Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

RESEARCH ARTICLE

Learning Rotations

Alberto Pepe[1] | Joan Lasenby[1] | Pablo Chacón[2]

[1]Signal Processing & Communications Lab, Engineering Department, University of Cambridge, UK
[2]Chacon Lab, Rocasolano Physical Chemistry Institute, Madrid, Spain

Correspondence
Alberto Pepe, Signal Processing & Communications Lab, CUED, Trumpington Street, Cambridge, UK. Email: ap2219@cam.ac.uk

Present Address
CUED, Trumpington Street, Cambridge, UK

Abstract

Many problems in computer vision today are solved via deep learning. Tasks like pose estimation from images, pose estimation from point clouds or structure from motion can all be formulated as a regression on rotations.

However, there is no unique way of parametrizing rotations mathematically: matrices, quaternions, axis-angle representation or Euler angles are all commonly used in the field. Some of them, however, present intrinsic limitations, including discontinuities, gimbal lock or antipodal symmetry. These limitations may make the learning of rotations via neural networks a challenging problem, potentially introducing large errors.

Following recent literature, we propose three case studies: a sanity check, a pose estimation from 3D point clouds and an inverse kinematic problem. We do so by employing a full geometric algebra (GA) description of rotations. We compare the GA formulation with a 6D continuous representation previously presented in the literature in terms of regression error and reconstruction accuracy. We empirically demonstrate that parametrizing rotations as bivectors outperforms the 6D representation. The GA approach overcomes the continuity issue of representations as the 6D representation does, but it also needs fewer parameters to be learned and offers an enhanced robustness to noise.

GA hence provides a broader framework for describing rotations in a simple and compact way that is suitable for regression tasks via deep learning, showing high regression accuracy and good generalizability in realistic high-noise scenarios.

KEYWORDS:
geometric algebra; bivectors; rotation representation; computer vision; deep learning; inverse kinematics; pose estimation

1 | INTRODUCTION

1.1 | Background

Computer vision has not been immune to the disruptive power of deep learning [1, 2]. In the last decade, many computer vision tasks traditionally considered difficult have been solved through neural networks (NNs) reaching unprecedented accuracy, including image classification [3, 4], image segmentation [5] and depth estimation [6].

In problems such as pose prediction and generation, motion capture and inverse kinematics, NNs are generally trained to perform a regression on rotations [7, 8]. For these tasks, it is common to describe 3D rotations through 3 or 4D representations,

including axis-angle representation [9], Euler angles [10] and quaternions [11]. However, it has been shown that 3 and 4D representations present some limitations. Grassia et al. [12], for example, demonstrated how an exponential parametrization of rotations is more suitable for differentiation and integration with respect to Euler angles or quaternions. Zhou et al. [13] found the source of large regression errors in the discontinuity of these representations, in which a representation is said to be discontinuous when the mapping $g : SO(3) \rightarrow \mathbb{R}^D$ from the 3×3 rotation matrix in $SO(3)$ onto a given representation space is a discontinuous function. Saxena et al. [14] also highlighted representation discontinuity as a critical factor.

On the other hand, according to the universal approximation theorem, a NN with one hidden layer can approximate any *continuous* function for inputs that fall within a specific range. Moreover, the process of training requires extensive computations of derivatives. Hence, trying to learn rotations when parametrized with a $discontinuous$ representation might lead to large errors.

1.2 | 6D representations

How many parameters, or degrees of freedom (DoF), are enough to represent a rotation so that it can be learned successfully? A possible solution is to increase the DoF to overcome the discontinuity issue. Especially in robotics, it is not uncommon to describe 3D rotations and translations with a total of 6 DoF [15, 16, 17]. In [18], for example, a convolutional neural network (CNN) approach to 6D pose estimation is proposed for cluttered scenes understanding.

In [13], 5 and 6D continuous representations have been proposed specifically for deep learning approaches to computer vision in order to represent rotations exclusively (i.e. without translation component). In particular, the 6D representation of 3D rotations provides a substantial improvement to commonly used 3 and 4D representations. This 6D representation can be visualized as a 3×2 matrix with columns a_1, a_2, which, after a Gram-Schmidt-like orthogonalization procedure, maps to the original rotation matrix in $SO(3)$ with orthonormal columns b_1, b_2, b_3 (Eq. 1).

$$g^{-1}\left(\begin{bmatrix} | & | \\ a_1 & a_2 \\ | & | \end{bmatrix}\right) \mapsto \begin{bmatrix} | & | & | \\ b_1 & b_2 & b_2 \\ | & | & | \end{bmatrix} \in SO(3) \tag{1}$$

However, in problems in which a large amount of rotational information has to be stored and processed, a parametrization with many DoF makes the problem computationally expensive. This is the case in López et al. [19], in which one distance and five angles are used to represent the orientation between amino acids in protein chains in order to produce a 6D joint probability distribution: representations with more DoF are informative and less error-prone, but at the same time they are cumbersome and memory consuming.

1.3 | Contributions

In this paper, we reproduce the experiments in [13] employing a full GA formulation. We show that representing rotations as bivectors and measuring the regression error in terms of rotors yield comparable - if not superior - results to the originally proposed 6D representation when learning rotations. We empirically show that a GA description constitutes a more elegant, compact and general way of describing rotation than the 6D representation, which we believe has some fundamental limitations.

Firstly, the 6D representation has more DoF than commonly employed representations, meaning more parameters to be learned. Secondly, it is fully learned. This guarantees continuity, but it also means that no closed form expression exists to convert a rotation from $SO(3)$ into its 6D representation. Third, it requires an *ad hoc* loss function to be minimized, namely the L2 norm between the original and predicted rotation matrices after the aforementioned orthogonalization procedure, that has to be performed at every iteration. This loss function is less intuitive and more computationally expensive than those normally used in regression tasks. Lastly, no conditions are imposed to avoid $a_1 = b_1; a_2 = b_2$ i.e. that the 6D representation is learned to be equal to the first two columns of the rotation matrix. This identity mapping is easier to "learn" and indeed a possible scenario, but makes the comparison with other representations biased.

