Projective Manifold Gradient Layer for Deep Rotation Regression

Jiayi Chen[1,2] Yingda Yin[1] Tolga Birdal[3,4] Baoquan Chen[1] Leonidas J. Guibas[3] He Wang[1†]
[1]CFCS, Peking University [2]Beijing Institute for General AI
[3]Stanford University [4]Imperial College London

Abstract

Regressing rotations on SO(3) manifold using deep neural networks is an important yet unsolved problem. The gap between the Euclidean network output space and the non-Euclidean SO(3) manifold imposes a severe challenge for neural network learning in both forward and backward passes. While several works have proposed different regression-friendly rotation representations, very few works have been devoted to improving the gradient backpropagating in the backward pass. In this paper, we propose a manifold-aware gradient that directly backpropagates into deep network weights. Leveraging Riemannian optimization to construct a novel projective gradient, our proposed regularized projective manifold gradient (RPMG) method helps networks achieve new state-of-the-art performance in a variety of rotation estimation tasks. Our proposed gradient layer can also be applied to other smooth manifolds such as the unit sphere. Our project page is at https://jychen18.github.io/RPMG.

1. Introduction

Estimating rotations is a crucial problem in visual perception that has broad applications, *e.g.*, in object pose estimation, robot control, camera relocalization, 3D reconstruction and visual odometry [8, 12, 15, 21, 34]. Recently, with the proliferation of deep neural networks, learning to accurately regress rotations is attracting more and more attention. However, the non-Euclidean characteristics of rotation space make accurately regressing rotation very challenging.

As we know, rotations reside in a non-Euclidean manifold, $\mathrm{SO}(3)$ group, whereas the unconstrained outputs of neural networks usually live in Euclidean spaces. This gap between the neural network output space and $\mathrm{SO}(3)$ manifold becomes a major challenge for deep rotation regression, thus tackling this gap becomes an important research topic. One popular research direction is to design learning-friendly rotation representations, *e.g.*, 6D continuous representation from [42] and 10D symmetric matrix representation from [26]. Recently, Levinson *et al.* [24] adopted the vanilla 9D matrix representation discovering that simply replacing the Gram-Schmidt process in the 6D representation [42] with symmetric SVD-based orthogonalization can make this representation superior to the others.

Despite the progress on discovering better rotation representations, the gap between a Euclidean network output space and the non-Euclidean $\mathrm{SO}(3)$ manifold hasn't been completely filled. One important yet long-neglected problem lies in optimization on non-Euclidean manifolds [1]: to optimize on $\mathrm{SO}(3)$ manifold, the optimization variable is a rotation matrix, which contains nine matrix elements; if we naively use *Euclidean gradient*, which simply computes the partial derivatives with respect to each of the nine matrix elements, to update the variable, this optimization step will usually lead to a new matrix off $\mathrm{SO}(3)$ manifold. Unfortunately, we observe that all the existing works on rotation regression simply rely upon *vanilla auto-differentiation* for backpropagation, exactly computing Euclidean gradient and performing such off-manifold updates to predicted rotations. We argue that, for training deep rotation regression networks, the off-manifold components will lead to noise in the gradient of neural network weights, hindering network training and convergence.

To tackle this issue, we draw inspiration from differential geometry, where people leverage *Riemannian optimization* to optimize on the non-Euclidean manifold, which finds the direction of the steepest geodesic path on the manifold and take an on-manifold step. We thus propose to leverage Riemannian optimization and delve deep into the study of the backward pass. Note that this is a fundamental yet currently under-explored avenue, given that most of the existing works focus on a holistic design of rotation regression that is agnostic to forward/backward pass. However, incorporating Riemannian optimization into network training is highly non-trivial and challenging. Although methods of Riemannian optimization allow for optimization on $\mathrm{SO}(3)$ [5, 29], matrix manifolds [1] or general Riemannian manifolds [32, 40], they are not directly applicable to update the weights of the neural networks that are Euclidean. Also,

[†] : He Wang is the corresponding author (hewang@pku.edu.cn).

approaches like [16] incorporate a Riemannian distance as well as its gradient into network training, however, they do not deal with the *representation* issue.

In this work, we want to *propose a better manifold-aware gradient in the backward pass of rotation regression that directly updates the neural network weights*. We begin by taking a Riemannian optimization step and computing the difference between the rotation prediction and the updated rotation, which is closer to the ground truth. Back-propagating this ”error”, we encounter the mapping function (or orthogonalization function) that transforms the raw network output to a valid rotation. This projection, which can be the Gram-Schmidt process or SVD orthogonalization [24], is typically a many-to-one mapping. This non-bijectivity provides us with a new design space for our gradient: if we were to use a gradient to update the raw output rotation, many gradients would result in the same update in the final output rotation despite being completely different for backpropagating into the neural network weights. Now the problem becomes: *which gradient is the best for back-propagation when many of them correspond to the same update to the output?*

We observe that this problem is somewhat similar to some problems with ambiguities or multi-ground-truth issues. One example would be the symmetry issue in pose estimation: a symmetric object, *e.g.* a textureless cube, appears the same under many different poses, which needs to be considered when supervising the pose predictions. For supervising the learning in such a problem, Wang *et. al.* [36] proposed to use min-of-N loss [13], which only penalizes the smallest error between the prediction and all the possible ground truths. We therefore propose to find the gradient with the smallest norm that can update the final output rotation to the goal rotation. This *back-projection* process involves finding an element closest to the network output in the inverse image of the goal rotation and projecting the network output to this inverse image space. We therefore coin our gradient *projective manifold gradient*. One thing to note is that this projective gradient tends to shorten the network output, causing the norms of network output to vanish. To fix this problem, we further incorporate a simple regularization into the gradient, leading to our full solution *regularized projective manifold gradient* (RPMG).

Note that our proposed gradient layer operates on the raw network output and can be directly backpropagated into the network weights. Our method is very general and is not tied to a specific rotation representation. It can be coupled with different non-Euclidean rotation representations, including quaternion, 6D representation [42], and 9D rotation matrix representation [24], and can even be used for regressing other non-manifold variables.

We evaluate our devised projective manifold gradient layers on a diverse set of problems involving rotation regression: 3D object pose estimation from 3D point clouds/images, rotation estimation problems without using ground truth rotation supervisions, and please see supplementary material Section 5 for more experiments on camera relocalization. Our method demonstrates significant and consistent improvements on all these tasks and all different rotation representations tested. Going beyond rotation estimation, we also demonstrate performance improvements on regressing unit vectors (lie on a unit sphere) as an example of an extension to other non-Euclidean manifolds.

We summarize our contribution as below:

- We propose a novel manifold-aware gradient layer, namely *RPMG*, for the backward pass of rotation regression, which can be applied to different rotation representations and losses and used as a “plug-in” at no actual cost.
- Our extensive experiments over different tasks and rotation representations demonstrate the significant improvements from using RPMG.
- Our method can also benefit regression tasks on other manifolds, *e.g.* $\mathcal{S}^2$.

