On the Continuity of Rotation Representations in Neural Networks

Yi Zhou*
University of Southern California
zhou859@usc.edu

Connelly Barnes*
Adobe Research
connellybarnes@yahoo.com

Jingwan Lu
Adobe Research
jlu@adobe.com

Jimei Yang
Adobe Research
jimyang@adobe.com

Hao Li
University of Southern California, Pinscreen
USC Institute for Creative Technologies
hao@hao-li.com

Abstract

In neural networks, it is often desirable to work with various representations of the same space. For example, 3D rotations can be represented with quaternions or Euler angles. In this paper, we advance a definition of a continuous representation, which can be helpful for training deep neural networks. We relate this to topological concepts such as homeomorphism and embedding. We then investigate what are continuous and discontinuous representations for 2D, 3D, and n-dimensional rotations. We demonstrate that for 3D rotations, all representations are discontinuous in the real Euclidean spaces of four or fewer dimensions. Thus, widely used representations such as quaternions and Euler angles are discontinuous and difficult for neural networks to learn. We show that the 3D rotations have continuous representations in 5D and 6D, which are more suitable for learning. We also present continuous representations for the general case of the n dimensional rotation group $SO(n)$. While our main focus is on rotations, we also show that our constructions apply to other groups such as the orthogonal group and similarity transforms. We finally present empirical results, which show that our continuous rotation representations outperform discontinuous ones for several practical problems in graphics and vision, including a simple autoencoder sanity test, a rotation estimator for 3D point clouds, and an inverse kinematics solver for 3D human poses.

1. Introduction

Recently, there has been an increasing number of applications in graphics and vision, where deep neural networks are used to perform regressions on rotations. This has been done for tasks such as pose estimation from images [13, 31] and from point clouds [15], structure from motion [29], and skeleton motion synthesis, which generates the rotations of joints in skeletons [30]. Many of these works represent 3D rotations using 3D or 4D representations such as quaternions, axis-angles, or Euler angles.

However, for 3D rotations, we found that 3D and 4D representations are not ideal for network regression, when the full rotation space is required. Empirically, the converged networks still produce large errors at certain rotation angles. We believe that this actually points to deeper topological problems related to the continuity in the rotation representations. Informally, all else being equal, discontinuous representations should in many cases be “harder” to approximate by neural networks than continuous ones. Theoretical results suggest that functions that are smoother [33] or have stronger continuity properties such as in the modulus of continuity [32, 10] have lower approximation error for a given number of neurons.

Based on this insight, we first present in Section 3 our definition of the continuity of representation in neural networks. We illustrate this definition based on a simple example of 2D rotations. We then connect it to key topological concepts such as homeomorphism and embedding.

Next, we present in Section 4 a theoretical analysis of the continuity of rotation representations. We first investigate in Section 4.1 some discontinuous representations, such as Euler angle and quaternion representations. We show that for 3D rotations, all representations are discontinuous in four or fewer dimensional real Euclidean space with the Euclidean topology. We then investigate in Section 4.2 some continuous rotation representations. For the n dimensional rotation group $SO(n)$, we present a continuous $n^2 - n$ dimensional representation. We additionally present an option to reduce the dimensionality of this representation by an additional 1 to $n - 2$ dimensions in a continuous way. We show that these allow us to represent 3D rotations continuously in 6D and 5D. While we focus on rotations, we show how our continuous representations can also apply to other groups such as orthogonal groups $O(n)$ and similarity transforms.

Finally, in Section 5 we test our ideas empirically. We conduct experiments on 3D rotations and show that our 6D and 5D continuous representations always outperform

[*] Authors have equal contribution.

the discontinuous ones for several tasks, including a rotation autoencoder "sanity test," rotation estimation for 3D point clouds, and 3D human pose inverse kinematics learning. We note that in our rotation autoencoder experiments, discontinuous representations can have up to 6 to 14 times higher mean errors than continuous representations. Furthermore they tend to converge much slower while still producing large errors over 170° at certain rotation angles even after convergence, which we believe are due to the discontinuities being harder to fit. This phenomenon can also be observed in the experiments on different rotation representations for homeomorphic variational auto-encoding in Falorsi et al. [14], and in practical applications, such as 6D object pose estimation in Xiang et al. [31].

We also show that one can perform direct regression on 3x3 rotation matrices. Empirically this approach introduces larger errors than our 6D representation as shown in Section 5.2. Additionally, for some applications such as inverse and forward kinematics, it may be important for the network itself to produce orthogonal matrices. We therefore require an orthogonalization procedure in the network. In particular, if we use a Gram-Schmidt orthogonalization, we then effectively end up with our 6D representation.

Our contributions are: 1) a definition of continuity for rotation representations, which is suitable for neural networks; 2) an analysis of discontinuous and continuous representations for 2D, 3D, and n-D rotations; 3) new formulas for continuous representations of $SO(3)$ and $SO(n)$; 4) empirical results supporting our theoretical views and that our continuous representations are more suitable for learning.

2. Related Work

In this section, we will first establish some context for our work in terms of neural network approximation theory. Next, we discuss related works that investigate the continuity properties of different rotation representations. Finally, we will report the types of rotation representations used in previous learning tasks and their performance.

Neural network approximation theory. We review a brief sampling of results from neural network approximation theory. Hornik [17] showed that neural networks can approximate functions in the L^p space to arbitrary accuracy if the L^p norm is used. Barron et al. [6] showed that if a function has certain properties in its Fourier transform, then at most $O(\epsilon^{-2})$ neurons are needed to obtain an order of approximation ϵ. Chapter 6.4.1 of LeCun et al. [23] provides a more thorough overview of such results. We note that results for continuous functions indicate that functions that have better smoothness properties can have lower approximation error for a particular number of neurons [32, 10, 33]. For discontinuous functions, Llanas et al. [24] showed that a real and piecewise continuous function can be approximated in an almost uniform way. However, Llanas et al. [24] also noted that piecewise continuous functions when trained with gradient descent methods require many neurons and training iterations, and yet do not give very good results. These results suggest that continuous rotation representations might perform better in practice.