The rest of the paper is structured as follows: in Section 2 the basics of rotations in GA are introduced. In Section 3 three experiments, namely the sanity check, the pose estimation from 3D point clouds and the inverse kinematics problem, are introduced in detail and results are presented. Lastly, in Section 4, conclusions are drawn.

2 | GEOMETRIC ALGEBRA

The branch of mathematics called Geometric Algebra was developed in the second half of the 19th century by William Kingdon Clifford [20]. GA was revived in 1966 thanks to Hestenes, who reiterated the unifying power of GA and its relevance in physics due to the close ties with geometry [21]. GA provides an intuitive and unified framework for describing geometric entities and transformations acting on them, and it has found application in computer graphics, computer vision, robotics, and physics. In this section, the fundamental concepts to understand rotations in GA are briefly presented. For a more detailed discussion see [22].

2.1 | Fundamentals

A geometric algebra of size m is defined over a scalar field and a set of m independent basis vectors $\{e_i\}_{i=1,...m}$. There are 2^m elements in a geometric algebra of size m. A general geometric algebra is defined as $\mathcal{G}_{p,q,r}$, with p basis vectors that square to 1, q basis vectors that square to -1 and r basis vectors that square to 0.

Elements in a GA have a *grade* associated with it, and elements of each grade can be multiplied or added together. A linear combination of elements in GA is called a *multivector*, that can have multiple grades associated to it. By grade we define the dimension of the hyperplane an object specifies. e.g. scalars are grade 0, vectors are grade 1, bivectors are grade 2, etc. Geometric algebra gets its name from the geometric product. The geometric product between two vectors is given by

$$ab = a \cdot b + a \wedge b \tag{2}$$

in which the scalar (or *inner*) product $a \cdot b$ is the usual scalar product of linear algebra equal to the cosine of the angle between a and b, while the wedge (or *outer*) product $a \wedge b$ produces a bivector (e.g. an oriented plane). The geometric product between two vectors is hence the sum of a scalar and a bivector, that have different grade. Any multivector of a unique grade r that can be defined as $A = a_1 \wedge a_2 \wedge ... \wedge a_r$ is called a blade. The reversion operator for a multivector is given by $\tilde{A}$. The general rule to reverse an r-blade is given by

$$\tilde{A}_r = (-1)^{\frac{r(r+1)}{2}} A_r \tag{3}$$

The scalar part of the geometric product between a multivector and its reverse gives the squared magnitude: $|A|^2 = \langle A\tilde{A}\rangle_0$. The grade projector operator is denoted by $\langle A\rangle_r$, where r is the grade we want to extract from A. This comes from the fact that a multivector in an m dimensional algebra can be written as

$$A = \sum_{i=0}^{m} \langle A\rangle_i \tag{4}$$

2.2 | Rotations

We focus our attention on $\mathcal{G}_{3,0,0}$, which is the GA of Euclidean space. It contains $2^3 = 8$ elements, a scalar, three vectors (e_1, e_2, e_3), three bivectors (e_{12}, e_{23}, e_{13}, where $e_{ij} = e_1 \wedge e_j$) and one trivector ($e_{123} = e_1 \wedge e_2 \wedge e_3 = I_3$) called the pseudoscalar, that is the element with the highest grade in the algebra.

The reflection of a vector a in the plane orthogonal to a unit vector n is given by $a' = a_\perp - a_\parallel$, where $a_\perp$ and $a_\parallel$ are the perpendicular and parallel components of a with respect to n, respectively. This is done as $a = n^2 a = n(n \cdot a + n \wedge a) = a_\parallel + a_\perp$, in which $a_\parallel = a \cdot (nn)$ and $a_\perp = (nn) \wedge a$. This is equivalent to

$$a' = nn \wedge a - a \cdot (nn) = -n \cdot (an) - n \wedge (an) = -nan \tag{5}$$

The formula $a = -nan$ works for spaces in any dimension and it leaves lengths and angles unchanged. For bivectors $B = a \wedge b$, we have that

$$B' = (-nan) \wedge (-nbn) = (nannbn - nbnnan) = n(ab - ba)n = nBn \tag{6}$$

Sandwiching a multivector between a vector always preserves the grade of the multivector.

A rotation in the plane is generated by successive reflections in the planes perpendicular to two unit vectors m and n. Let up consider two reflections: $b = -mam$, and then $c = -nbn = -n(-mam)n = nmamn$. We define $R = nm$, so that $c = Ra\tilde{R}$: R is called a rotor and represents a rotation operator. A rotor is the geometric product of two unit vectors. Expanding the geometric product we have $R = nm = n \cdot m + n \wedge m = \cos(\theta) + n \wedge m$, where θ is the angle between vectors n, m. The magnitude of the bivector part is given by $-\sin^2(\theta)$. By defining the unit bivector B in the $m \wedge n$ plane as $B = (m \wedge n)/\sin(\theta)$ with $B^2 = -1$, we have that $R = \cos(\theta) - B\sin(\theta)$. It can be seen that B "behaves" like a unit imaginary, hence from the Euler formula we can express

the rotor as $R = \exp(-B\theta)$, in which the exponential function is defined by its Taylor expansion where the multiplications are replaced by the geometric product.

Summarizing: for a rotation of an angle θ, we have a corresponding rotor $R = \exp(-B\theta/2)$. The rotation operation on a vector a is:

$$a' = Ra\tilde{R} = e^{-B\theta/2} a e^{B\theta/2} \tag{7}$$

where $R\tilde{R} = 1$. The same rule applies when rotating a multivector of any grade. As R is a geometric product between two unit vectors, we need four real numbers to parametrize it as $R = (\alpha, \mathbf{b}) = (\alpha, b_{12}, b_{23}, b_{13})$, where α is the scalar part and b_{ij} are the coefficients of the bivectors e_{ij}. Similarly, a bivector B in $\mathcal{G}_{3,0,0}$ can be parametrized with three real numbers (c_{12}, c_{23}, c_{13}). In this paper, we will focus on representing rotations in the Euclidean space in terms of rotors (4D representation) and bivectors (3D representation), and on two kinds of rotor-to-bivector maps, namely (i) the exponential map, for which $B = -2 \log R$ and (ii) the Cayley transform, for which $B = (1 - R)/(1 + R)$.