2. Related Work

Both rotation parameterization and optimization on SO(3) are well-studied topics. Early deep learning models leverage various rotation representations for pose estimation, *e.g.*, direction cosine matrix (DCM) [18, 39], axis-angle [11, 14, 33], quaternion [10, 20, 22, 38, 41] and Euler-angle [23,28,31]. Recently, [42] points out that Euler-angle, axis-angle, and quaternion are not continuous rotation representations, since their representation spaces are not homeomorphic to SO(3). As better representations for rotation regression, 6D [42], 9D [24], 10D [26] representations are proposed to resolve the discontinuity issue and improve the regression accuracy. A concurrent work [7] examines different manifold mappings theoretically and experimentally, finding out that SVD orthogonalization performs the best when regressing arbitrary rotations. Originating from general Riemannian optimization, [29] presents an easy approach for minimization on the $SO(3)$ group by constructing a local axis-angle parameterization, which is also the tangent space of $SO(3)$ manifold. They backpropagate gradient to the tangent space and use the exponential map to update the current rotation matrix. Most recently, [30] constructs a PyTorch library that supports tangent space gradient backpropagation for 3D transformation groups, (*e.g.*, $SO(3)$, $SE(3)$, $Sim(3)$). This proposed library can be used to implement the Riemannian gradient in our layer.

3. Preliminaries

3.1. Riemannian Geometry

Following [3, 4], we define an m-dimensional *Riemannian manifold* embedded in an ambient Euclidean space

$\mathcal{X} = \mathbb{R}^d$ and endowed with a *Riemannian metric* $\mathbf{G} \triangleq (\mathbf{G}_\mathbf{x})_{\mathbf{x}\in\mathcal{M}}$ to be a smooth curved space $(\mathcal{M}, G)$. A vector $\mathbf{v} \in \mathcal{X}$ is said to be *tangent* to $\mathcal{M}$ at $\mathbf{x}$ iff there exists a smooth curve $\gamma : [0,1] \mapsto \mathcal{M}$ s.t. $\gamma(0) = \mathbf{x}$ and $\dot{\gamma}(0) = \mathbf{v}$. The velocities of all such curves through $\mathbf{x}$ form the *tangent space* $\mathcal{T}_\mathbf{x}\mathcal{M} = \{\dot{\gamma}(0) \,|\, \gamma : \mathbb{R} \mapsto \mathcal{M}$ is smooth around 0 and $\gamma(0) = \mathbf{x}\}$.

Definition 1 (Riemannian gradient)**.** *For a smooth function $f : \mathcal{M} \mapsto \mathbb{R}$ and $\forall(\mathbf{x}, \mathbf{v}) \in \mathcal{TM}$, we define the* Riemannian gradient *of f as the unique vector field* grad$f$ *satisfying [6]:*

$$\mathrm{D}f(\mathbf{x})[\mathbf{v}] = \langle \mathbf{v}, \mathrm{grad}f(\mathbf{x})\rangle_\mathbf{x} \tag{1}$$

where $\mathrm{D}f(\mathbf{x})[\mathbf{v}]$ is the derivation of f by $\mathbf{v}$. It can further be shown (see supplementary material Section 2.1) that an expression for grad$f$ *can be obtained through the projection of the* Euclidean *gradient orthogonally onto the tangent space*

$$\mathrm{grad}f(\mathbf{x}) = \nabla f(\mathbf{x})_{\|} = \Pi_\mathbf{x}\big(\nabla f(\mathbf{x})\big). \tag{2}$$

where $\Pi_\mathbf{x} : \mathcal{X} \mapsto \mathcal{T}_\mathbf{x}\mathcal{M} \subseteq \mathcal{X}$ is an orthogonal projector with respect to $\langle\cdot,\cdot\rangle_\mathbf{x}$.

Definition 2 (Riemannian optimization)**.** *We consider gradient descent to solve the problems of $\min_{\mathbf{x}\in\mathcal{M}} f(\mathbf{x})$. For a local minimizer or a* stationary point $\mathbf{x}^\star$ *of f, the Riemannian gradient vanishes* grad$f(\mathbf{x}^\star) = 0$ *enabling a simple algorithm,* Riemannian gradient descent *(RGD):*

$$\mathbf{x}_{k+1} = R_{\mathbf{x}_k}(-\tau_k \,\mathrm{grad}f(\mathbf{x}_k)) \tag{3}$$

where τ_k is the step size at iteration k and $R_{\mathbf{x}_k}$ is the retraction *usually chosen related to the exponential map.*

3.2. Rotation Representations

There are many ways of representing a rotation: classic rotation representations, *e.g.* Euler angles, axis-angle, and quaternion; and recently introduced regression-friendly rotation representations such as *e.g.* 5D [42], 6D [42], 9D [24] and 10D [26] representations. A majority of deep neural networks can output an *unconstrained*, arbitrary n-dimensional vector $\mathbf{x}$ in a Euclidean space $\mathcal{X} = \mathbb{R}^n$. For Euler angle and axis-angle representations which use a vector from $\mathbb{R}^3$ to represent a rotation, a neural network can simply output a 3D vector; however, for quaternions, 6D, 9D or 10D representations that lies on non-Euclidean manifolds, manifold mapping function $\pi : \mathbb{R}^n \mapsto \mathcal{M}$ is generally needed for normalization or orthogonalization purposes to convert network outputs to valid elements belonging to the representation manifold. This network Euclidean output space $\mathcal{X}$ is where the representation manifolds reside and therefore are also called ambient space.

Definition 3 (Rotation representation)**.** *One rotation representation, which lies on a representation manifold $\mathcal{M}$, defines a surjective rotation mapping $\phi : \hat{\mathbf{x}} \in \mathcal{M} \to \phi(\hat{\mathbf{x}}) \in$* SO(3) *and a representation mapping function $\psi : \mathbf{R} \in$* SO(3) *$\to \psi(\mathbf{R}) \in \mathcal{M}$, such that $\phi(\psi) = \mathbf{R} \in$* SO(3).

Definition 4 (Manifold mapping function)**.** *From an ambient space $\mathcal{X}$ to the representation manifold $\mathcal{M}$, we can define a manifold mapping function $\pi : \mathbf{x} \in \mathcal{X} \to \pi(\mathbf{x}) \in \mathcal{M}$, which projects a point $\mathbf{x}$ in the ambient, Euclidean space to a valid element $\hat{\mathbf{x}} = \pi(\mathbf{x})$ on the manifold $\mathcal{M}$.*

We summarize the manifold mappings, the rotation mappings and representation mappings for several non-Euclidean rotation representations below.

Unit quaternion. Unit quaternions represent a rotation using a 4D unit vector $\mathbf{q} \in \mathcal{S}^3$ double covering the non-Euclidean 3-sphere *i.e.* $\mathbf{q}$ and $-\mathbf{q}$ identify the same rotation. A network with a final linear activation can only predict $\mathbf{x} \in \mathbb{R}^4$. The corresponding manifold mapping function is usually chosen to be a normalization step, which reads $\pi_q(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|$. For rotation and representation mapping, we leverage the standard mappings between rotation and quaternion (see supplementary material Section 7).

6D representation and Gram-Schmidt orthogonalization. 6D rotation representation [42], lying on Stiefel manifold $\mathcal{V}_2(\mathbb{R}^3)$, uses two orthogonal unit 3D vectors $(\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2)$ to represent a rotation, which are essentially the first two columns of a rotation matrix. Its manifold mapping π_{6D} is done through Gram-Schmidt orthogonalization. Its rotation mapping ϕ_{6D} is done by adding the third column $\hat{\mathbf{c}}_3 = \hat{\mathbf{c}}_1 \times \hat{\mathbf{c}}_2$. Its representation mapping ψ_{6D} is simply getting rid of the third column $\hat{\mathbf{c}}_3$ from a rotation matrix.