Continuity for rotations. Grassia et al. [16] pointed out that Euler angles and quaternions are not suitable for orientation differentiation and integration operations and proposed exponential map as a more robust rotation representation. Saxena et al. [28] observed that the Euler angles and quaternions cause learning problems due to discontinuities. However, they did not propose general rotation representations other than direct regression of 3x3 matrices, since they focus on learning representations for objects with specific symmetries.

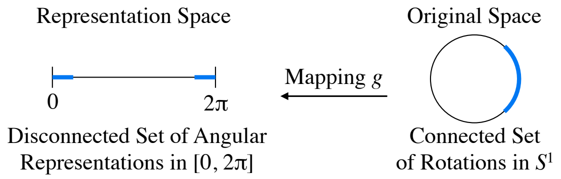
Neural networks for 3D shape pose estimation. Deep networks have been applied to estimate the 6D poses of object instances from RGB images, depth maps or scanned point clouds. Instead of directly predicting 3x3 matrices that may not correspond to valid rotations, they typically use more compact rotation representations such as quaternion [31, 21, 20] or axis-angle [29, 15, 13]. In PoseCNN [31], the authors reported a high percentage of errors between 90° and 180°, and suggested that this is mainly caused by the rotation ambiguity for some symmetric shapes in the test set. However, as illustrated in their paper, the proportion of errors between 90° to 180° is still high even for non-symmetric shapes. In this paper, we argue that discontinuity in these representations could be one cause of such errors.

Neural networks for inverse kinematics. Recently, researchers have been interested in training neural networks to solve inverse kinematics equations. This is because such networks are faster than traditional methods and differentiable so that they can be used in more complex learning tasks such as motion re-targeting [30] and video-based human pose estimation [19]. Most of these works represented rotations using quaternions or axis-angle [18, 19]. Some works also used other 3D representations such as Euler angles and Lie algebra [19, 34], and penalized the joint position errors. Csiszar et al. [11] designed networks to output the sine and cosine of the Euler angles for solving the inverse kinematics problems in robotic control. Euler angle representations are discontinuous for $SO(3)$ and can result in large regression errors as shown in the empirical test in Section 5. However, those authors limited the rotation angles to be within a certain range, which avoided the discontinuity points and thus achieved very low joint alignment errors in their test. However, many real-world tasks require the networks to be able to output the full range of rotations. In such cases, continuous rotation representations will be a better choice.

3. Definition of Continuous Representation

In this section, we begin by defining the terminology we will use in the paper. Next, we analyze a simple motivating example of 2D rotations. This allows us to develop our general definition of continuity of representation in neural networks. We then explain how this definition of continuity is related to concepts in topology.

Terminology. To denote a matrix, we typically use M, and M_{ij} refers to its (i, j) entry. We use the term $SO(n)$ to denote the *special orthogonal group*, the space of n dimen-



Representation Space
Original Space
Mapping g
0
2π
Disconnected Set of Angular Representations in [0, 2π]
Connected Set of Rotations in S^1

Figure 1. A simple 2D example, which motivates our definition of continuity of representation. See Section 3 for details.

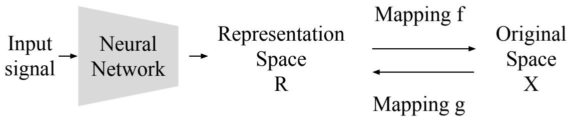


Figure 2. Our definition of continuous representation, as well as how it can apply in a neural network. See the body for details.

sional rotations. This group is defined on the set of $n \times n$ real matrices with $MM^T = M^TM = I$ and $\det(M) = 1$. The group operation is multiplication, which results in the concatenation of rotations. We denote the n *dimensional unit sphere* as $S^n = \{x \in \mathbb{R}^{n+1} : ||x|| = 1\}$.

Motivating example: 2D rotations. We now consider the representation of 2D rotations. For any 2D rotation $M \in SO(2)$, we can also express the matrix as:

$$M = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (1)$$

We can represent any rotation matrix $M \in SO(2)$ by choosing $\theta \in R$, where R is a suitable set of angles, for example, $R = [0, 2\pi]$. However, this particular representation intuitively has a problem with continuity. The problem is that if we define a mapping g from the original space $SO(2)$ to the angular representation space R, then this mapping is discontinuous. In particular, the limit of g at the identity matrix, which represents zero rotation, is undefined: one directional limit gives an angle of 0 and the other gives 2π. We depict this problem visually in Figure 1. On the right, we visualize a connected set of rotations $C \subset SO(2)$ by visualizing their first column vector $[\cos(\theta), \sin(\theta)]^T$ on the unit sphere S^1. On the left, after mapping them through g, we see that the angles are disconnected. In particular, we say that this representation is discontinuous because the mapping g from the original space to the representation space is discontinuous. We argue that these kind of discontinuous representations can be harder for neural networks to fit. Contrarily, if we represent the 2D rotation $M \in SO(2)$ by its first column vector $[\cos(\theta), \sin(\theta)]^T$, then the representation would be continuous.

Continuous representation: We can now define what we consider a continuous representation. We illustrate our definitions graphically in Figure 2. Let R be a subset of a real vector space equipped with the Euclidean topology. We call R the *representation space*: in our context, a neural network produces an intermediate representation in R. This neural network is depicted on the left side of Figure 2. We will come back to this neural network shortly. Let X be a compact topological space. We call X the *original space*. In our context, any intermediate representation in R produced by the network can be *mapped* into the original space X. Define the *mapping to the original space* $f : R \to X$, and the *mapping to the representation space* $g : X \to R$. We say (f, g) is a *representation* if for every $x \in X, f(g(x)) = x$, that is, f is a left inverse of g. We say the *representation is continuous* if g is continuous.

