal of Computer Vision (2015)
5. Aoki, Y., Goforth, H., Srivatsan, R.A., Lucey, S.: Pointnetlk: Robust & efficient point cloud registration using pointnet. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7163–7172 (2019)
6. Bao, E., Song, L.: Equivariant neural networks and equivarification. arXiv preprint arXiv:1906.07172 (2019)
7. Becigneul, G., Ganea, O.E.: Riemannian adaptive optimization methods. In: International Conference on Learning Representations (2019)
8. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures. vol. 1611, pp. 586–606. International Society for Optics and Photonics (1992)
9. Birdal, T., Arbel, M., Simsekli, U., Guibas, L.J.: Synchronizing probability measures on rotations via optimal transport. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1569–1579 (2020)
10. Birdal, T., Ilic, S.: Point pair features based object detection and pose estimation revisited. In: 2015 International Conference on 3D Vision. pp. 527–535. IEEE (2015)
11. Birdal, T., Ilic, S.: A point sampling algorithm for 3d matching of irregular geometries. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE (2017)
12. Birdal, T., Simsekli, U., Eken, M.O., Ilic, S.: Bayesian pose graph optimization via bingham distributions and tempered geodesic mcmc. In: Advances in Neural Information Processing Systems. pp. 308–319 (2018)
13. Boomsma, W., Frellsen, J.: Spherical convolutions and their application in molecular modelling. In: Advances in Neural Information Processing Systems 30. pp. 3433–3443 (2017)
14. Burrus, C.S.: Iterative reweighted least squares. OpenStax CNX. Available online: http://cnx. org/contents/92b90377-2b34-49e4-b26f-7fe572db78a1 **12** (2012)
15. Busam, B., Birdal, T., Navab, N.: Camera pose filtering with local regression geodesics on the riemannian manifold of dual quaternions. In: IEEE International Conference on Computer Vision Workshop (ICCVW) (October 2017)
16. Chakraborty, R., Banerjee, M., Vemuri, B.C.: H-cnns: Convolutional neural networks for riemannian homogeneous spaces. arXiv preprint arXiv:1805.05487 (2018)
17. Cohen, T., Weiler, M., Kicanaoglu, B., Welling, M.: Gauge equivariant convolutional networks and the icosahedral CNN. In: Proceedings of the 36th International Conference on Machine Learning. pp. 1321–1330 (2019)
18. Cohen, T., Welling, M.: Group equivariant convolutional networks. In: International conference on machine learning. pp. 2990–2999 (2016)
19. Cohen, T.S., Geiger, M., Köhler, J., Welling, M.: Spherical cnns. In: 6th International Conference on Learning Representations, (ICLR) (2018)