9D representation and SVD orthogonalization. To map a raw 9D network output $\mathbf{M}$ to a rotation matrix, [24] use SVD orthogonalization as the manifold mapping function π_{9D}, as follows: π_{9D} first decomposes $\mathbf{M}$ into its left and right singular vectors $\{\mathbf{U}, \mathbf{V}^\top\}$ and singular values Σ, $\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^\top$; then it replaces Σ with $\Sigma' = \mathrm{diag}(1, 1, \det(\mathbf{U}\mathbf{V}^\top))$ and finally, computes $\mathbf{R} = \mathbf{U}\Sigma'\mathbf{V}^\top$ to get the corresponding rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$. As this representation manifold is SO(3), both the rotation and representation mapping functions are simply identity.

10D representation. [26] propose a novel 10D representation for rotation matrix. The manifold mapping function π_{10D} maps $\boldsymbol{\theta} \in \mathbb{R}^{10}$ to $\mathbf{q} \in \mathcal{S}^3$ by computing the eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}(\boldsymbol{\theta})$, expressed as $\pi_{10D}(\mathbf{x}) = \min\limits_{\mathbf{q}\in\mathcal{S}^3} \mathbf{q}^\top\mathbf{A}(\mathbf{x})\mathbf{q}$, in which

$$\mathbf{A}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \\ \theta_2 & \theta_5 & \theta_6 & \theta_7 \\ \theta_3 & \theta_6 & \theta_8 & \theta_9 \\ \theta_4 & \theta_7 & \theta_9 & \theta_{10} \end{bmatrix}. \tag{4}$$

Since the representation manifold is $\mathcal{S}^3$, the rotation and representation mapping are the same as unit quaternion.

3.3. Deep Rotation Regression

We conclude this section by describing the ordinary forward and backward passes of a neural network based rota-

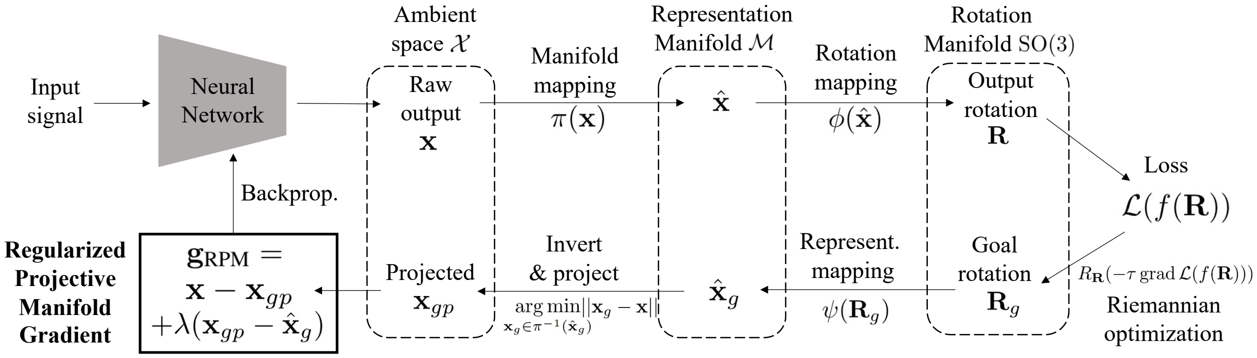
Input signal
Neural Network
Ambient space $\mathcal{X}$
Raw output $\mathbf{x}$
Manifold mapping $\pi(\mathbf{x})$
Representation Manifold $\mathcal{M}$
$\hat{\mathbf{x}}$
Rotation mapping $\phi(\hat{\mathbf{x}})$
Rotation Manifold $\mathrm{SO}(3)$
Output rotation $\mathbf{R}$
Loss $\mathcal{L}(f(\mathbf{R}))$
$R_{\mathbf{R}}(-\tau \operatorname{grad} \mathcal{L}(f(\mathbf{R})))$
Riemannian optimization
Goal rotation $\mathbf{R}_g$
Represent. mapping $\psi(\mathbf{R}_g)$
$\hat{\mathbf{x}}_g$
Invert & project $\underset{\mathbf{x}_g \in \pi^{-1}(\hat{\mathbf{x}}_g)}{\arg\min} \|\mathbf{x}_g - \mathbf{x}\|$
Projected $\mathbf{x}_{gp}$
$\mathbf{g}_{\text{RPM}} = \mathbf{x} - \mathbf{x}_{gp} + \lambda(\mathbf{x}_{gp} - \hat{\mathbf{x}}_g)$
Backprop.
Regularized Projective Manifold Gradient

Figure 1. **Projective Manifold Gradient Layer.** In the forward pass, the network predicts a raw output $\mathbf{x}$, which is then transformed into a valid rotation $\mathbf{R} = \phi(\pi(\mathbf{x}))$. We leave this forward pass unchanged and only modify the backward pass. In the backward pass, we first use Riemannian optimization to get a goal rotation $\mathbf{R}_g$ and map it back to $\hat{\mathbf{x}}_g$ on the representation manifold $\mathcal{M}$. After that we find the element $\mathbf{x}_{gp}$ which is closest to the raw output in the inverse image of $\hat{\mathbf{x}}_g$, and finally get the gradient $\mathbf{g}_{\mathbf{RPM}}$ we want.

tion regression, as used in [24, 42].

Forward and backward passes. Assume, for a rotation representation, the network predicts $\mathbf{x} \in \mathcal{X}$, then the manifold mapping π will map $\mathbf{x}$ to $\hat{\mathbf{x}} = \pi(\mathbf{x}) \in \mathcal{M}$, followed by a rotation mapping ϕ that finally yields the output rotation $\mathbf{R} = \phi(\hat{\mathbf{x}}) = \phi(\pi(\mathbf{x}))$. Our work only tackles the backward pass and keeps the forward pass unchanged, as shown in the top part of Figure 1. The gradient in the backward-pass is simply computed using Pytorch autograd method, that is $\mathbf{g} = f'(\mathbf{R})\phi'(\hat{\mathbf{x}})\pi'(\mathbf{x})$.

Loss function. The most common choice for supervising rotation matrix is L2 loss, $\|\mathbf{R} - \mathbf{R}_{gt}\|_F^2$, as used by [24,42]. This loss is equal to $4 - 4\cos(<\mathbf{R}, \mathbf{R}_{gt}>)$, where $<\mathbf{R}, \mathbf{R}_{gt}>$ represents the angle between $\mathbf{R}$ and $\mathbf{R}_{gt}$.

4. Method

Overview. In this work, we propose a *projective manifold gradient layer*, without changing the forward pass of a given rotation regressing network, as shown in Figure 1. Our focus is to find a better gradient $\mathbf{g}$ of the loss function $\mathcal{L}$ with respect to the network raw output $\mathbf{x}$ for backpropagation into the network weights.

Let's start with examining the gradient of network output $\mathbf{x}$ in a general case – regression in Euclidean space. Given a ground truth $\mathbf{x}_{gt}$ and the L2 loss $\|\mathbf{x} - \mathbf{x}_{gt}\|^2$ that maximizes the likelihood in the presence of Gaussian noise in $\mathbf{x}$, the gradient would be $\mathbf{g} = 2(\mathbf{x} - \mathbf{x}_{gt})$.

In the case of rotation regression, we therefore propose to find a proper $\mathbf{x}^* \in \mathcal{X}$ for a given ground truth $\mathbf{R}_{gt}$ or a computed goal rotation $\mathbf{R}_g$ when the ground truth rotation is not available, and then simply use $\mathbf{x} - \mathbf{x}^*$ as our gradient to backpropagate into the network.