Connection with neural networks: We now return to the neural network on the left side of Figure 2. We imagine that inference runs from left to right. Thus, the neural network accepts some input signals on its left hand side, outputs a representation in R, and then passes this representation through the mapping f to get an element of the original space X. Note that in our context, *the mapping f* is implemented as a mathematical function that is used as part of the forward pass of the network at both training and inference time. Typically, at training time, we might impose losses on the original space X. We now describe the intuition behind why we ask that g be continuous. Suppose that we have some connected set C in the original space, such as the one shown on the right side of Figure 1. Then if we map C into representation space R, and g is continuous, then the set $g(C)$ will remain connected. Thus, if we have continuous training data, then this will effectively create a continuous training signal for the neural network. Contrarily, if g is not continuous, as shown in Figure 1, then a connected set in the original space may become disconnected in the representation space. This could create a discontinuous training signal for the network. We note that the units in the neural network are typically continuous, as defined on Euclidean topology spaces. Thus, we require the representation space R to have Euclidean topology because this is consistent with the continuity of the network units.

Domain of the mapping f: We additionally note that for neural networks, it is specifically beneficial for the mapping f to be defined almost everywhere on a set where the neural network outputs are expected to lie. This enables f to *map arbitrary representations produced by the network* back to the original space X.

Connection with topology: Suppose that (f, g) is a continuous representation. Note that g is a continuous one-to-one function from a compact topological space to a Hausdorff space. From a theorem in topology [22], this implies that if we restrict the codomain of g to $g(X)$ (and use the subspace topology for $g(X)$) then the resulting mapping is a homeomorphism. A *homeomorphism* is a continuous bijection with a continuous inverse. For geometric intuition, a homeomorphism is often described as a continuous and invertible stretching and bending of one space to another, with also a finite number of cuts allowed if one later glues back together the same points. One says that two spaces are topologically equivalent if there is a homeomorphism between them. Additionally, g is a topological *embedding* of the original space X into the representation space R. Note that we also have the inverse of g: if we restrict f to the domain $g(X)$ then the resulting function $f|_{g(X)}$ is simply the inverse of g. Conversely, if the original space X is not homeomorphic to any subset of the representation space R

then there is no possible continuous representation (f, g) on these spaces. We will return to this later when we show that there is no continuous representation for the 3D rotations in four or fewer dimensions.

4. Rotation Representation Analysis

Here we provide examples of rotation representations that could be used in networks. We start by looking in Section 4.1 at some discontinuous representations for 3D rotations, then look in Section 4.2 at continuous rotation representations in n dimensions, and show that how for the 3D rotations, these become 6D and 5D continuous rotation representations. We believe this analysis can help one to choose suitable rotation representations for learning tasks.

4.1. Discontinuous Representations

Case 1: Euler angle representation for the 3D rotations. Let the original space $X = SO(3)$, the set of 3D rotations. Then we can easily show discontinuity in an Euler angle representation by considering the azimuth angle θ and reducing this to the motivating example for 2D rotations shown in Section 3. In particular, the identity rotation I occurs at a discontinuity, where one directional limit gives $\theta = 0$ and the other directional limit gives $\theta = 2\pi$. We visualize the discontinuities in this representation, and all the other representations, in the supplemental Section F.

Case 2: Quaternion representation for the 3D rotations. Define the original space $X = SO(3)$, and the representation space $Y = \mathbb{R}^4$, which we use to represent the quaternions. We can now define the mapping to the representation space $g_q(M) =$

$$\begin{cases} \left[M_{32} - M_{23},\, M_{13} - M_{31},\, M_{21} - M_{12},\, t \right]^T & \text{if } t \neq 0 \\ \left[\sqrt{M_{11} + 1},\, c_2\sqrt{M_{22} + 1},\, c_3\sqrt{M_{33} + 1},\, 0 \right]^T & \text{if } t = 0 \end{cases} \tag{2}$$

$$t = \mathrm{Tr}(M) + 1, c_i = \begin{cases} 1 & \text{if } M_{i,1} + M_{i,2} > 0 \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

Likewise, one can define the mapping to the original space $SO(3)$ as in [2]:

$$f_q\left([x_0, y_0, z_0, w_0]\right) = \\ \begin{bmatrix} 1 - 2y^2 - 2z^2, & 2xy - 2zw, & 2xz + 2yw \\ 2xy + 2zw, & 1 - 2x^2 - 2z^2, & 2yz - 2xw \\ 2xz - 2yw, & 2yz + 2xw, & 1 - 2x^2 - 2y^2 \end{bmatrix}, \\ (x, y, z, w) = N([x_0, y_0, z_0, w_0]) \tag{4}$$

Here the normalization function is defined as $N(q) = q/||q||$. By expanding in terms of the axis-angle representation for the matrix M, one can verify that for every $M \in SO(3), f(g(M)) = M$.

However, we find that the representation is not continuous. Geometrically, this can be seen by taking different directional limits around the matrices with 180 degree rotations, which are defined by $R_\pi = \{M \in SO(3) : \mathrm{Tr}(M) = -1\}$. Specifically, in the top case of Equation (2), where $t \neq 0$, the limit of g_q as we approach a 180 degree rotation is $[\,0, 0, 0, 0\,]$, and meanwhile, the first three coordinates of $g_q(r)$ for $r \in R_\pi$ are nonzero. Note that our definition of continuous representation from Section 3 requires a Euclidean topology for the representation space Y, in contrast to the usual topology for the quaternions of the real projective space $\mathbb{R}P^3$, which we discuss in the next paragraph. In a similar way, we can show that other popular representations for the 3D rotations such as axis-angle have discontinuities, e.g. the axis in axis-angle has discontinuities at the 180 degree rotations.

Representations for the 3D rotations are discontinuous in four or fewer dimensions. The 3D rotation group $SO(3)$ is homeomorphic to the real projective space $\mathbb{R}P^3$. The space $\mathbb{R}P^n$ is defined as the quotient space of $\mathbb{R}^{n+1} \setminus \{0\}$ under the equivalence relation that $x \sim \lambda x$ for all $\lambda \neq 0$. In a graphics and vision context, it may be most intuitive to think of $\mathbb{R}P^3$ as being the homogeneous coordinates in $\mathbb{R}^4$ with the previous equivalence relation used to construct an appropriate topology via the quotient space.