20. Cohen, T.S., Geiger, M., Weiler, M.: A general theory of equivariant cnns on homogeneous spaces. In: Advances in Neural Information Processing Systems. pp. 9145–9156 (2019)
21. Cohen, T.S., Welling, M.: Steerable cnns. International Conference on Learning Representations (ICLR) (2017)
22. Cruz-Mota, J., Bogdanova, I., Paquier, B., Bierlaire, M., Thiran, J.P.: Scale invariant feature transform on the sphere: Theory and applications. International Journal of Computer Vision **98**(2), 217–241 (Jun 2012)
23. Deng, H., Birdal, T., Ilic, S.: Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In: European Conference on Computer Vision (ECCV) (2018)
24. Deng, H., Birdal, T., Ilic, S.: Ppfnet: Global context aware local features for robust 3d point matching. In: Conference on Computer Vision and Pattern Recognition (2018)
25. Esteves, C., Allen-Blanchette, C., Makadia, A., Daniilidis, K.: Learning so (3) equivariant representations with spherical cnns. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 52–68 (2018)
26. Esteves, C., Sud, A., Luo, Z., Daniilidis, K., Makadia, A.: Cross-domain 3d equivariant image embeddings. In: International Conference on Machine Learning (ICML) (2019)
27. Esteves, C., Xu, Y., Allen-Blanchette, C., Daniilidis, K.: Equivariant multi-view networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1568–1577 (2019)
28. Fey, M., Eric Lenssen, J., Weichert, F., Müller, H.: Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
29. Giles, C.L., Maxwell, T.: Learning, invariance, and generalization in high-order neural networks. Applied optics **26**(23), 4972–4978 (1987)
30. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: International Conference on Artificial Neural Networks. pp. 44–51. Springer (2011)
31. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points, vol. 26.2. ACM (1992)
32. Jaiswal, A., AbdAlmageed, W., Wu, Y., Natarajan, P.: Capsulegan: Generative adversarial capsule network. In: Computer Vision – ECCV 2018 Workshops. pp. 526–535. Springer International Publishing (2019)
33. Jiang, C.M., Huang, J., Kashinath, K., Prabhat, Marcus, P., Niessner, M.: Spherical CNNs on unstructured grids. In: International Conference on Learning Representations (2019)
34. Khoury, M., Zhou, Q.Y., Koltun, V.: Learning compact geometric features. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 153–161 (2017)
35. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
36. Kondor, R., Lin, Z., Trivedi, S.: Clebsch–gordan nets: a fully fourier space spherical convolutional neural network. In: Advances in Neural Information Processing Systems (2018)
37. Kondor, R., Trivedi, S.: On the generalization of equivariance and convolution in neural networks to the action of compact groups. In: International Conference on Machine Learning. pp. 2747–2755 (2018)
38. Kosiorek, A., Sabour, S., Teh, Y.W., Hinton, G.E.: Stacked capsule autoencoders. In: Advances in Neural Information Processing Systems. pp. 15512–15522 (2019)

39. Laue, S., Mitterreiter, M., Giesen, J.: Computing higher order derivatives of matrix and tensor expressions. In: Advances in Neural Information Processing Systems (2018)
40. Lenssen, J.E., Fey, M., Libuschewski, P.: Group equivariant capsule networks. In: Advances in Neural Information Processing Systems. pp. 8844–8853 (2018)
41. Li, J., Chen, B.M., Hee Lee, G.: So-net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2018)
42. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: Advances in Neural Information Processing Systems (2018)
43. Liao, S., Gavves, E., Snoek, C.G.: Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-spheres. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9759–9767 (2019)
44. Liu, M., Yao, F., Choi, C., Ayan, S., Ramani, K.: Deep learning 3d shapes using alt-az anisotropic 2-sphere convolution. In: International Conference on Learning Representations (ICLR) (2019)
45. Liu, X., Han, Z., Liu, Y.S., Zwicker, M.: Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 8778–8785 (2019)
46. Magnus, J.R.: On differentiating eigenvalues and eigenvectors. Econometric Theory **1**(2) (1985)
47. Marcos, D., Volpi, M., Komodakis, N., Tuia, D.: Rotation equivariant vector field networks. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
48. Markley, F.L., Cheng, Y., Crassidis, J.L., Oshman, Y.: Averaging quaternions. Journal of Guidance, Control, and Dynamics **30**(4), 1193–1197 (2007)
49. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: Intelligent Robots and Systems (IROS). IEEE (2015)
50. Mehr, E., Lieutier, A., Sanchez Bermudez, F., Guitteny, V., Thome, N., Cord, M.: Manifold learning in quotient spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9165–9174 (2018)
51. Melzi, S., Spezialetti, R., Tombari, F., Bronstein, M.M., Stefano, L.D., Rodola, E.: Gframes: Gradient-based local reference frame for 3d shape matching. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
52. Petrelli, A., Di Stefano, L.: On the repeatability of the local reference frame for partial shape matching. In: 2011 International Conference on Computer Vision. IEEE (2011)
53. Petrelli, A., Di Stefano, L.: A repeatable and efficient canonical reference for surface matching. In: 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission. pp. 403–410. IEEE (2012)
54. Poulenard, A., Ovsjanikov, M.: Multi-directional geodesic neural networks via equivariant convolution. In: SIGGRAPH Asia 2018 Technical Papers. p. 236. ACM (2018)
55. 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)
56. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5648–5656 (2016)