Note that finding such an $\mathbf{x}^*$ can be challenging. Assuming we know $\mathbf{R}_{gt}$, finding an $\mathbf{x}^*$ involves inverting ϕ and π since the network output $\mathbf{R} = \phi(\pi(\mathbf{x}))$. Furthermore, we may not know $\mathbf{R}_{gt}$ under indirect rotation supervision (*e.g.*, flow loss as used in PoseCNN [38]) and self-supervised rotation estimation cases (*e.g.*, 2D mask loss as used in [35]).

In this work, we introduce the following techniques to mitigate these problems: (i) we first take a Riemannian gradient to compute a goal rotation $\mathbf{R}_g \in \mathrm{SO}(3)$, which does not rely on knowing $\mathbf{R}_{gt}$, as explained in Section 4.1; (ii) we then find the set of all possible $\mathbf{x}_g$s that can be mapped to $\mathbf{R}_g$, or in other words, the inverse image of $\mathbf{R}_g$ under π and ϕ; (iii) we find $\mathbf{x}_{gp}$ which is the element in this set closest to $\mathbf{x}$ in the Euclidean metric and set it as "$\mathbf{x}^*$". We will construct our gradient using this $\mathbf{x}^*$, as explained in 4.2. (iv) we add a regularization term to this gradient forming $\mathbf{g}_{RPMG}$ as explained in 4.3. The whole backward pass leveraging our proposed regularized projective manifold gradient is shown in the lower half of Figure 1.

4.1. Riemannian Gradient and Goal Rotation

To handle rotation estimation with/without direct rotation supervision, we first propose to compute the Riemannian gradient of the loss function $\mathcal{L}$ with respect to the output rotation $\mathbf{R}$ and find a goal rotation $\mathbf{R}_g$ that is presumably closer to the ground truth rotation than $\mathbf{R}$.

Assume the loss function is in the following form $\mathcal{L}(f(\mathbf{R}))$, where $\mathbf{R} = \pi(\phi(\mathbf{x}))$ is the output rotation and f constructs a loss function that compares $\mathbf{R}$ to the ground truth rotation $\mathbf{R}_{gt}$ directly or indirectly. Given $\mathbf{R}(\mathbf{x})$ and $\mathcal{L}(f(\mathbf{R}(\mathbf{x})))$, we can perform one step of Riemannian optimization yielding our goal rotation $\mathbf{R}_g \leftarrow R_{\mathbf{R}}(-\tau \operatorname{grad} \mathcal{L}(f(\mathbf{R})))$, where τ is the step size of Riemannian gradient and can be set to a constant as a hyperparameter or varying during the training. For L2 loss $\|\mathbf{R} - \mathbf{R}_{gt}\|_F^2$, the Riemannian gradient is always along the geodesic path between $\mathbf{R}$ and $\mathbf{R}_{gt}$ on SO(3) [19]. In this case, $\mathbf{R}_g$ can generally be seen as an intermediate goal between $\mathbf{R}$ and $\mathbf{R}_{gt}$ dependent on τ. Gradually increasing τ from 0 will first make $\mathbf{R}_g$ approach $\mathbf{R}_{gt}$ starting with $\mathbf{R}_g = \mathbf{R}$, and then reach $\mathbf{R}_{gt}$ where we denote $\tau = \tau_{gt}$, and finally going beyond $\mathbf{R}_{gt}$. Although, when $\mathbf{R}_{gt}$ is available, one can simply set $\mathbf{R}_g = \mathbf{R}_{gt}$, we argue that this is just a special case under $\tau = \tau_{gt}$. For scenarios where $\mathbf{R}_{gt}$ is unavailable, *e.g.*, in self-supervised learning cases (see in Section 5.3), we don't know $\mathbf{R}_{gt}$ and τ_{gt}, thus we need to compute $\mathbf{R}_g$ using Riemannian optimization. In the sequel, we only use $\mathbf{R}_g$ for explaining our methods without loss of generality. See Section 4.3 for how to choose τ.

4.2. Projective Manifold Gradient

Given $\mathbf{R}_g$, we can use the representation mapping ψ to find the corresponding $\hat{\mathbf{x}}_g = \psi(\mathbf{R}_g)$ on the representation manifold $\mathcal{M}$. However, further inverting π and finding the corresponding $\mathbf{x}_g \in \mathcal{X}$ is a non-trivial problem, due to the projective nature of π. In fact, there are many $\mathbf{x}_g$s that satisfy $\pi(\mathbf{x}_g) = \hat{\mathbf{x}}_g$. It seems that we can construct a gradient $\mathbf{g} = (\mathbf{x} - \mathbf{x}_g)$ using any $\mathbf{x}_g$ that satisfies $\pi(\mathbf{x}_g) = \hat{\mathbf{x}}_g$. No matter which $\mathbf{x}_g$ we choose, if this gradient were to update $\mathbf{x}$, it will result in the same $\mathbf{R}_g$. But, when backpropagating into the network, those gradients will update the network weights differently, potentially resulting in different learning efficiency and network performance.

Formally, we formulate this problem as *a multi-ground-truth problem* for $\mathbf{x}$: we need to find the best $\mathbf{x}^*$ to supervise from the inverse image of $\hat{\mathbf{x}}_g$ under the mapping π. We note that similar problems have been seen in pose supervision dealing with symmetry as in [36], where one needs to find one pose to supervise when there are many poses under which the object appears the same. [36] proposed to use a min-of-N strategy introduced by [13]: from all possible poses, taking the pose that is closest to the network prediction as ground truth. A similar strategy is also seen in supervising quaternion regression, as $\mathbf{q}$ and $-\mathbf{q}$ stand for the same rotation. One common choice of the loss function is therefore $\min\{\mathcal{L}(\mathbf{q}, \mathbf{q}_{gt}), \mathcal{L}(\mathbf{q}, -\mathbf{q}_{gt})\}$ [26], which penalizes the distance to the closest ground truth quaternion.

Inspired by these works, we propose to choose our gradient among all the possible gradients with the lowest level of redundancy, *i.e.*, we require $\mathbf{x}^*$ to be the one closest to $\mathbf{x}$, or in other words, the gradient to have the smallest norm, meaning that we need to find the projection point $\mathbf{x}_{gp}$ of $\mathbf{x}$ to all the valid $\mathbf{x}_g$:

$$\mathbf{x}_{gp} = \underset{\pi(\mathbf{x}_g) = \hat{\mathbf{x}}_g}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{x}_g\|_2 \tag{5}$$

We then can construct our *projective manifold gradient* (PMG) as $\mathbf{g}_{PM} = \mathbf{x} - \mathbf{x}_{gp}$. We will denote the naive gradient $\mathbf{g}_M = \mathbf{x} - \hat{\mathbf{x}}_g$ as *manifold gradient* (MG).