3 | EXPERIMENTS

3.1 | Sanity Check

3.1.1 | Reproducing the experiment

In this section we reproduce and validate results in [13], confirming the claim that discontinuous representation lead to large error when learning rotations.

The goal of the sanity check is to use a multi-layer perceptron (MLP) to learn the mapping from rotation matrix $M \in SO(3)$ to a given representation $\mathcal{R} \in \mathbb{R}^D$, in which D is the number of DoF associated to $\mathcal{R}$, or its dimensionality. Then the representation $\mathcal{R}$ is converted back to an estimate M' and the closeness between M' and M is measured. The MLP architecture is shown in Figure 1 . We first tested Zhou's 6D representation along with quaternions, axis-angle, Euler angles and bivectors.

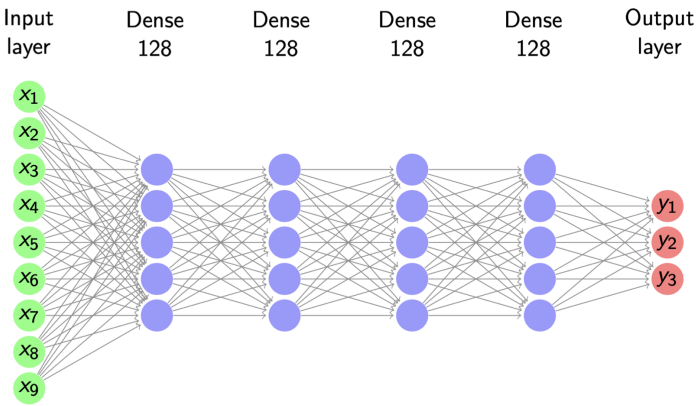

FIGURE 1 The network employed for the sanity check. Zhou's approach: learning a representation (3 or 4D) from the rotation matrix (9D).

The network we employed has been kept unchanged from [13], with four dense layers with 128 neurons each and a Leaky ReLU activation function between them. As a loss function, we adopted either the mean squared error (mse) or the mean absolute error (mae) between the target and input representation, defined as:

$$mse = \frac{1}{D} \sum_{i=1}^{D} (y_i - y'_i)^2 \tag{8}$$

$$mae = \frac{1}{D} \sum_{i=1}^{D} \|y_i - y'_i\| \tag{9}$$

where D, y and y' are the *representation* dimensionality, original and predicted *representations*, respectively. In the case in which y is a bivector, for example, then the y_i's will be the coefficients b_{12}, b_{23}, b_{13} of the bivectors in $\mathcal{G}_{3,0,0}$; if y are Euler angles, then the y_i's will correspond to the angles α, β, γ. We adopted the mse as it is simple and commonly used in regression tasks. Only for the 6D representation we employed the L2 distance between rotation matrix M and predicted matrix M', defined as:

$$L2 = \sqrt{\sum_{i=1}^{3}\sum_{j=1}^{3}(y_{ij} - y'_{ij})^2} \tag{10}$$

where $y \equiv M$, $y' \equiv M'$ are the original and predicted rotation *matrices*, respectively. Note that, according to its definition, the 6D representation can only have the L2 norm between M, M' as a loss. This is due to two reasons: firstly, because the main feature of the 6D representation, i.e. that the rotation matrix is obtained through orthogonalization of the 6D representation, is implemented within the loss function itself. Secondly, because there is no closed form expression to represent a rotation matrix as its 6D counterpart (in other words, the mapping $g : SO(3) \to \mathbb{R}^6$ is fully learned). This means that the ground truth, when learning the 6D representation, cannot be expressed in 6D, but only as the original rotation matrix. Note that in [13] all the experiments share the L2 norm as a loss function. We believe that this is not only a more complex approach, as the mapping $g^{-1} : \mathbb{R}^D \to SO(3)$ has to be evaluated at each iteration, but it is also penalizing discontinuous representations (i.e. those representations for which g is discontinuous): we want to demonstrate that these representation can be learned successfully when this mapping is omitted from the learning strategy. The optimizer has been kept as Adam, the batch size as 64 and the number of epochs as 100.

The results from [13] have been reproduced and validated on a set of 10^5 rotation matrices with a 70-30 train-test split. The maximum, mean and standard deviation of the geodesic error measured on the testing set are shown in Table 1 , along with the loss function employed during training and the kind of error employed. The geodesic error is defined as

$$L^M = \arccos\left(\frac{tr(M'') - 1}{2}\right) \tag{11}$$

Where $M'' = MM'^T$. Table 1 shows that the 6D representation indeed overcomes the discontinuity issue and outperforms significantly the other representations, including the bivector representation that we tested in addition to the others. Note how, when using mse between quaternions, the matrix-to-quaternion/rotor learning does not converge to a minimum, most likely due to the antipodal symmetry of the representation, meaning that quaternions q and $-q$ represent the same rotation [23]. Our results, however, are in agreement with Zhou's when employing the same L2 loss.