57. 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)
58. Rezatofighi, S.H., Milan, A., Abbasnejad, E., Dick, A., Reid, I., et al.: Deepsetnet: Predicting sets with deep neural networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 5257–5266. IEEE (2017)
59. Sabour, S., Frosst, N., Hinton, G.: Matrix capsules with em routing. In: 6th International Conference on Learning Representations, ICLR (2018)
60. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in neural information processing systems. pp. 3856–3866 (2017)
61. Schütt, K., Kindermans, P.J., Sauceda Felix, H.E., Chmiela, S., Tkatchenko, A., Müller, K.R.: Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In: Advances in Neural Information Processing Systems (2017)
62. Shen, Y., Feng, C., Yang, Y., Tian, D.: Mining point cloud local structures by kernel correlation and graph pooling. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4548–4557 (2018)
63. Spezialetti, R., Salti, S., Stefano, L.D.: Learning an effective equivariant 3d descriptor without supervision. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6401–6410 (2019)
64. Srivastava, N., Goh, H., Salakhutdinov, R.: Geometric capsule autoencoders for 3d point clouds. arXiv preprint arXiv:1912.03310 (2019)
65. Steenrod, N.E.: The topology of fibre bundles, vol. 14. Princeton University Press (1951)
66. Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., Riley, P.: Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. arXiv preprint arXiv:1802.08219 (2018)
67. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: European conference on computer vision. pp. 356–369. Springer (2010)
68. Wang, D., Liu, Q.: An optimization view on dynamic routing between capsules (2018), https://openreview.net/forum?id=HJjtFYJDf
69. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2642–2651 (2019)
70. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
71. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) (2019)
72. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) **38**(5), 1–12 (2019)
73. Weiler, M., Geiger, M., Welling, M., Boomsma, W., Cohen, T.: 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In: Advances in Neural Information Processing Systems. pp. 10381–10392 (2018)
74. Weiler, M., Hamprecht, F.A., Storath, M.: Learning steerable filters for rotation equivariant cnns. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)

75. Worrall, D., Brostow, G.: Cubenet: Equivariance to 3d rotation and translation. In: The European Conference on Computer Vision (ECCV) (September 2018)
76. Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Harmonic networks: Deep translation and rotation equivariance. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
77. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
78. Xinyi, Z., Chen, L.: Capsule graph neural network. In: International Conference on Learning Representations (ICLR) (2019), openreview.net/forum?id=Byl8BnRcYm
79. You, Y., Lou, Y., Liu, Q., Tai, Y.W., Ma, L., Lu, C., Wang, W.: Pointwise rotation-invariant network with adaptive sampling and 3d spherical voxel convolution. In: AAAI. pp. 12717–12724 (2020)
80. Yuan, W., Held, D., Mertz, C., Hebert, M.: Iterative transformer network for 3d point cloud. arXiv preprint arXiv:1811.11209 (2018)
81. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Advances in Neural Information Processing Systems (2017)
82. Zhang, X., Qin, S., Xu, Y., Xu, H.: Quaternion product units for deep learning on 3d rotation groups. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7304–7313 (2020)
83. Zhao, Y., Birdal, T., Deng, H., Tombari, F.: 3d point capsule networks. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

## A 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: $\mathcal{A}(\{\mathbf{q}_{\sigma(1)}, \ldots, \mathbf{q}_{\sigma(Q)}\}, \mathbf{w}_\sigma) = \mathcal{A}(\{\mathbf{q}_1, \ldots, \mathbf{q}_Q\}, \mathbf{w})$.
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:

$$\begin{aligned}
\mathbf{q}^\top \mathbf{M} \mathbf{q} &= \mathbf{q}^\top \Big( \sum_{i=1}^{Q} w_i \mathbf{q}_i \mathbf{q}_i^\top \Big) \mathbf{q} && (6)\\
&= \mathbf{q}^\top \Big( \sum_{i=1}^{Q} w_i (\mathbf{g} \circ \mathbf{q}_i)(\mathbf{g} \circ \mathbf{q}_i)^\top \Big) \mathbf{q} && (7)\\
&= \mathbf{q}^\top \Big( \sum_{i=1}^{Q} w_i \mathbf{G} \mathbf{q}_i \mathbf{q}_i^\top \mathbf{G}^\top \Big) \mathbf{q} && (8)\\
&= \mathbf{q}^\top \Big( \mathbf{G} \mathbf{M}_1 \mathbf{G}^\top + \cdots + \mathbf{G} \mathbf{M}_Q \mathbf{G}^\top \Big) \mathbf{q} \\
&= \mathbf{q}^\top \mathbf{G} \Big( \mathbf{M}_1 \mathbf{G}^\top + \cdots + \mathbf{M}_Q \mathbf{G}^\top \Big) \mathbf{q} && (9)\\
&= \mathbf{q}^\top \mathbf{G} \Big( \mathbf{M}_1 + \cdots + \mathbf{M}_Q \Big) \mathbf{G}^\top \mathbf{q} && (10)\\
&= \mathbf{q}^\top \mathbf{G} \mathbf{M} \mathbf{G}^\top \mathbf{q} && (11)\\
&= \mathbf{p}^\top \mathbf{M} \mathbf{p}, && (12)
\end{aligned}$$

   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 (9).
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 $\mathbf{g} \in \mathbb{H}_1$:

$$\begin{aligned}
|(\mathbf{g} \circ \mathbf{q}_1)^\top (\mathbf{g} \circ \mathbf{q}_2)| &= |\mathbf{q}_1^\top \mathbf{G}^\top \mathbf{G} \mathbf{q}_2| && (13)\\
&= |\mathbf{q}_1^\top \mathbf{I} \mathbf{q}_2| && (14)\\
&= |\mathbf{q}_1^\top \mathbf{q}_2|, && (15)
\end{aligned}$$

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

## B 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 9 (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:*

$$S = \Big\{ \sum_{i=1}^{d+1} \alpha_i \mathbf{x}_i \mid \sum_{i=1}^{d+1} \alpha_i = 1 \Big\}. \tag{16}$$

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

**Definition 10 (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:*

$$\mathit{\Pi}_i(\mathbf{x}) \triangleq proj_S(\mathbf{x}) = \mathbf{c} + \mathbf{A}(\mathbf{x} - \mathbf{c}). \tag{17}$$

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

**Definition 11 (Distance to Affine Subspaces)** *Distance from a given point* $\mathbf{x}$ *to a set of affine subspaces $\{S_1, S_2 \dots S_k\}$ can be written as [4]:*

$$C(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, S_i) = \sum_{i=1}^{k} \|\mathbf{x} - proj_{S_i}(\mathbf{x})\|^2. \tag{18}$$

**Lemma 2.** *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:*

$$h_i(\mathbf{x}) = \mathbf{q}_i^\top \mathbf{x} + q_d := 0. \tag{19}$$

*$q_d$ 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:*

$$proj_{S_i}(\mathbf{p}) = (\mathbf{I} - \mathbf{q}_i \mathbf{q}_i^\top)\mathbf{p} \tag{20}$$

*Proof.* We consider Eq (20) 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:

$$\mathbf{A}^\top\mathbf{A} = (\mathbf{I} - \mathbf{q}\mathbf{q}^\top)^\top(\mathbf{I} - \mathbf{q}\mathbf{q}^\top) \tag{21}$$
$$= (\mathbf{I} - \mathbf{q}\mathbf{q}^\top)(\mathbf{I} - \mathbf{q}\mathbf{q}^\top) \tag{22}$$
$$= \mathbf{I} - 2\mathbf{q}\mathbf{q}^\top + \mathbf{q}\mathbf{q}^\top\mathbf{q}\mathbf{q}^\top \tag{23}$$
$$= \mathbf{I} - 2\mathbf{q}\mathbf{q}^\top + \mathbf{q}\mathbf{q}^\top \tag{24}$$
$$= \mathbf{I} - \mathbf{q}\mathbf{q}^\top \triangleq \mathbf{A} \tag{25}$$