Here we provide another perspective on why a network may prefer PMG. In the case where a deep network is trained using stochastic gradient descent (SGD), the final gradient used to update the network weights is averaged across the gradients of all the batch instances. If gradients from different batch instances contain different levels of redundancy, then the averaged gradient may be biased or not even appropriate. This argument is generally applicable to all stochastic optimizers (*e.g.*, Adam [2])

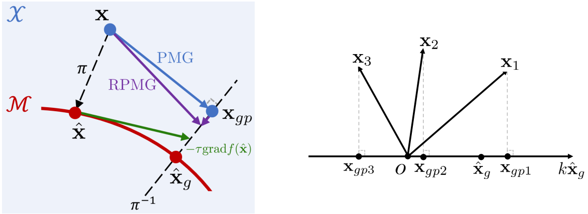

Figure 2. **Illustration for regularized projective manifold gradient. Left**: In the forward pass, we simply project $\mathbf{x}$ to $\hat{\mathbf{x}}$ by π. In the backward pass, first we compute a Riemannian gradient, which is shown as the *green* arrow. After getting a next goal $\hat{\mathbf{x}}_g \in \mathcal{M}$ by Riemannian optimization, we find the inverse projection $\mathbf{x}_{gp}$ of $\hat{\mathbf{x}}_g$, which leads to our *projective manifold gradient*, shown as the *blue* arrow. With a regularization term, we can get our final *regularized projective manifold gradient*, as the *purple* arrow. **Right**: Projection point $\hat{\mathbf{x}}_{gp}$ in the case of quaternion.

Inverting π. There are many ways to solve this projection problem for different manifold mapping functions π. For example, we can formulate this as a constrained optimization problem. For the manifold mapping functions we consider, we propose the following approach: we first solve for the inverse image $\pi^{-1}(\hat{\mathbf{x}}_g)$ of $\hat{\mathbf{x}}_g$ in the ambient space $\mathcal{X}$ analytically, which reads $\pi^{-1}(\hat{\mathbf{x}}_g) = \{\mathbf{x}_g \in \mathcal{X} \mid \pi(\mathbf{x}_g) = \hat{\mathbf{x}}_g\}$; we then project $\mathbf{x}$ onto this inverse image space. Note that, sometimes only a superset of this inverse image can be found analytically, requiring certain constraints on $\mathbf{x}_{gp}$ to be enforced.

Here we list the inverse image $\pi^{-1}(\hat{\mathbf{x}}_g)$ and the projection point $\mathbf{x}_{gp}$ for different rotation representations and their corresponding manifold mapping π. Please refer to supplementary material Section 2.2 for detailed derivations.

Quaternion. With $\pi_q(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|$, $\mathbf{x} \in \mathbb{R}^4$, and $\hat{\mathbf{x}}_g \in \mathcal{S}^3$: $\pi_q^{-1}(\hat{\mathbf{x}}_g) = \{\mathbf{x} \mid \mathbf{x} = k\hat{\mathbf{x}}_g, k \in \mathbb{R} \text{ and } k > 0\}$, which is a ray in the direction of $\hat{\mathbf{x}}_g$ starting from the origin. Without considering the constraint of $k > 0$, an analytical solution to this projection point $\mathbf{x}_{gp}$ of $\mathbf{x}$ onto this line can be derived: $\mathbf{x}_{gp} = (\mathbf{x} \cdot \hat{\mathbf{x}}_g)\hat{\mathbf{x}}_g$.

6D representation. With π_{6D} as Gram-Schmidt process, $\mathbf{x} = [\mathbf{u}, \mathbf{v}] \in \mathbb{R}^6$, and $\hat{\mathbf{x}}_g \in \mathcal{V}_2(\mathbb{R}^3)$: $\pi_{6D}^{-1}(\hat{\mathbf{x}}_g) = \{[k_1\hat{\mathbf{u}}_g, k_2\hat{\mathbf{u}}_g + k_3\hat{\mathbf{v}}_g] \mid k_1, k_2, k_3 \in \mathbb{R} \text{ and } k_1, k_3 > 0\}$ (the former is a ray whereas the latter spans a half plane). Without considering the constraint of $k_1, k_3 > 0$, the projection point $\mathbf{x}_{gp}$ can be analytically represented as $\mathbf{x}_{gp} = [(\mathbf{u} \cdot \hat{\mathbf{u}}_g)\hat{\mathbf{u}}_g, (\mathbf{v} \cdot \hat{\mathbf{u}}_g)\hat{\mathbf{u}}_g + (\mathbf{v} \cdot \hat{\mathbf{v}}_g)\hat{\mathbf{v}}_g]$

9D representation. With $\pi_{9D}(\mathbf{x})$ as SVD orthogonalization, $\mathbf{x} \in \mathbb{R}^{3\times3}$, and $\hat{\mathbf{x}}_g \in \mathrm{SO}(3)$, the analytical expression for π_{9D}^{-1} is available when we ignore the positive singular value constraints, which gives $\pi_{9D}^{-1}(\hat{\mathbf{x}}_g) = \{\mathbf{S}\hat{\mathbf{x}}_g \mid \mathbf{S} =$

$\mathbf{S}^\top\}$. We can further solve the projection point $\mathbf{x}_{gp}$ with an elegant representation $\mathbf{x}_{gp} = \frac{\mathbf{x}\hat{\mathbf{x}}_g^T + \hat{\mathbf{x}}_g\mathbf{x}^T}{2}$.

10D representation. Please refer to supplementary material Section 2.2 for the derivation and expression of $\mathbf{x}_{gp}$.

4.3. Regularized Projective Manifold Gradient

Issues in naive projective manifold gradient. In the right plot of Figure 2, we illustrate this projection process for several occasions where $\mathbf{x}$ takes different positions relative to $\mathbf{x}_g$. We demonstrate that there are two issues in this process.

First, no matter where $\mathbf{x}$ is in, the projection operation will shorten the length of our prediction because $\|\mathbf{x}_{gp}\| < \|\mathbf{x}\|$ is always true for all of 4D/6D/9D/10D representation. This will cause the length norm of the prediction of the network to become very small as the training progresses (see Figure 3). The shrinking network output will keep increasing the effective learning rate, preventing the network from convergence and leading to great harm to the network performance (see Table 2 and Figure 3 for ablation study).

Second, when the angle between $\mathbf{x}$ and $\hat{\mathbf{x}}_g$ becomes larger than $\pi/2$ (in the case of $\mathbf{x} = \mathbf{x}_3$), the naive projection $\mathbf{x}_{gp}$ will be in the opposite direction of $\hat{\mathbf{x}}_g$ and can not be mapped back to $\hat{\mathbf{x}}_g$ under π_q, resulting in a wrong gradient. The same set of issues also happens to 6D, 9D and 10D representations. The formal reason is that the analytical solution of the inverse image assumes certain constraints are satisfied, which is usually true only when either $\hat{\mathbf{x}}_g$ is not far from $\mathbf{x}$ or the network is about to converge.

Regularized projective manifold gradient. To solve the first issue, we propose to add a regularization term $\mathbf{x}_{gp} - \hat{\mathbf{x}}_g$ to the projective manifold gradient, which can avoid the length vanishing problem. The *regularized projective manifold gradient* then reads:

$$\mathbf{g}_{RPM} = \mathbf{x} - \mathbf{x}_{gp} + \lambda(\mathbf{x}_{gp} - \hat{\mathbf{x}}_g), \tag{6}$$

where λ is a regularization coefficient. See the left plot of Figure 2 for an illustration.

Discussion on the hyperparameters λ and τ. Our method apparently introduces two additional hyperparameters, λ and τ, however, we argue that this doesn't increase the searching space of hyperparameters for our method.