TABLE 1 Average Geodesic Error (°), Sanity Check, $M \mapsto \mathcal{R}$

Representation	Loss	Error	Max	Mean	STD
Euler	mse	L^M	179.67	5.93	8.68
Axis-Angle	mse	L^M	179.69	4.22	8.79
Quaternion/Rotor	mse	L^M	179.99	80.26	44.83
Quaternion/Rotor	L2	L^M	179.40	5.29	8.90
Bivector $\left(\frac{1-R}{1+R}\right)$	mse	L^M	179.11	4.49	8.54
6D	L2	L^M	**2.64**	**1.06**	**0.41**
Euler (Zhou et al.)	L2	L^M	179.95	6.98	17.31
Axis Angle (Zhou et al.)	L2	L^M	179.22	3.69	5.99
Quaternion (Zhou et al.)	L2	L^M	179.93	3.32	5.97
6D (Zhou et al.)	L2	L^M	**1.98**	**0.49**	**0.27**

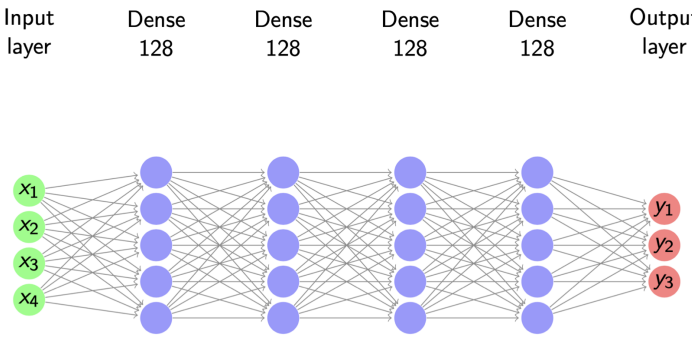

FIGURE 2 The network employed for the sanity check. Our approach: learning the bivector (3D) from the rotor (4D).

3.1.2 | GA-based Sanity Check

We believe that the errors shown in Table 1 are specific to the kind of mapping learned rather than intrinsic to the representation itself. The maximum 180° error, said to come from the discontinuity of the representation, in fact, is present only when adopting Eq.11 as a metric and when the representation is learned starting from the rotation matrix. We now proceed to show that the maximum geodesic error is much smaller when keeping the rotations in GA.

We trained the same network to learn the bivector B from the rotor R, either as its Cayley transform or as its logarithm, and then converted back to estimated rotor R', where $R' = e^{-B/2}$ or $R' = (1 - B)/(1 + B)$ (See Figure 2). We measured the regression error without passing through the matrix representation, i.e. as the angle between the original and predicted rotors R, R', respectively:

$$L^R = \arccos\langle R\tilde{R}'\rangle_0 \tag{12}$$

Results are shown in Table 2 . Note how comparable results between the 6D representation and the bivector are obtained when expressing the bivector as the Cayley transform of the rotor, with a mean geodesic error of 1.03° for the rotor-to-bivector mapping and of 1.06° for the matrix-to-6D mapping. The bivector, however, only requires 3 DoF to be learned instead of 6. Note also how the same bivector yields a mean geodesic error about three times smaller when learned from the rotor (in Table 2) than when learned from the rotation matrix (in Table 1)

TABLE 2 Average Geodesic Error (°), Sanity Check, $R \mapsto B$

Representation	Loss	Error	Max	Mean	STD
Bivector $(-2 \log R)$	mse	L^R	156.05	1.55	5.04
Bivector $\left(\frac{1-R}{1+R}\right)$	mse	L^R	28.41	7.49	4.43
Bivector $\left(\frac{1-R}{1+R}\right)$	mae	L^R	16.78	**1.03**	0.53
6D	L2	L^M	**2.64**	1.06	**0.41**

3.2 | Pose Estimation from 3D Point Clouds

For this task, we trained a network that takes as input a reference and a target point cloud $P_r, P_t \in \mathbb{R}^{N\times 3}$ respectively, where N is the number of points per point cloud, to predict a representation $\mathcal{R} \in \mathbb{R}^D$ of the rotation between the two, where D is the dimensionality of the representation. These include the rotation matrix ($D = 9$), the 6D representation ($D = 6$), Euler angles ($D = 3$), the axis-angle representation ($D = 3$), the quaternion/rotor ($D = 4$) and the bivector ($D = 3$).

The network is a slightly modified version of PointNet [24], see Figure 3 . We assume that the registration between the point clouds is known, i.e. we know which points in P_r correspond to which in P_t. The input of this network is then the concatenated point clouds $[P_r, P_t]$ of size $N \times 6$, passed through five layers of 1D convolutions of size $6 \times 64 \times 128 \times 256 \times 1024$, and then through three more dense layers of size $512 \times 512 \times D$, where D is the dimensionality of $\mathcal{R}$. Concatenating the inputs prevents us using the Siamese structure of PointNet, that assumes separate inputs through two nets with shared weights instead. This choice was made to facilitate the learning procedure.

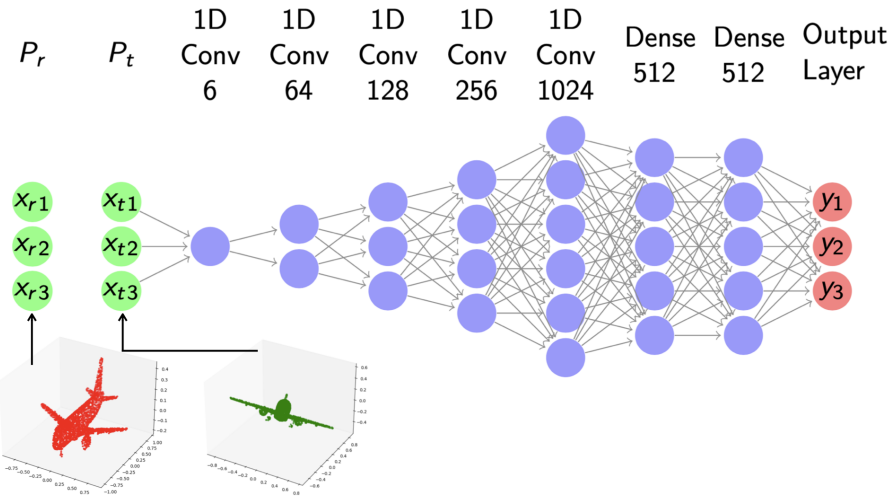

FIGURE 3 The modified PointNet for the pose estimation from 3D point clouds problem. Input: concatenated reference and target point clouds $P_r, P_t \in \mathbb{R}^{N\times 3}$. Output: rotation from P_r to P_t with D parameters.

The optimizer has been kept as Adam, the batch size as 32, the epoch number to 100 and the loss to mae, that for this specific task we empirically verified that it performed better than mse. A Leaky ReLU activation function has been employed between layers, with the exception of the last layer that uses a linear activation function. The dataset is composed of 726 airplane point clouds taken from the ModelNet40 dataset [25], with 626 training and 100 testing samples. Each point cloud is composed of $N = 3000$ points. To each of the 726 point clouds is associated a random rotation matrix from which we obtained the 8 different representations tested.