Based on standard embedding and non-embedding results in topology [12], we know that $\mathbb{R}P^3$ (and thus $SO(3)$) embeds in $\mathbb{R}^5$ with the Euclidean topology, but does not embed in $\mathbb{R}^d$ for any $d < 5$. By the definition of embedding, there is no homeomorphism from $SO(3)$ to any subset of $\mathbb{R}^d$ for any $d < 5$, but a continuous representation requires this. Thus, there is no such continuous representation.

4.2. Continuous Representations

In this section, we develop two continuous representations for the n dimensional rotations $SO(n)$. We then explain how for the 3D rotations $SO(3)$ these become 6D and 5D continuous rotation representations.

Case 3: Continuous representation with $n^2 - n$ dimensions for the n dimensional rotations. The rotation representations we have considered thus far are all not continuous. One possibility to make a rotation representation continuous would be to just use the identity mapping, but this would result in matrices of size $n \times n$ for the representation, which can be excessive, and would still require orthogonalization, such as a Gram-Schmidt process in the mapping f to the original space, if we want to ensure that network outputs end up back in $SO(n)$. Based on this observation, we propose to perform an orthogonalization process in the representation itself. Let the original space $X = SO(n)$, and the representation space be $R = \mathbb{R}^{n \times (n-1)} \setminus D$ (D will be defined shortly). Then we can define a mapping g_{GS} to the representation space that simply drops the last column vector of the input matrix:

$$g_{\mathrm{GS}}\left(\begin{bmatrix} | & & | \\ a_1 & \ldots & a_n \\ | & & | \end{bmatrix}\right) = \begin{bmatrix} | & & | \\ a_1 & \ldots & a_{n-1} \\ | & & | \end{bmatrix} \tag{5}$$

, where $a_i, i = 1, 2, ..., n$ are column vectors. We note that the set $g_{\mathrm{GS}}(X)$ is a Stiefel manifold [3]. Now for the mapping f_{GS} to the original space, we can define the following Gram-Schmidt-like process:

$$f_{\mathrm{GS}}\left(\begin{bmatrix} | & & | \\ a_1 & \ldots & a_{n-1} \\ | & & | \end{bmatrix}\right) = \begin{bmatrix} | & & | \\ b_1 & \ldots & b_n \\ | & & | \end{bmatrix} \tag{6}$$

$$b_i = \left[\begin{cases} N(a_1) & \text{if i=1} \\ N(a_i - \sum_{j=1}^{i-1}(b_j \cdot a_i)b_j) & \text{if } 2 \le i < n \\ \det \begin{bmatrix} | & & | & e_1 \\ b_1 & \ldots & b_{n-1} & \vdots \\ | & & | & e_n \end{bmatrix} & \text{if } i = n. \end{cases} \right]^T \tag{7}$$

Here $N(\cdot)$ denotes a normalization function, the same as before, and $e_1, \ldots, e_n$ are the n canonical basis vectors of the Euclidean space. The only difference of f_{GS} from an ordinary Gram-Schmidt process is that the last column is computed by a generalization of the cross product to n dimensions. Now clearly, g_{GS} is continuous. To check that for every $M \in SO(n)$, $f_{\text{GS}}(g_{\text{GS}}(M)) = M$, we can use induction and the properties of the orthonormal basis vectors in the columns of M to show that the Gram-Schmidt process does not modify the first $n-1$ components. Lastly, we can use theorems for the generalized cross product such as Theorem 5.14.7 of Bloom [8], to show that the last component of $f_{\text{GS}}(g_{\text{GS}}(M))$ agrees with M. Finally, we can define the set D as that where the above Gram-Schmidt-like process does not map back to $SO(n)$: specifically, this is where the dimension of the span of the $n-1$ vectors input to g is less than $n-1$.

6D representation for the 3D rotations: For the 3D rotations, Case 3 gives us a 6D representation. The generalized cross product for b_n in Equation (7) simply reduces to the ordinary cross product $b_1 \times b_2$. We give the detailed equations in Section B in the supplemental document. We specifically note that using our 6D representation in a network can be beneficial because the mapping f_{GS} in Equation (7) ensures that the resulting 3x3 matrix is orthogonal. In contrast, suppose a direct prediction for 3x3 matrices is used. Then either the orthogonalization can be done in-network or as a postprocess. If orthogonalization is done in network, the last 3 components of the matrix will be discarded by the Gram-Schmidt process in Equation (7), so the 3x3 matrix representation is effectively our 6D representation plus 3 useless parameters. If orthogonalization is done as a postprocess, then this prevents certain applications such as forward kinematics, and the error is also higher as shown in Section 5.

Group operations such as multiplication: Suppose that the original space is a group such as the rotation group, and we want to multiply two representations $r_1, r_2 \in R$. In general, we can do this by first mapping to the original space, multiplying the two elements, and then mapping back: $r_1 r_2 = g(f(r_1)f(r_2))$. However, for the proposed representation here, we can gain some computational efficiency as follows. Since the mapping to the representation space in Equation (5) drops the last column, when computing $f(r_2)$, we can simply drop the last column and compute the product representation as the product of an $n \times n$ and an $n \times (n-1)$ matrix.

Case 4: Further reducing the dimensionality for the n dimensional rotations. For $n \geq 3$ dimensions, we can reduce the dimension for the representation in the previous case, while still keeping a continuous representation. Intuitively, a lower dimensional representation that is less redundant could be easier to learn. However, we found in our experiments that the dimension reduced representation does not outperform the Gram-Schmidt-like representation from Case 3. However, we still develop this representation because it allows us to show that continuous rotation representations can outperform discontinuous ones.

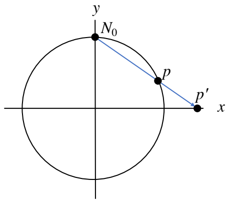


Figure 3. An illustration of stereographic projection in 2D. We are given as input a point p on the unit sphere S^1. We construct a ray from a fixed projection point $N_0 = (0, 1)$ through p and find the intersection of this ray with the plane $y = 0$. The resulting point p' is the stereographic projection of p.