For λ, the only requirement is that λ is small (we simply set to 0.01), because: (1) we want the projective manifold gradient $(\mathbf{x} - \mathbf{x}_{gp})$ to be the major component of our gradient; (2) since this regularization is roughly proportional to the difference in prediction length and a constant, a small lambda is enough to prevent the length from vanishing and, in the end, the prediction length will stay roughly constant at the equilibrium under projection and regularization. In the ablation study of Section 5.1, we show that the performance is robust to the change of λ. Note that, on the other extreme, when $\lambda = 1$, $\mathbf{g}_{RPM}$ becomes $\mathbf{g}_M$.

For τ, we propose a ramping up schedule which is well-motivated. To tackle the second problem of reversed gradient, we need a small τ_{init} to keep $\mathbf{R}_g$ close to $\mathbf{R}$ at the beginning of training. But when the network is about to converge, we will prefer a $\tau_{converge}$ which can keep $\mathbf{R}_g$ close to $\mathbf{R}_{gt}$ for better convergence. We cannot directly set $\tau_{converge}$ to τ_{gt}, which is introduced in 4.1, because τ_{gt} is not a constant and cannot be used in Riemannian Optimization. However, if we want to tackle the problem of reversed gradient, we must need Riemannian Optimization and τ_{init}. Thus we need a constant approximation of τ_{gt} when the angle between $\mathbf{R}$ and $\mathbf{R}_{gt}$ converges to 0. Note that $\tau_{converge}$ can be derived analytically when the loss function is the most widely used L2 loss or geodesic loss(please refer to supplementary material Section 2.1 for details), and therefore doesn't need to be tuned. Therefore we propose to increase τ from a small value τ_{init}, leading to a slow warm-up and, as the training progresses, we gradually increase it to the final $\tau = \tau_{converge}$ by ten uniform steps. This strategy further improves our performance.

5. Experiments

We investigate popular rotation representations and find our methods greatly improve the performance in different kinds of tasks. For our regularized projective manifold gradient (**RPMG**), we apply it in the backpropagation process of Quaternion, 6D, 9D and 10D, without changing the forward pass, leading to three new methods **RPMG-Quat**, **RPMG-6D**, **RPMG-9D** and **RPMG-10D**. We compare the following seven baselines: **Euler angle**, **axis-angle**, **Quaternion**, **6D** [42], **9D** [24], **9D-Inf** [24] and **10D** [26]. We adopt three evaluation metrics: mean, median, and 5° accuracy of (geodesic) errors between predicted rotation and ground truth rotation. For most of our experiments, we set the regularization term $\lambda = 0.01$ and increase τ from $\tau_{init} = 0.05$ to $\tau_{converge} = 0.25$ by ten uniform steps. We further show and discuss the influence of different choices of these two hyperparameters in our ablation studies.

5.1. 3D Object Pose Estimation from Point Clouds

Experimental setting. As in [9], we use the complete point clouds generated from the models in ModelNet-40 [37]. We use the same train/test split as in [9] and report the results of *airplane*, *chair*, *sofa*, *toilet* and *bed* those five categories because they exhibit less rotational symmetries. Given one shape point clouds of a specific category, the network learns to predict the 3D rotation of the input point clouds from the predefined canonical view of this category [36]. We replace the point clouds alignment task used in [24, 42] (which has almost been solved) by this experiment since it is more challenging and closer to real-world applications (no canonical point clouds is given to the network).

We use a PointNet++ [27] network as our backbone, supervised by L2 loss between the predicted rotation matrix

Methods	Airplane			Chair			Sofa			Toilet			Bed		
	Mn↓	Md↓	Acc↑	Mn↓	Md↓	Acc↑	Mn↓	Md↓	Acc↑	Mn↓	Md↓	Acc↑	Mn↓	Md↓	Acc↑
Euler	125	131	0	13.6	9.0	17	120	125	0	127	133	0	113	122	0
Axis-Angle	10.8	8.2	22	16.4	10.9	9	24.1	14.6	6	21.9	13.0	9	25.5	11.0	16
Quaternion	9.7	7.6	27	16.7	11.4	12	20.4	12.7	10	16.0	9.3	17	27.8	11.3	14
6D	5.5	4.7	54	9.8	6.4	35	14.6	9.5	15	9.3	6.8	33	24.7	9.6	17
9D	4.7	3.9	67	7.9	5.4	44	15.7	10.0	14	10.3	6.9	30	22.3	8.5	20
9D-Inf (MG-9D)	3.1	2.5	90	5.3	3.7	69	7.8	5.0	50	4.2	3.3	75	12.9	4.6	55
10D	5.3	4.2	61	8.9	6.0	38	15.1	10.3	13	10.7	6.5	35	23.1	8.7	19
RPMG-Quat	3.2	2.4	88	6.3	3.7	67	8.1	4.5	57	4.9	3.5	74	13.3	3.6	70
RPMG-6D	2.6	2.1	**94**	**5.0**	**3.1**	74	6.6	3.6	70	**3.8**	2.9	**83**	13.5	2.7	81
RPMG-9D	**2.5**	**2.0**	**94**	5.1	**3.1**	**76**	**6.1**	**3.1**	**77**	4.3	**2.7**	**83**	**10.9**	**2.5**	**86**
RPMG-10D	2.8	2.2	93	5.1	3.2	75	6.5	3.2	72	4.9	2.8	82	13.5	2.7	82

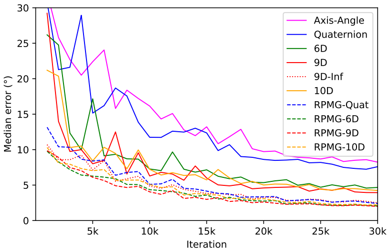

Table 1. **Pose estimation from ModelNet40 point clouds.** Left: a comparison of methods by mean, median, and 5° accuracy of (geodesic) errors after 30k training steps. Mn, Md and Acc are abbreviations of mean, median and 5° accuracy. Right: median test error of *airplane* in different iterations during training.

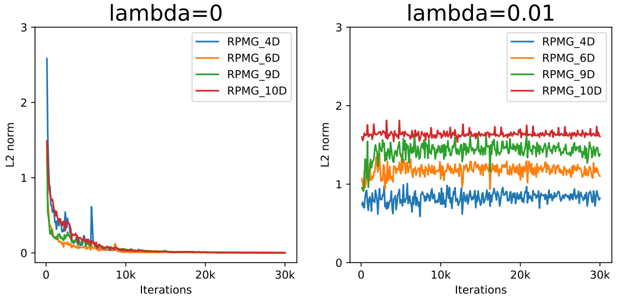

Figure 3. **Average L2 norm of the network raw output x** during training. Left: PMG-4D/6D/9D/10D (w/o reg. $\lambda = 0$). Right: RPMG-4D/6D/9D/10D (w/ reg. $\lambda = 0.01$)

$\mathbf{R}$ and the ground truth rotation matrix $\mathbf{R}_{gt}$. To facilitate a fair comparison between multiple methods, we use the same set of hyperparameters in all the experiments. Please see supplementary material Section 6.1 for more details.

Analysis of results. The results are shown in Table 1. We see a great improvement of our methods in all three rotation representations. In this experiment, one may find **9D-Inf** also leads to a good performance, which is actually a special case of our method with $\lambda = 1$, or in other words, it is MG with $\tau = \tau_{gt}$. Nonetheless, in Table 3, we can observe a larger gap. Also, this simple loss may lead to bad performance when $\mathbf{R}_{gt}$ is unavailable in Section 5.3.