Results averaged over five experiments are summarized in Table 3 , ordered by increasing mean geodesic error along with the number of DoF of the representation. Firstly, it can be noticed that using mae instead of the L2 norm between M and M', yields a lower error for the matrix representation compared to the 6D representation, as opposed to findings in [13]. It is intuitive to think that more DoF are more informative than fewer DoF. Secondly, that the 180° maximum error is present only in some of the measurements, and not consistently, for all the representations analyzed. This leads to a maximum geodesic error averaged over five experiments always below 180°, also for those representations defined discontinuous by Zhou et al. (e.g. axis-angle, Euler angles, etc.), for which we would expect an error around 180°. We believe that the difference between our results and Zhou's results rely in the different learning strategy and mapping learned, not in the type of representation.

It can be seen that the bivector representation, expressed as the logarithm of the rotor R that rotates P_r to P_t has very similar geodesic error distribution to the 6D representation, with half the number of DoF and simpler loss function. Examples of the rotated point clouds through the bivector and the 6D method are given in Figures 4 - 5 , while the Euclidean distance between the target point clouds and the point cloud rotated via predicted representations in the testing set is presented in Table 4 . Recall that point clouds are normalized to fit the unit cube with center in the origin of the reference frame, reason for which a numerically small Euclidean distance might still correspond to an erroneous prediction.

These results show that a pure geometric algebra description with 3 DoF is almost as good as a representation with 6 DoF, with a simpler loss function employed and an equivalent geodesic error formulation measured exclusively through rotors, without the need to employ rotation matrices.

TABLE 3 Average Geodesic Error (°), Pose Estimation from 3D Point Clouds.

Representation	Loss	Error	Max	Mean	STD	DoF
Matrix	mae	L^M	89.32	7.66	12.10	9
6D	L2	L^M	120.94	8.36	12.00	6
Bivector $(-2\log R)$	mae	L^R	145.93	9.15	15.25	3
Bivector $\left(\frac{1-R}{1+R}\right)$	mae	L^R	89.36	11.02	12.87	3
Axis-Angle	mae	L^M	121.55	14.24	17.69	3
Quaternion	mae	L^M	165.63	15.52	22.60	4
Rotor	mae	L^R	101.60	17.38	18.00	4
Euler Angles	mae	L^M	111.59	23.82	21.16	3
6D (Zhou et al.)	L2	L^M	179.83	2.85	9.16	6
Matrix (Zhou et al.)	L2	L^M	180.0	4.21	9.44	9
Quaternion (Zhou et al.)	L2	L^M	179.66	9.03	16.33	4
Angle Axis (Zhou et al.)	L2	L^M	179.70	11.93	21.35	3
Euler Angles (Zhou et al.)	L2	L^M	179.67	14.13	23.8	3

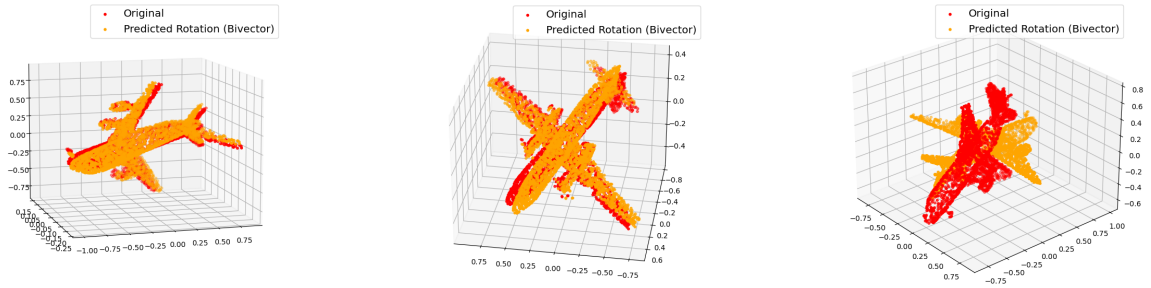

FIGURE 4 Original test point cloud and rotated via predicted bivector representation. Left: best case. Centre: average case. Right: worst case. Note how the worst case matches the maximum geodesic error in Table 3 .

TABLE 4 Per-point Euclidean distance between original and predicted point clouds for the two best performing representations.

Representation	Min	Mean	Max	STD
Bivector $(-2\log R)$	0.00039	0.0020	0.0164	0.00193
6D	0.000091	0.0009	0.0092	0.0001

3.2.1 | Noisy Point Clouds

We then studied the case in which Gaussian noise is added to the point clouds. In a real case scenario, in fact, it is safe to assume imperfect acquisition of samples from a scene (e.g., due to a shaky camera). We restricted the analysis to the two representations with lowest error in the noiseless case, namely the (exponential) bivector and the 6D. We added noise to the reference and target point clouds P_r, P_t, respectively. Noise has been modelled as additive white Gaussian noise (AWGN) $\sim \mathcal{N}(0, \sigma^2)$ with variable $\sigma = \{0.1; 1; 2; 5\}$. The geodesic error measured for the two representations on the testing set is given in Figure 6 . Note also how little noise ($\sigma = 0.1$) has a regularizing effect, and hence lower geodesic error, when compared to the noiseless case [26, 27, 28].

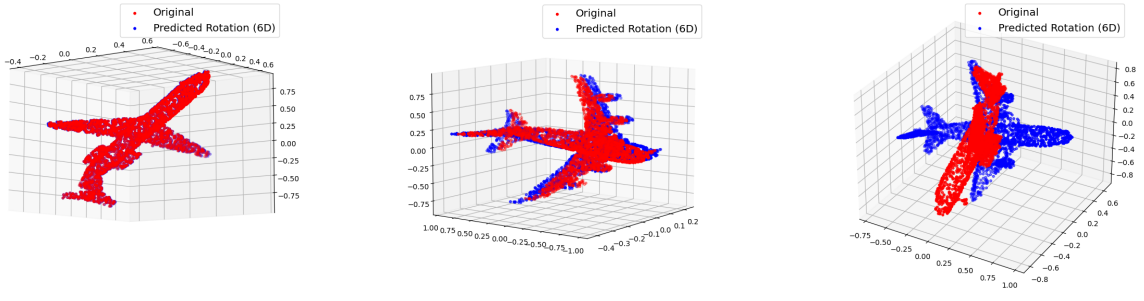

FIGURE 5 Original test point cloud and rotated via predicted 6D representation. Left: best case. Centre: average case. Right: worst case. Note how the worst case matches the maximum geodesic error in Table 3 .