We show that we can perform such dimension reduction using one or more stereographic projections combined with normalization. We show an illustration of a 2D stereographic projection in Figure 3, which can be easily generalized to higher dimensions. Let us first normalize the input point, so it projects to a sphere, and then stereographically project the result using a projection point of $(1, 0, \ldots, 0)$. We call this combined operation a *normalized projection,* and define it as $P : \mathbb{R}^m \to \mathbb{R}^{m-1}$:

$$P(u) = \left[\begin{array}{cccc} \frac{v_2}{1-v_1}, & \frac{v_3}{1-v_1}, & \ldots, & \frac{v_m}{1-v_1} \end{array} \right]^T, v = u/||u||. \tag{8}$$

Now define a function $Q : \mathbb{R}^{m-1} \to \mathbb{R}^m$, which does a stereographic un-projection:

$$Q(u) = \frac{1}{||u||} \left[\begin{array}{cccc} \frac{1}{2}(||u||^2 - 1), & u_1, & \ldots, & u_{m-1} \end{array} \right]^T \tag{9}$$

Note that the un-projection is not actually back to the sphere, but in a way that coordinates 2 through m are a unit vector. Now we can use between 1 and $n-2$ normalized projections on the representation from the previous case, while still preserving continuity and one-to-one behavior.

For simplicity, we will first demonstrate the case of one stereographic projection. The idea is that we can flatten the representation from Case 3 to a vector and then stereographically project the last $n+1$ components of that vector. Note that we intentionally project as few components as possible, since we found that nonlinearities introduced by the projection can make the learning process more difficult. These nonlinearities are due to the square terms and the division in Equation (9). If u is a vector of length m, define the slicing notation $u_{i:j} = (u_i, u_{i+1}, \ldots, u_j)$, and $u_{i:} = u_{i:m}$. Let $M_{(i)}$ be the ith column of matrix M. Define a vectorized representation $\gamma(M)$ by dropping the last column of M like in Equation (5): $\gamma(M) = [M_{(1)}^T, \ldots, M_{(n-1)}^T]$. Now we can define the mapping to the representation space as:

$$g_{\text{P}}(M) = [\gamma_{1:n^2-2n-1}, P(\gamma_{n^2-2n:})] \tag{10}$$

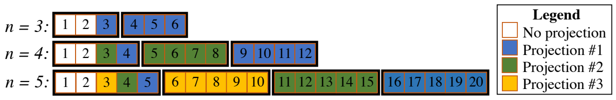


Figure 4. An illustration of how $n-2$ normalized projections can be made to reduce the dimensionality for the representation of $SO(n)$ from Case 3 by $n-2$. In each row we show the dimension n, and the elements of the vectorized representation $\gamma(M)$ containing the first $n-1$ columns of $M \in SO(n)$. Each column is length n: the columns are grouped by the thick black rectangles. Each unique color specifies a group of inputs for the "normalized projection" of Equation (8). The white regions are not projected.

Here we have dropped the implicit argument M to γ for brevity. Define the mapping to the original space as:

$$f_{\mathrm{P}}(u) = f_{\mathrm{GS}}\left(\left[u_{1:n^2-2n-1}, Q(u_{n^2-2n:})\right]^{(n\times(n-1))}\right) \tag{11}$$

Here the superscript $(n \times (n-1))$ indicates that the vector is reshaped to a matrix of the specified size before going through the Gram-Schmidt function f_{GS}. We can now see why Equation (9) is normalized the way it is. This is so that projection followed by un-projection can preserve the unit length property of the basis vector that is a column of M, and also correctly recover the first component of $Q(\cdot)$, so we get $f_{\mathrm{P}}(g_{\mathrm{P}}(M)) = M$ for all $M \in SO(3)$. We can show that g_{P} is defined on its domain and continuous by using properties of the orthonormal basis produced by the Gram-Schmidt process g_{GS}. For example, we can show that $\gamma \neq 0$ because components 2 through $n+1$ of γ are an orthonormal basis vector, and $N(\gamma)$ will never be equal to the projection point $[1, 0, \ldots, 0, 0]$ that the stereographic projection is made from. It can also be shown that for all $M \in SO(3)$, $f_{\mathrm{P}}(g_{\mathrm{P}}(M)) = M$. We show some of these details in the supplemental material.

As a special case, for the 3D rotations, this gives us a 5D representation. This representation is made by using the 6D representation from Case 3, flattening it to a vector, and then using a normalized projection on the last 4 dimensions.

We can actually make up to $n-2$ projections in a similar manner, while maintaining continuity of the representation, as follows. As a reminder, the length of γ, the vectorized result of the Gram-Schmidt process, is $n(n-1)$: it contains $n-1$ basis vectors each of dimension n. Thus, we can make $n-2$ projections, where each projection $i = 1, \ldots, n-2$ selects the basis vector $i+1$ from $\gamma(M)$, prepends to it an appropriately selected element from the first basis vector of $\gamma(M)$, such as γ_{n+1-i}, and then projects the result. The resulting projections are then concatenated as a row vector along with the two unprojected entries to form the representation. Thus, after doing $n-2$ projections, we can obtain a continuous representation for $SO(3)$ in $n^2 - 2n + 2$ dimensions. See Figure 4 for a visualization of the grouping of the elements that can be projected.