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

$$\mathbf{q}^\top\mathrm{proj}_S(\mathbf{q}) = \mathbf{q}^\top\mathbf{A}\mathbf{q} = \mathbf{q}^\top(\mathbf{I} - \mathbf{q}\mathbf{q}^\top)\mathbf{q} = \mathbf{q}^\top(\mathbf{q} - \mathbf{q}\mathbf{q}^\top\mathbf{q}) = \mathbf{q}^\top(\mathbf{q} - \mathbf{q}) = 0. \tag{26}$$

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

**Lemma 3.** *The quaternion mean we suggest to use in the main paper [48] 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 [4], which minimizes a $q$-norm variant of the cost defined in Eq (18):

$$C_q(\mathbf{x}) = \sum_{i=1}^{k} d(\mathbf{x}, S_i) = \sum_{i=1}^{k} \|\mathbf{x} - \mathrm{proj}_{S_i}(\mathbf{x})\|^q. \tag{27}$$

Defining $\mathbf{M}_i = \mathbf{I} - \mathbf{A}_i$, Alg. 3 summarizes the iterative procedure.

**Algorithm 3:** $L_q$ Weiszfeld Algorithm on Affine Subspaces [4].

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

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

**while** *not converged* **do**

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

$$w_i^t = \|\mathbf{M}_i(\mathbf{x}^t - \mathbf{c}_i)\|^{q-2} \quad \forall i = 1 \dots k \tag{28}$$

Solve:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}\in\mathbb{R}^N} \sum_{i=1}^{k} w_i^t \|\mathbf{M}_i(\mathbf{x} - \mathbf{c}_i)\|^2 \tag{29}$$

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 (29) and is

given by the normal equations:

$$\mathbf{x} = \Big(\sum_{i=1}^{k} w_i \mathbf{M}_i\Big)^{-1} \Big(\sum_{i=1}^{k} w_i \mathbf{M}_i \mathbf{c}_i\Big) \tag{30}$$

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

$$\Big(\sum_{i=1}^{k} \mathbf{M}_i\Big)\mathbf{x} = \Big(\frac{1}{k}\sum_{i=1}^{k} \mathbf{M}_i\Big)\mathbf{x} = \mathbf{0}. \tag{31}$$

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:

$$\mathbf{q}^\star = \underset{\mathbf{q}\in\mathcal{S}^3}{\arg\max}\, \mathbf{q}\mathbf{M}\mathbf{q} \tag{32}$$

which gives us the unweighted Quaternion mean [48].

## C 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 [2,3] :

**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 bounded convexity radius of $\rho < \pi/2$.
**H3.** The minimizer in Eq (29) 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 (29) will converge to a critical point unless $\mathbf{x}^t = \mathbf{y}_i$ for any $t$ and $i$ [3]. For $q = 1$, this critical point is on one of the subspaces specified in Eq (19) 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

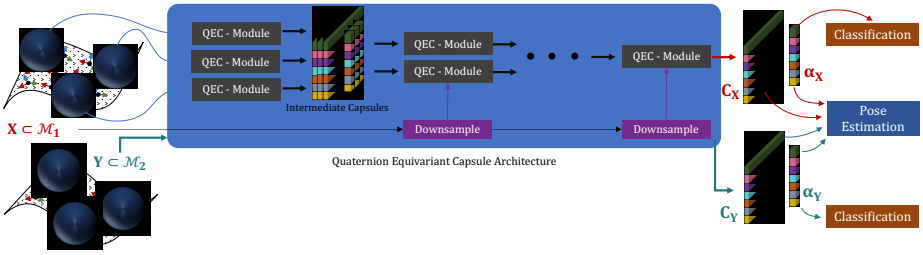



**Fig. 5.** Our siamese architecture used in the estimation of relative poses. We use a shared network to process two distinct point clouds $(\mathbf{X}, \mathbf{Y})$ to arrive at the latent representations $(\mathbf{C}_X, \boldsymbol{\alpha}_X)$ and $(\mathbf{C}_Y, \boldsymbol{\alpha}_Y)$ respectively. 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.

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.