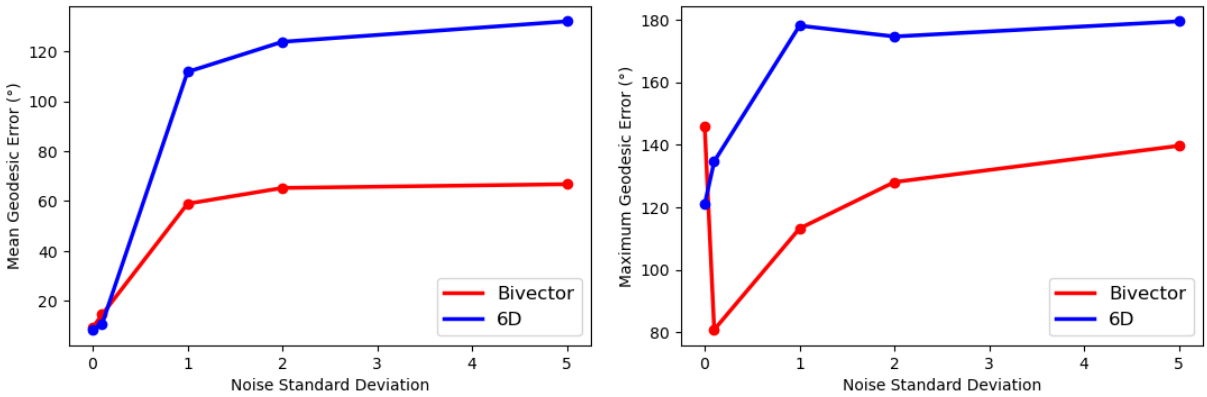

FIGURE 6 Average (left) and maximum (right) geodesic error as a function of the standard deviation of the noise applied on P_r, P_t

It can be seen how, in high noise scenarios, the network likely fits to the noise when trying to predict more parameters of the 6D representation instead of the 3 DoF of the bivector. The maximum and average geodesic errors are noticeably higher for the 6D representation than for the bivector case as noise is increased. This is also mirrored in the Euclidean distance between original and predicted point clouds, with a generally low difference between the 6D and the bivector representations that tends to decrease in high-noise scenario (see Figure 7).

The 6D representation is very accurate with little-to-no noise, but it fails when the standard deviation of the noise increases. The bivector representation, on the other hand, is more tolerant to high noise conditions, showing a consistently lower geodesic error at a fraction of the required DoF and with a simpler loss function employed during training.

3.3 | Inverse Kinematics

Lastly, we trained a NN to solve the inverse kinematics problem. The goal is to estimate the rotation from the T-pose to any arbitrary pose of a skeleton. By T-pose we define the skeleton in standing position and arms stretched out that is commonly used in computer graphics for calibration purposes. The input of the network takes the 3D positions of the $N = 31$ joints of a skeleton as $P = (p_1, p_2, p_3, ..., p_N)$, where $p_i = (x, y, z)^T$. The output of the network are the rotations that a T-pose skeleton undergoes to reach the input position. The output is expressed as $R = (r_1, r_2, r_3, ..., r_N)$, with $r_i \in R^D$, where D is the dimension of the representation. The rotations are hierarchical, in the sense that the rotation of each joint depends on the previous one and on how they are linked to each other. The adopted NN is a four-layer MLP with 1024 neurons per layer (see Figure 8). The problem is

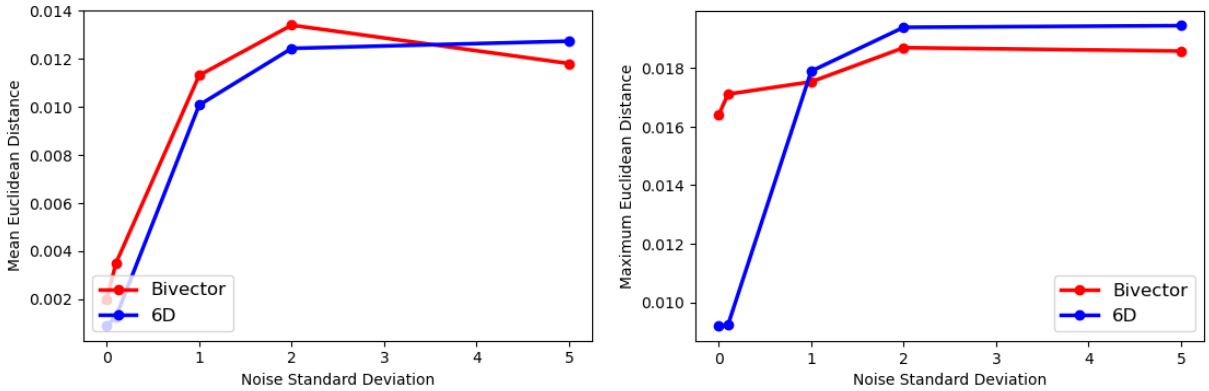

FIGURE 7 Average (left) and maximum (right) Euclidean distance as a function of the standard deviation of the noise applied on P_r, P_t

formulated as a supervised learning problem, that allows us to use mse loss consistently with the only exception being the 6D representation.

We used 760 clips from the CMU Motion Capture Database (MoCap) [29]. From these clips we selected 10000 frames, using 3300 of these for testing. We centered the position of the root joint (defined as the midpoint between the two hips) the origin so as to predict solely the rotational component of the joints and not the translation of the skeleton. The network has been trained for 100 epochs with batch size 64. The average geodesic error between ground truth (from the testing set) and predicted rotation and Euclidean distance of the predicted poses from the ground truth are displayed in ascending order in Tables 5 - 6 , respectively.