Other groups: $O(n)$, similarity transforms, quaternions. In this paper, we focus mainly on the representation of rotations. However, we note that the preceding representations can easily be generalized to $O(n)$, the group of orthogonal $n \times n$ matrices M with $MM^T = M^TM = I$. We can also generalize to the similarity transforms, which we denote as $\mathrm{Sim}(n)$, defined as the affine maps $\rho(x)$ on $\mathbb{R}^n$, $\rho(x) = \alpha Rx + u$, where $\alpha > 0$, R is an $n \times n$ orthogonal matrix, and $u \in \mathbb{R}^n$ [4]. For the orthogonal group $O(n)$, we can use any of the representations in Case 3 or 4, but with an additional component in the representation that indicates whether the determinant is +1 or -1. Then the Gram-Schmidt process in Equation (7) needs to be modified slightly: if the determinant is -1 then the last vector b_n needs to be negated. Meanwhile, for the similarity transforms, the translation component u can easily be represented as-is. The matrix component αR of the similarity transform can be represented using any of the options in Case 3 or 4. The only needed change is that the Gram-Schmidt process in Equations 7 or 11 should multiply the final resulting matrix by α. The term α is simply the norm of any of the basis vectors input to the Gram-Schmidt process, e.g. $||a_1||$ in Equation (7). Clearly, if the projections of Case 4 are used, at least one basis vector must remain not projected, so α can be determined. In the supplemental material, we also explain how one might adapt an existing network that outputs 3D or 4D rotation representations so that it can use our 6D or 5D representations.

5. Empirical Results

We investigated different rotation representations and found that those with better continuity properties work better for learning. We first performed a sanity test and then experimented on two real world applications to show how continuity properties of rotation representations influence the learning process.

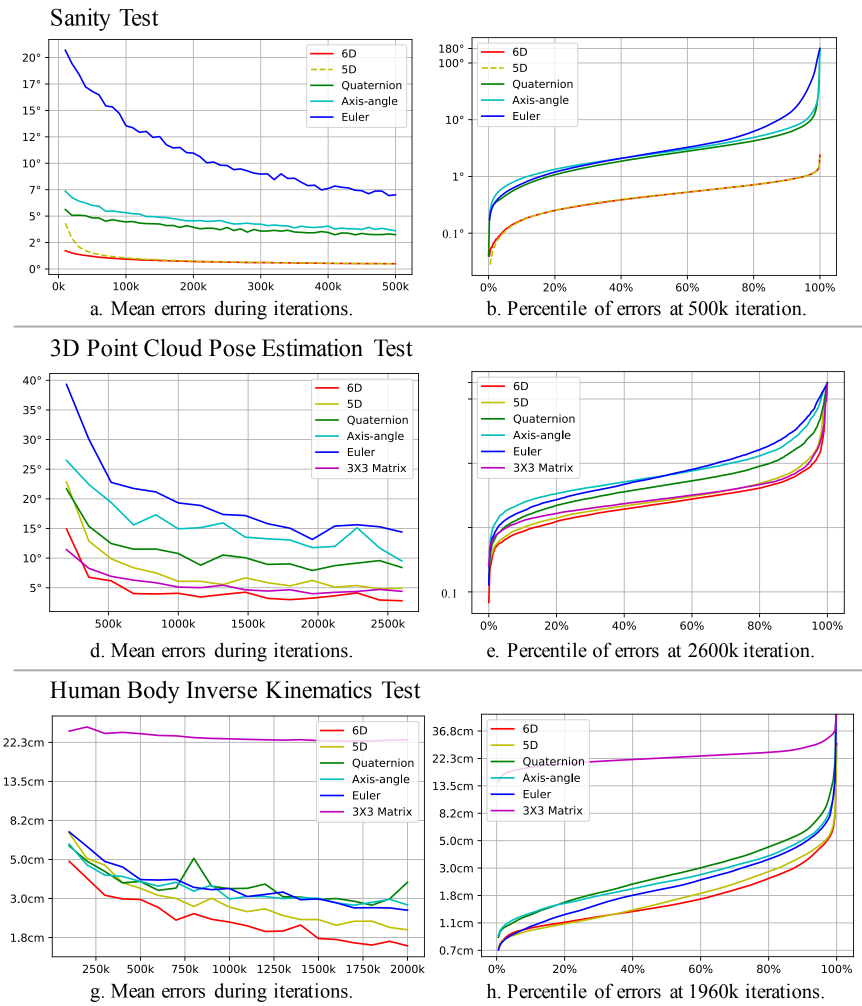
5.1. Sanity Test

We first perform a sanity test using an auto-encoder structure. We use a multi-layer perceptron (MLP) network as an encoder to map $SO(3)$ to the chosen representation R. We test our proposed 6D and 5D representations, quaternions, axis-angle, and Euler angles. The encoder network contains four fully-connected layers, where hidden layers have 128 neurons and Leaky ReLU activations. The fixed "decoder" mapping $f : R \mapsto SO(3)$ is defined in Section 4.

For training, we compute the loss using the L2 distance between the input $SO(3)$ matrix M and the output $SO(3)$ matrix M': note that this is invariant to the particular representation used, such as quaternions, axis-angle, etc. We use Adam optimization with batch size 64 and learning rate 10^{-5} for the first 10^4 iterations and 10^{-6} for the remaining iterations. For sampling the input rotation matrices during training, we uniformly sample axes and angles. We test the networks using 10^5 rotation matrices generated by randomly sampling axes and angles and calculate geodesic errors between the input and the output rotation matrices. The geodesic error is defined as the minimal angular difference between two rotations, written as

$$L_{\text{angle}} = \cos^{-1}((M''_{00} + M''_{11} + M''_{22} - 1)/2) \tag{12}$$

$$M'' = MM'^{-1} \tag{13}$$



a. Mean errors during iterations.

b. Percentile of errors at 500k iteration.

d. Mean errors during iterations.

e. Percentile of errors at 2600k iteration.

g. Mean errors during iterations.

h. Percentile of errors at 1960k iterations.

	Mean(°)	Max(°)	Std(°)
6D	**0.49**	**1.98**	**0.27**
5D	**0.49**	1.99	**0.27**
Quat	3.32	179.93	5.97
AxisA	3.69	179.22	5.99
Euler	6.98	179.95	17.31

c. Errors at 500k iteration.

	Mean(°)	Max(°)	Std(°)
6D	**2.85**	179.83	**9.16**
5D	4.78	179.87	12.25
Quat	9.03	179.66	16.33
AxisA	11.93	179.7	21.35
Euler	14.13	179.67	23.8
Matrix	4.21	180.0	9.44

f. Errors at 2600k iteration.