Ablation study on λ. As mentioned in Section 4.3, naively using **PMG** without any regularization, corresponding to setting $\lambda = 0$, will lead to length vanishing; To maintain the length of prediction roughly constant, we only need to add a small λ. In Figure 3, We show the length vanishing problem without regularization and stabilized length with a small regularization. In Table 1, we show that the network performs much better when we have a small λ (**RPMG**) than $\lambda = 0$ (**PMG**) or $\lambda = 1$ (**MG**), which deviates too far away from the desired projective manifold gradient. As for the exact value of λ, our experiments show that our method is robust to the choice of λ as long as it is small. Table 2 also shows that $\lambda = 0.01, 0.005, 0.05$ all lead to similar performance, thus freeing us from tuning the parameter λ.

Ablation study on τ. For the choices of τ, Table 2 shows that our proposed strategy, which ramps up τ from a small τ_{init} to τ_{converge}, works the best. The reason is that: a big τ, when training begins, may cause the problem of reversed gradient discussed in Section 4.3. On the other side, a small τ at the end of training will slow down the training process and can do harm to convergence. Note that, the performance is not very sensitive to the exact value, which means we don't require a parameter tuning for τ even in general cases. We are good even with simply setting $\tau = \tau_{gt}$.

Methods			Mean (°)↓	Med (°)↓	5°Acc (%)↑
L2 6D	-	-	5.50	4.67	54.4
MG-6D	$\lambda = 1$	$\tau_{converge}$	3.51	2.95	85.2
		τ_{gt}	3.19	2.72	87.8
PMG-6D	$\lambda = 0$	$\tau_{converge}$	57.65	45.22	0.2
		τ_{gt}	133	136	0.0
RPMG-6D	$\lambda = 0.01$	τ_{init}	2.67	2.18	93.1
		$\tau_{converge}$	2.71	2.14	93.2
		τ_{gt}	3.02	2.14	89.5
		$\tau_{init} \rightarrow \tau_{converge}$	2.59	2.07	93.6
	$\lambda = 0.05$	$\tau_{init} \rightarrow \tau_{converge}$	2.73	2.23	92.9
	$\lambda = 0.005$		**2.52**	**2.05**	**94.3**

Table 2. **Ablation study of pose estimation from *airplane* point clouds.** Here MG stands for manifold gradient $\mathbf{x} - \hat{\mathbf{x}}_g$, corresponding to set $\lambda = 1$; PMG stands for projective manifold gradient $\mathbf{x} - \mathbf{x}_{gp}$, corresponding to set $\lambda = 0$.

5.2. 3D Rotation Estimation from ModelNet Images

In this experiment, we follow the setting in [24] to estimate poses from 2D images. Images are rendered from ModelNet-10 [37] objects from arbitrary viewpoints [25]. A MobileNet [17] is used to extract image features and three MLPs to regress rotations. We use the same categories as in Experiment 5.1 except *airplane*, since ModelNet-10 doesn't have this category. We didn't quote the numbers from [24] since we conduct all the experiments using the same set of hyperparameters to ensure a fair comparison. Please see supplementary material Section 6.2 for more details.

The results are shown in Table 3. Our RPMG layer boosts the performance of all three representations significantly. See the curves with the same color for comparison.

Methods	Chair			Sofa			Toilet			Bed		
	Mn↓	Md↓	Acc↑	Mn↓	Md↓	Acc↑	Mn↓	Md↓	Acc↑	Mn↓	Md↓	Acc↑
Euler	21.5	10.9	10	27.5	12.0	9	14.9	8.5	19	27.6	9.6	17
Axis-Angle	25.7	14.3	7	30.3	14.6	6	20.3	13.0	8	36.3	16.7	4
Quaternion	25.8	15.0	6	30.0	15.7	6	20.6	13.0	8	34.1	15.5	5
6D	19.6	9.1	19	17.5	7.3	27	10.9	6.2	37	32.3	11.7	11
9D	17.5	8.3	23	19.8	7.6	25	11.8	6.5	34	30.4	11.1	13
9D-Inf	12.1	5.1	49	12.5	3.5	70	7.6	3.7	67	22.5	4.5	56
10D	18.4	9.0	20	20.9	8.7	20	11.5	5.9	39	29.9	11.5	11
RPMG-Quat	13.0	5.9	40	13.0	3.6	67	8.6	4.2	61	23.2	4.9	51
RPMG-6D	12.9	4.7	53	11.5	2.8	77	7.8	3.4	71	20.3	3.6	67
RPMG-9D	**11.9**	**4.4**	**58**	**10.5**	**2.4**	**82**	7.5	3.2	75	20.0	**2.9**	**76**
RPMG-10D	12.8	4.5	55	11.2	**2.4**	**82**	**7.2**	**3.0**	**76**	**19.2**	**2.9**	75

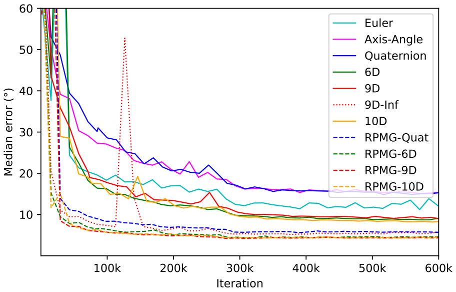

Table 3. **Pose estimation from ModelNet10 images.** Left: a comparison of methods by mean(°), median(°), and 5° accuracy(%) of (geodesic) errors after 600k training steps. Mn, Md and Acc are abbreviations of mean, median and 5° accuracy. Right: median test error of *chair* in different iterations during training.

5.3. Rotation Estimation without Supervision

Self-supervised instance-level rotation estimation from point clouds. For one complete chair instance Z, given a complete observation X, we estimate its pose $\mathbf{R}$. We then use Chamfer distance between Z and $\mathbf{R}^{-1}X$ as a self-supervised loss. The network structure and training settings are all the same as Experiment 5.1, except here we use $\tau = 2$. See supplementary material Section 5.2 for how to find a suitable τ.

The interesting thing here is that vanilla **9D-Inf** fails while our methods still perform very well. We think that this is because the Chamfer distance loss will greatly enlarge the effect of the noisy part (which is introduced by λ) in gradient, leading to a very bad performance.

Methods	Mean (°)↓	Med (°)↓	3°Acc (%)↑
Euler	131.9	139.1	0.0
Axis-Angle	4.5	3.8	34.5
Quaternion	4.3	3.5	37.5
6D	55.1	6.7	20.0
9D	1.8	1.6	88.0
9D-Inf	118.2	119.5	0.0
10D	1.6	1.5	91.0
RPMG-Quat	3.5	2.4	70.0
RPMG-6D	15.0	2.9	55.0
RPMG-9D	**1.3**	**1.2**	**97.5**
RPMG-10D	1.5	1.4	97.0

Table 4. **Self-supervised Instance-Level Rotation Estimation from Point Clouds.** We report mean, median and 3° accuracy of (geodesic) errors after 30K iterations.

5.4. Regression on Other Non-Euclidean Manifolds

In addition to SO(3), our method can also be applied for regression on other non-Euclidean manifolds as long as the target manifold meets some conditions: 1) the manifold should support Riemannian optimization. 2) the inverse projection π^{-1} should be calculable, although it doesn't need to be mathematically complete. Here we show the experiment of *Sphere manifold* $\mathcal{S}^2$.

Unit vector regression. For rotational symmetric categories (e.g., *bottle*), the pose of an object is ambiguous. We'd rather regress a unit vector for each object indicating the *up* direction of it. We use the ModelNet-40 [37] *bottle* point cloud dataset. The network architecture is the same as in Experiment 5.1 except the dimension of output is 3.