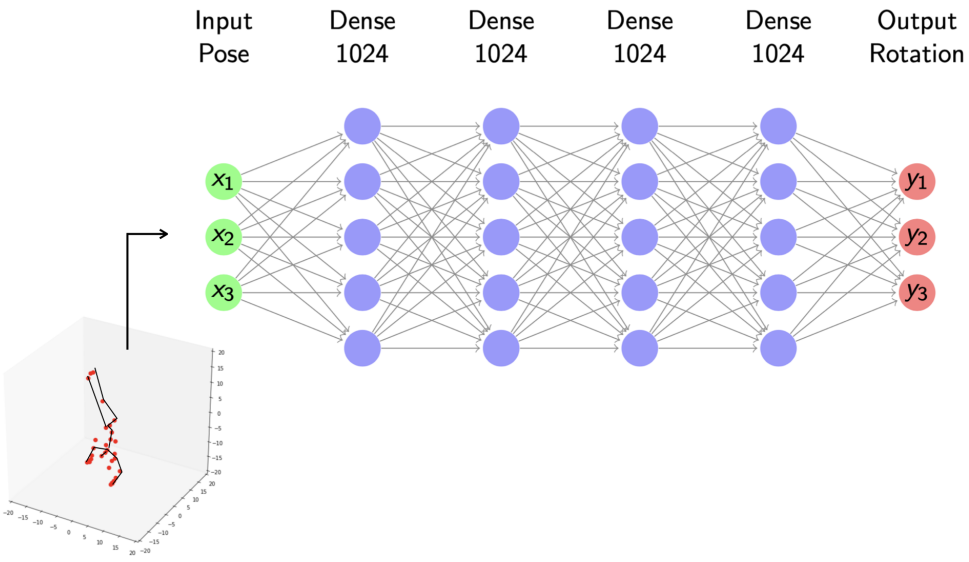

FIGURE 8 The network for the inverse kinematic problem. Input: frames with 31×3 spatial coordinates. Output: 31 rotations with D parameters.

Firstly, it can be noticed that there is no perfect correspondence between a low geodesic error and an accurate motion estimation. Secondly, the difference in Euclidean distance between different representations is small and at most 1.5 cm. The small difference between the representations is probably due to the supervised formulation of the problem. The bivector expressed as the Cayley transform of the rotor allowed the second best pose estimation after the rotation matrix. The rotation matrix has been proven to be consistently the best representation for all three experiments, as expected due to the higher number of DoF. Examples of reconstructed poses for the 6D and bivector representation are given in Figures 9 - 10 .

TABLE 5 Average Geodesic Error (°), Inverse Kinematics.

Representation	Loss	Error	Max	Mean	STD
Bivector $\left(\frac{1-R}{1+R}\right)$	mse	L^R	170.92	37.51	31.90
Matrix	mse	L^M	180.00	38.04	57.95
6D	L2	L^M	179.98	58.67	45.91
Axis-Angle	mse	L^M	179.97	58.01	40.61
Bivector, $-2\log R$	mse	L^R	165.76	58.33	28.82
Euler Angles	mse	L^M	180.00	59.89	41.50
Quaternion	mse	L^M	179.99	62.76	46.00

TABLE 6 Average Euclidean Distance (cm), Inverse Kinematics.

Representation	Loss	Error	Max	Mean	STD
Matrix	mse	L^M	8.07	4.34	0.81
Bivector $\left(\frac{1-R}{1+R}\right)$	mse	L^R	8.09	4.83	0.83
6D	L2	L^M	9.24	5.45	1.12
Axis-Angle	mse	L^M	9.36	5.63	0.89
Quaternion	mse	L^M	9.16	5.66	0.96
Euler Angles	mse	L^M	9.36	5.78	0.89
Bivector, $-2\log R$	mse	L^R	9.26	6.99	0.65

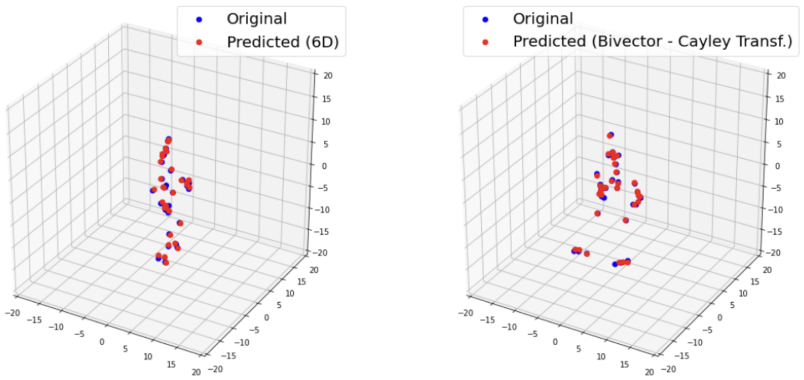

FIGURE 9 Best pose reconstruction in the testing set for the 6D and bivector representations. As expected, best reconstruction is attested for poses that don't deviate excessively from the T-pose.