	Mean(cm)	Max(cm)	Std(cm)
6D	**1.9**	**28.7**	**1.2**
5D	2.0	33.3	1.4
Quat	3.3	87.1	3.1
AxisA	3.0	120.0	2.3
Euler	2.7	48.7	2.1
Matrix	22.9	53.6	4.0

i. Errors at 1960k iteration.

Figure 5. Empirical results. In (b), (e), (h) we plot on the x axis a percentile p and on the y axis the error at the given percentile p.

Figure 5(a) illustrates the mean geodesic errors for different representations as training progresses. Figure 5(b) illustrates the percentiles of the errors at 500k iterations. The results show that the 6D and 5D representations have similar performance with each other. They converge much faster than the other representations and produce smallest mean, maximum and standard deviation of errors. The Euler angle representation performs the worst, as shown in Table (c) in Figure 5. For the quaternion, axis angle and Euler angle representations, the majority of the errors fall under 25°, but certain test samples still produce errors up to 180°. The proposed 6D and 5D representations do not produce errors higher than 2°. We conclude that using continuous rotation representations for network training leads to lower errors and faster convergence.

In Appendix G.2, we report additional results, where we trained using a geodesic loss, uniformly sampled $SO(3)$, and compared to 3D Rodriguez vectors and quaternions constrained to one hemisphere [20]. Again, our continuous representations outperform common discontinuous ones.

5.2. Pose Estimation for 3D Point Clouds

In this experiment, we test different rotation representations on the task of estimating the rotation of a target point cloud from a reference point cloud. The inputs of the networks are the reference and target point clouds $P_\mathrm{r}, P_\mathrm{t} \in \mathbb{R}^{N\times3}$, where N is the number of points. The network output is the estimated rotation $R \in \mathbb{R}^D$ between P_r and P_t, where D is the dimension of the chosen representation.

We employ a weight-sharing Siamese network where each half is a simplified PointNet structure [27], $\Phi : \mathbb{R}^{N\times3} \mapsto \mathbb{R}^{1024}$. The simplified PointNet uses a 4-layer MLP of size $3 \times 64 \times 128 \times 1024$ to extract features for each point and then applies max pooling across all points to produce a single feature vector z. One half of the Siamese network maps the reference point cloud to a feature vector $z_\mathrm{r} = \Phi(P_\mathrm{r})$ and the other half maps the target point cloud to $z_\mathrm{t} = \Phi(P_\mathrm{t})$. Then we concatenate z_r and z_t and pass this through another MLP of size $2048 \times 512 \times 512 \times D$ to produce the D dimensional rotation representation. Finally, we transform the rotation representations to $SO(3)$ with one of the mapping functions f defined in Section 4.

We train the network with 2,290 airplane point clouds from ShapeNet [9], and test it with 400 held-out point clouds augmented with 100 random rotations. At each training iteration, we randomly select a reference point cloud and transform it with 10 randomly-sampled rotation matrices to get 10 target point clouds. We feed the paired reference-target point clouds into the Siamese network and minimize the L2 loss between the output and the ground-truth rotation matrices.

We trained the network with 2.6×10^6 iterations. Plot (d) in Figure 5 shows the mean geodesic errors as training progresses. Plot (e) and Table (f) show the percentile, mean, max and standard deviation of errors. Again, the 6D representation has the lowest mean and standard deviation of errors with around 95% of errors lower than 5°, while Euler representation is the worst with around 10% of errors higher than 25°. Unlike the sanity test, the 5D representation here performs worse than the 6D representation, but outperforms the 3D and 4D representations. We hypothesize that the distortion in the gradients caused by the stereographic projection makes it harder for the network to do the regression. Since the ground-truth rotation matrices are available, we can directly regress the 3×3 matrix using L2 loss. During testing, we use the Gram-Schmidt process to transform the predicted matrix into $SO(3)$ and then report the geodesic error (see the bottom row of Table (f) in Figure 5). We hypothesize that the reason for the worse performance of the 3×3 matrix compared to the 6D representation is due to the orthogonalization post-process, which introduces errors.

5.3. Inverse Kinematics for Human Poses

In this experiment, we train a neural network to solve human pose inverse kinematics (IK) problems. Similar to the method of Villegas et al. [30] and Hsu et al. [18], our network takes the joint positions of the current pose as inputs and predicts the rotations from the T-pose to the current pose. We use a fixed forward kinematic function to transform predicted rotations back to joint positions and penalize their L2 distance from the ground truth. Previous work for this task used quaternions. We instead test on different rotation representations and compare their performance.

The input contains the 3D positions of the N joints on the skeleton marked as $P = (p_1, p_2, p_3, ..., p_N)$, $p_i = (x, y, z)^\intercal$. The output of the network are the rotations of the joints in the chosen representation $R = (r_1, r_2, r_3, ..., r_N), r_i \in \mathbb{R}^D$, where D is the dimension of the representation.

We train a four-layer MLP network that has 1024 neurons in hidden layers with the L2 reconstruction loss $L = ||P - P'||_2^2$, where $P' = \Pi(T, R)$. Here Π is the forward kinematics function which takes as inputs the "T" pose of the skeleton and the predicted joints rotations, and outputs the 3D positions of the joints. Due to the recursive computational structure of forward kinematics, the accuracy of the hip orientation is critical for the overall skeleton pose prediction and thus the joints adjacent to the hip contribute more weight to the loss (10 times higher than other joints).

We use the CMU Motion Capture Database [25] for training and testing because it contains complex motions like dancing and martial arts, which cover a wide range of joint rotations. We picked in total 865 motion clips from 37 motion categories. We randomly chose 73 clips for testing and the rest for training. We fix the global position of the hip so that we do not need to worry about predicting the global translation. The whole training set contains 1.14×10^6 frames of human poses and the test set contains 1.07×10^5 frames of human poses. We train the networks with 1,960k iterations with batch size 64. During training, we augmented the poses with random rotation along the y-axis. We augmented each instance in the test set with three random rotations along the y-axis as well. The results, as displayed in subplots (g), (h) and (i) in Figure 5, show that the 6D representation performs the best with the lowest errors and fastest convergence. The 5D representation has similar performance as the 6D one. On the contrary, the 4D and 3D representations have higher average errors and higher percentages of big errors that exceed 10 cm.