L2-loss-w/-norm computes L2 loss between the normalized predictions and the ground truth. L2-loss-w/o-norm computes L2 loss between the raw predictions and the ground truth, similar to $\lambda = 1$ and $\tau = \tau_{gt}$. For MG-3D, PMG-3D and RPMG-3D, We increase τ from 0.1 to 0.5 since here $\tau_{converge} = 0.5$ (please see supplementary material Section 3.1 for the derivation).

The results are shown in Table 5. MG-3D performs on par with L2-loss-w/o-norm, and PMG-3D leads to a large error since the length vanishing problem similar to Figure 3. RPMG-3D outperforms all the baselines and variants.

Methods	Mean (°)↓	Med (°)↓	1°Acc (%)↑
L2 loss w/ norm	8.73	2.71	0.0
L2 loss w/o norm	5.71	1.10	37.4
MG-3D (λ=1)	5.37	1.20	22.2
PMG-3D (λ=0)	21.96	14.79	0.0
RPMG-3D (λ=0.01)	**4.69**	**0.76**	**72.7**

Table 5. **Unit vector estimation from ModelNet bottle point clouds.** We report mean, median, and 1° accuracy of (geodesic) errors after 30K iterations.

6. Conclusion and Future Work

Our work tackles the problem of designing a gradient layer to facilitate the learning of rotation regression. Our extensive experiments have demonstrated the effectiveness of our method coupled with different rotation representations in diverse tasks dealing with rotation estimation.

The limitation of our methods mainly lies in two fronts: 1) we introduce two new hyperparameters, *i.e.*, τ and λ, though our performance is not sensitive to them, as long as they are in a reasonable range; 2) as discussed in Sec 5.4, our method can only be applied to manifolds with certain constraints. We leave how to relax those to future works.

References

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009. 1

[2] Henry Adams, M. Aminian, Elin Farnell, M. Kirby, C. Peterson, Joshua Mirth, R. Neville, P. Shipman, and C. Shonkwiler. A fractal dimension for measures via persistent homology. *arXiv: Dynamical Systems*, pages 1–31, 2020. 5

[3] Tolga Birdal and Umut Simsekli. Probabilistic permutation synchronization using the riemannian structure of the birkhoff polytope. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11105–11116, 2019. 2

[4] Tolga Birdal, Umut Simsekli, Mustafa Onur Eken, and Slobodan Ilic. Bayesian pose graph optimization via bingham distributions and tempered geodesic mcmc. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[5] Jose-Luis Blanco. A tutorial on se (3) transformation parameterizations and on-manifold optimization. *University of Malaga, Tech. Rep*, 3:6, 2010. 1

[6] Nicolas Boumal. An introduction to optimization on smooth manifolds. *Available online, May*, 2020. 3

[7] Romain Brégier. Deep regression on manifolds: a 3d rotation case study. *CoRR*, abs/2103.16317, 2021. 2

[8] Mai Bui, Tolga Birdal, Haowen Deng, Shadi Albarqouni, Leonidas Guibas, Slobodan Ilic, and Nassir Navab. 6d camera relocalization in ambiguous scenes via continuous multimodal inference. *arXiv preprint arXiv:2004.04807*, 2020. 1

[9] Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Randall Hill. Equivariant point network for 3d point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14514–14523, 2021. 6

[10] Haowen Deng, Mai Bui, Nassir Navab, Leonidas Guibas, Slobodan Ilic, and Tolga Birdal. Deep bingham networks: Dealing with uncertainty and ambiguity in pose estimation. *arXiv preprint arXiv:2012.11002*, 2020. 2

[11] Thanh-Toan Do, Ming Cai, Trung Pham, and Ian D. Reid. Deep-6dpose: Recovering 6d object pose from a single RGB image. *CoRR*, abs/1802.10367, 2018. 2

[12] Siyan Dong, Qingnan Fan, He Wang, Ji Shi, Li Yi, Thomas Funkhouser, Baoquan Chen, and Leonidas Guibas. Robust neural routing through space partitions for camera relocalization in dynamic indoor environments. *arXiv preprint arXiv:2012.04746*, 2020. 1

[13] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 2, 5

[14] Ge Gao, Mikko Lauri, Jianwei Zhang, and Simone Frintrop. Occlusion resistant object rotation regression from point cloud segments. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018. 2

[15] Zan Gojcic, Caifa Zhou, Jan D Wegner, Leonidas J Guibas, and Tolga Birdal. Learning multiview 3d point cloud registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1759–1769, 2020. 1

[16] Benjamin Hou, Nina Miolane, Bishesh Khanal, Matthew CH Lee, Amir Alansary, Steven McDonagh, Jo V Hajnal, Daniel Rueckert, Ben Glocker, and Bernhard Kainz. Computing cnn loss and gradients for pose estimation with riemannian geometry. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 756–764. Springer, 2018. 2

[17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. 7

[18] Jiahui Huang, He Wang, Tolga Birdal, Minhyuk Sung, Federica Arrigoni, Shi-Min Hu, and Leonidas J Guibas. Multibodysync: Multi-body segmentation and motion estimation via 3d scan synchronization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7108–7118, 2021. 2

[19] Du Q Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009. 4

[20] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[21] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 1

[22] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2

[23] Abhijit Kundu, Yin Li, and James M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[24] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *arXiv preprint arXiv:2006.14616*, 2020. 1, 2, 3, 4, 6, 7

[25] Shuai Liao, Efstratios Gavves, and Cees G. M. Snoek. Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-spheres. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, USA, June 2019. 7

[26] Valentin Peretroukhin, Matthew Giamou, David M. Rosen, W. Nicholas Greene, Nicholas Roy, and Jonathan Kelly. A Smooth Representation of SO(3) for Deep Rotation Learning with Uncertainty. In *Proceedings of Robotics: Science and Systems (RSS'20)*, Jul. 12–16 2020. 1, 2, 3, 5, 6

[27] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 6

[28] Hao Su, Charles R. Qi, Yangyan Li, and Leonidas J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2

[29] Camillo J Taylor and David J Kriegman. Minimization on the lie group so (3) and related manifolds. *Yale University*, 16(155):6, 1994. 1, 2

[30] Zachary Teed and Jia Deng. Tangent space backpropagation for 3d transformation groups. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[31] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 2

[32] Constantin Udriste. *Convex functions and optimization methods on Riemannian manifolds*, volume 297. Springer Science & Business Media, 2013. 1

[33] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[34] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019. 1

[35] Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. Self6d: Self-supervised monocular 6d object pose estimation. In *The European Conference on Computer Vision (ECCV)*, August 2020. 4

[36] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 2, 5, 6

[37] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Computer Vision and Pattern Recognition*, 2015. 6, 7, 8

[38] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 2, 4

[39] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *ACM Transactions on Graphics*, 37(6), 2019. 2

[40] Hongyi Zhang, Sashank J Reddi, and Suvrit Sra. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. *arXiv preprint arXiv:1605.07147*, 2016. 1

[41] Yongheng Zhao, Tolga Birdal, Jan Eric Lenssen, Emanuele Menegatti, Leonidas Guibas, and Federico Tombari. Quaternion equivariant capsule networks for 3d point clouds. In *European Conference on Computer Vision*, pages 1–19. Springer, 2020. 2

[42] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 1, 2, 3, 4, 6