## D 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 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:

$$\begin{aligned} C_{DR} &= LM + M(K + k(2L) + L) = M(K + 2(k+1)L) \\ C_{MLP} &= 64N_c + 4MN_c^2. \end{aligned} \tag{33}$$

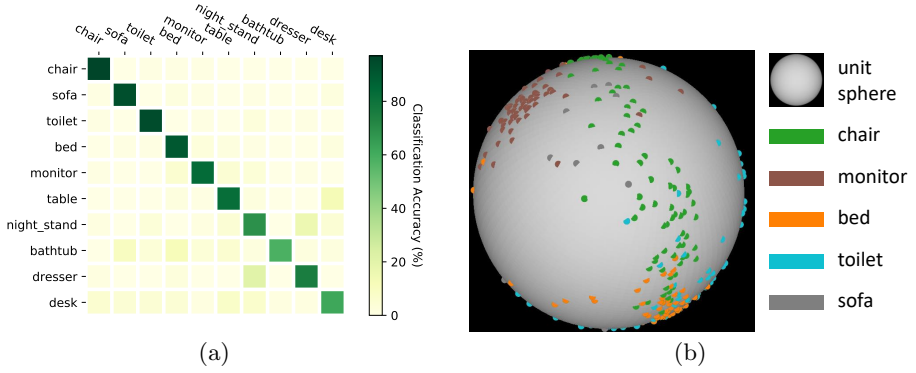



**Fig. 6. (a)** Confusion matrix on ModelNet10 for classification. **(b)** Distribution of initial poses per class.

Note that Eq (33) 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. Number 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.

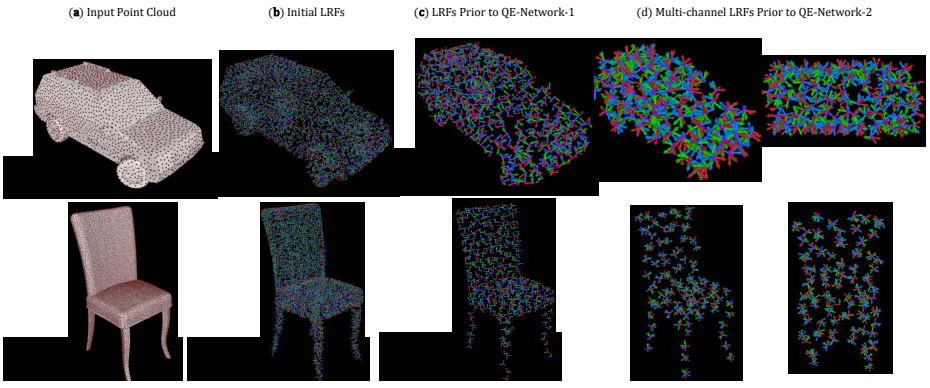



**Fig. 7.** Additional intermediate results on car (first row) and chair (second row) objects. This figure supplements Fig. 1(a) of the main paper.

***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. 6(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. 6(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.

## E 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. 5.

We show additional results from the computation of local reference frames and the multi-channel capsules deduced from our network in Fig. 7.

## F 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 [55,57]. The objects in training and testing dataset are different, but

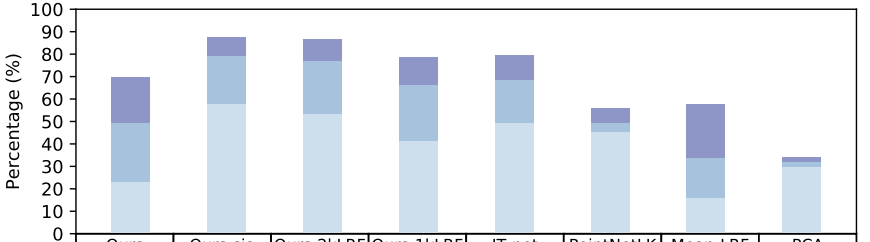
**Fig. 8.** Additional pairwise shape alignment on more categories in Modelnet10 dataset. We do not perform any ICP and the transformations that align the two point clouds are direct results of the forward pass of our Siamese network.

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 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. 6(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. 9 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.




| | 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. 9.** 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 $\theta$. 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.