We also perform the test of using the 3×3 matrix without orthogonalization during training and using the Gram-Schmidt process to transform the predicted matrix into SO(3) during testing. We find this method creates huge errors as reported in the bottom line of Table (i) in Figure 5. One possible reason for this bad performance is that the 3×3 matrix may cause the bone lengths to scale during the forward kinematics process. In Appendix G.1, we additionally visualize some human body poses for quaternions and our 6D representation.

6. Conclusion

We investigated the use of neural networks to approximate the mappings between various rotation representations. We found empirically that neural networks can better fit continuous representations. For 3D rotations, the commonly used quaternion and Euler angle representations have discontinuities and can cause problems during learning. We present continuous 5D and 6D rotation representations and demonstrate their advantages using an auto-encoder sanity test, as well as real world applications, such as 3D pose estimation and human inverse kinematics.

7. Acknowledgements

We thank Noam Aigerman, Kee Yuen Lam, and Sitao Xiang for fruitful discussions; Fangjian Guo, Xinchen Yan, and Haoqi Li for helping with the presentation. This research was conducted at USC and Adobe and was funded by in part by the ONR YIP grant N00014-17-S-FO14, the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, the Andrew and Erna Viterbi Early Career Chair, the U.S. Army Research Laboratory (ARL) under contract number W911NF-14-D-0005, Adobe, and Sony. This project was not funded by Pinscreen, nor has it been conducted at Pinscreen or by anyone else affiliated with Pinscreen. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

[1] Cayley transform. https://en.wikipedia.org/wiki/Cayley_transform#Matrix_map.

[2] Rotation matrix. https://en.wikipedia.org/wiki/Rotation_matrix#Quaternion.

[3] Stiefel manifold. https://en.wikipedia.org/wiki/Stiefel_manifold.

[4] C. Allen-Blanchette, S. Leonardos, and J. Gallier. Motion interpolation in sim (3). 2014.

[5] M. J. Baker. Maths: Conversion matrix to quaternion. http://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToQuaternion/. Accessed: 2018-11-21.

[6] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

[7] S. Belongie. Rodrigues' rotation formula. http://mathworld.wolfram.com/RodriguesRotationFormula.html. Accessed: 2019-04-04.

[8] D. M. Bloom. *Linear algebra and geometry*. CUP Archive, 1979.

[9] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[10] Z. Chen and F. Cao. The construction and approximation of neural networks operators with gaussian activation function. *Mathematical Communications*, 18(1):185–207, 2013.

[11] A. Csiszar, J. Eilers, and A. Verl. On solving the inverse kinematics problem using neural networks. In *Mechatronics and Machine Vision in Practice (M2VIP), 2017 24th International Conference on*, pages 1–6. IEEE, 2017.

[12] D. M. Davis. Embeddings of real projective spaces. *Bol. Soc. Mat. Mexicana (3)*, 4:115–122, 1998.

[13] T.-T. Do, M. Cai, T. Pham, and I. Reid. Deep-6dpose: Recovering 6d object pose from a single rgb image. *arXiv preprint arXiv:1802.10367*, 2018.

[14] L. Falorsi, P. de Haan, T. R. Davidson, N. De Cao, M. Weiler, P. Forré, and T. S. Cohen. Explorations in homeomorphic variational auto-encoding. *arXiv preprint arXiv:1807.04689*, 2018.

[15] G. Gao, M. Lauri, J. Zhang, and S. Frintrop. Occlusion resistant object rotation regression from point cloud segments. *arXiv preprint arXiv:1808.05498*, 2018.

[16] F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of graphics tools*, 3(3):29–48, 1998.

[17] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[18] H.-W. Hsu, T.-Y. Wu, S. Wan, W. H. Wong, and C.-Y. Lee. Quatnet: Quaternion-based head pose estimation with multiregression loss. *IEEE Transactions on Multimedia*, 2018.

[19] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] A. Kendall, R. Cipolla, et al. Geometric loss functions for camera pose regression with deep learning. In *Proc. CVPR*, volume 3, page 8, 2017.

[21] A. Kendall, M. Grimes, and R. 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.

[22] C. Kosniowski. *A first course in algebraic topology*. CUP Archive, page 53, 1980.

[23] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[24] B. Llanas, S. Lantarón, and F. J. Sáinz. Constructive approximation of discontinuous functions by neural networks. *Neural Processing Letters*, 27(3):209–226, 2008.

[25] M. LLC. Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu/.

[26] X. Perez-Sala, L. Igual, S. Escalera, and C. Angulo. Uniform sampling of rotations for discrete and continuous learning of 2d shape models. In *Robotic Vision: Technologies for Machine Learning and Vision Applications*, pages 23–42. IGI Global, 2013.

[27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[28] A. Saxena, J. Driemeyer, and A. Y. Ng. Learning 3-d object orientation from images. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 794–800. IEEE, 2009.

[29] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on computer vision and pattern recognition (CVPR)*, volume 5, page 6, 2017.

[30] R. Villegas, J. Yang, D. Ceylan, and H. Lee. Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8639–8648, 2018.

[31] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems (RSS)*, 2018.

[32] Z. Xu and F. Cao. The essential order of approximation for neural networks. *Science in China Series F: Information Sciences*, 47(1):97–112, 2004.

[33] Z.-B. Xu and F.-L. Cao. Simultaneous lp-approximation order for neural networks. *Neural Networks*, 18(7):914–923, 2005.

[34] X. Zhou, X. Sun, W. Zhang, S. Liang, and Y. Wei. Deep kinematic pose regression. In *European Conference on Computer Vision*, pages 186–201. Springer, 2016.