4 | CONCLUSIONS

In this paper, we explored the issue of suitable rotation representations in learning problems. We have shown that different representations might yield significantly different results in rotation regression problems solved via machine learning. We have proven that certain representations might be more suitable than others according to the specific task to be solved. We verified

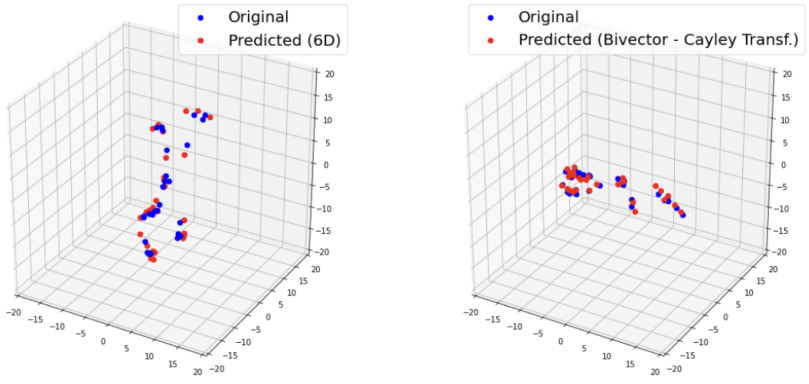

FIGURE 10 Worst pose reconstruction in the testing set for the 6D and bivector representations. As expected, worst reconstruction is attested for poses that greatly deviate from the T-pose.

that the issue of continuity is indeed present when considering the mapping from the 3×3 rotation matrix to the representation space, and that this issue implies a maximum geodesic error of 180° when predicting rotations. However, a fully GA-based description of the rotation allows us to bypass the continuity issue, yielding results which are comparable, if not superior, to the continuous 6D representation proposed in [13].

The GA approach allows us to have high accuracy on rotation regressions in all of the three toy problems proposed and it presents several advantages. Firstly, it relies on the broader framework of GA, meaning that there is no need to introduce a new, *ad hoc* representation as in [13] specifically crafted for learning problems. This allows a more intuitive problem formulation and a simpler learning strategy, that can employ commonly used loss functions for regression tasks (mse, mae). Despite its simplicity, we empirically verified that a GA approach yields better results with respect to the 6D representation and other commonly used representations in the field with a lower number of parameters to be learned, namely 3 for the bivector as opposed to the 6 of the 6D representation. In addition, bivectors present a closed form expression that can be easily retrieved from the rotor, as opposed to the 6D representation, that is learned and cannot be retrieved analytically from the rotation matrix. Moreover, the bivector representation shows enhanced robustness to noise, as demonstrated in the pose estimation from point clouds. In a realistic scenario, in fact, it is likely that a moving camera might produce noisy measurements. With more DoF, the network may fit to the noise and introduce larger errors.

We believe that the errors introduced when learning rotations are due to the specific rotation matrix-to-representation mapping, and not an intrinsic limitation of the representation itself: the type of problem to be solved, the problem formulation, the adopted learning strategy and loss function employed are more important parameters than the kind of representation used. We believe that representing rotations as bivectors will be particularly useful in complex problems in which a large number of rotations has to be learned, stored or processed.

References

1. Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience* 2018; 2018.

2. O'Mahony N, Campbell S, Carvalho A, et al. Deep learning vs. traditional computer vision. In: Science and Information Conference. ; 2019: 128–144.

3. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 2012; 25: 1097–1105.

4. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* 2014.

5. Minaee S, Boykov YY, Porikli F, Plaza AJ, Kehtarnavaz N, Terzopoulos D. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2021.

6. Khan F, Salahuddin S, Javidnia H. Deep learning-based monocular depth estimation methods—A state-of-the-art review. *Sensors* 2020; 20(8): 2272.

7. Phaniteja S, Dewangan P, Guhan P, Sarkar A, Krishna KM. A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots. In: 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO). ; 2017: 1818–1823.

8. Mathis A, Schneider S, Lauer J, Mathis MW. A primer on motion capture with deep learning: principles, pitfalls, and perspectives. *Neuron* 2020; 108(1): 44–65.

9. Huang Z, Wan C, Probst T, Van Gool L. Deep learning on lie groups for skeleton-based action recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. ; 2017: 6099–6108.

10. Fang Q, Zhao K, Tang D, et al. Euler angles based loss function for camera localization with deep learning. In: 2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE. ; 2018: 61–66.

11. Pavllo D, Grangier D, Auli M. Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485* 2018.

12. Grassia FS. Practical parameterization of rotations using the exponential map. *Journal of graphics tools* 1998; 3(3): 29–48.

13. Zhou Y, Barnes C, Lu J, Yang J, Li H. On the continuity of rotation representations in neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. ; 2019: 5745–5753.

14. Saxena A, Driemeyer J, Ng AY. Learning 3-d object orientation from images. In: 2009 ieee international conference on robotics and automation. IEEE. ; 2009: 794–800.

15. Nissler C, Marton ZC. Robot-to-camera calibration: a generic approach using 6D detections. In: 2017 First IEEE International Conference on Robotic Computing (IRC). IEEE. ; 2017: 299–302.

16. Do T, Pham T, Cai M, Reid I. Real-time monocular object instance 6d pose estimation. 2019.

17. Wang C, Martín-Martín R, Xu D, et al. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE. ; 2020: 10059–10066.

18. Xiang Y, Schmidt T, Narayanan V, Fox D. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199* 2017.

19. López-Blanco JR, Chacón P. KORP: knowledge-based 6D potential for fast protein and loop modeling. *Bioinformatics* 2019; 35(17): 3013–3019.

20. Clifford W. Preliminary Sketch of Biquaternions (1873). *Mathematical Papers* 1882; 658.

21. Hestenes D. *Space-time algebra*. 67. Springer . 2015.

22. Lasenby J, Lasenby AN, Doran CJ. A unified mathematical language for physics and engineering in the 21st century. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 2000; 358(1765): 21–39.

23. Li K, Pfaff F, Hanebeck UD. Grid-based quaternion filter for SO (3) estimation. In: 2020 European Control Conference (ECC). IEEE. ; 2020: 1738–1744.

24. Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. ; 2017: 652–660.

25. Wu Z, Song S, Khosla A, et al. 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. ; 2015: 1912–1920.

26. Karpukhin V, Levy O, Eisenstein J, Ghazvininejad M. Training on synthetic noise improves robustness to natural noise in machine translation. *arXiv preprint arXiv:1902.01509* 2019.

27. Xie Q, Dai Z, Hovy E, Luong MT, Le QV. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848* 2019.

28. Liu X, Si S, Cao Q, Kumar S, Hsieh CJ. How does noise help robustness? explanation and exploration under the neural sde framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. ; 2020: 282–290.

29. Hodgins F, Macey J. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. *CMU-RI-TR-08-22* 2009.