Journal of Field Robotics

WILEY

RESEARCH ARTICLE

Pose Estimation Accuracy Improvement Using Different Orientation Representations With Neural Networks: Case Study for the VIVE HTC Tracker

Sławomir Romaniuk[1] | Milica Petrović[2] | Adam Wolniakowski[1] | Roman Trochimczuk[1] | Grzegorz Masłowski[3]

[1]Department of Automatic Control and Robotics, Faculty of Electrical Engineering, Białystok University of Technology, Białystok, Poland | [2]Department of Production Engineering, Faculty of Mechanical Engineering, University of Belgrade, Belgrade, Serbia | [3]Department of Electrical and Computer Engineering Fundamentals, Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, Rzeszów, Poland

Correspondence: Sławomir Romaniuk (s.romaniuk@pb.edu.pl)

Received: 14 March 2025 | **Revised:** 14 April 2026 | **Accepted:** 13 May 2026

Funding: Ministerstwo Edukacji i Nauki, Grant/Award Number: MEiN/2022/DPI/2577; Science Fund of the Republic of Serbia, Grant/Award Number: 17801; Ministry of Science, Technological Development, and Innovations of the Serbian Government, Grant/Award Number: 451-03-34/2026-03/200105

Keywords: accuracy enhancement | HTC VIVE tracker | learning from demonstration | neural network | polynomial model correction | robot positioning

ABSTRACT

In robotic applications, the HTC VIVE tracker is frequently used for Learning from Demonstration. This solution and similar devices are not industrial-grade, meaning that their accuracy in tracking movements in three-dimensional space needs to be refined to ensure it is sufficient for programming precise robot positioning tasks. Several different methods are often used to improve position tracking accuracy, such as polynomial correction and neural networks. Various methods of parameterizing position and orientation are also frequently used, such as Euler angles or quaternions, although trade-offs between compactness and numerical stability mean that they are suitable for different correction methods. In this study, we explore four different ways of representing the pose orientation (axis-angle representation, quaternion, rotation matrix, Zhou representation) for the purpose of implementing pose tracking accuracy correction for the HTC VIVE tracker. The paper investigates the applicability of these representations for the Neural Network correction methods and compares the results with the classical polynomial correction method. As part of the study, three experiments were conducted involving measurements of the actual and measured positions of the robot and the tracker. An industrial UR5e robot arm from Universal Robots was used as the reference system for collecting measurement data, with an HTC VIVE tracker mounted on its wrist. The obtained results confirm that both the representation and neural network architecture significantly influence calibration effectiveness of HTC VIVE tracker. The results of the experiments showed that the Zhou parametrization of the orientation, combined with neural network rectification, performs best and results in a 17-fold improvement in the pose estimation accuracy of the HTC VIVE tracking system. The PMC algorithm offers a valuable alternative when fast calibration is required, providing significant accuracy improvements with minimal computational cost.

1 | Introduction

Proper applications of industrial robots and cobots (Bi et al. 2021) depend on the correct programming of the robots and their subsystems, regardless of the specific parameters selected to meet task requirements. These parameters include workspace, ability to handle workpieces of different mass and geometry, speed and acceleration during positioning, accuracy and repeatability of the movements of the end-effector, and other relevant parameters. Programming a robot involves not

only the robot itself, but also additional systems such as vision or other sensory systems, conveyors, feeders, safety systems, machine tools, execution systems, and communication systems. A robotics engineer must possess advanced knowledge of programming, establishing process parameters, understanding the geometry of the workpiece, and consideration for the constraints related to the structure of the workcell or the singularities of the robot (Middleton et al. 1989). Currently, programming robots and other components of robotic workcells is typically done using dedicated off-line software environments. The development of a robot program continues to be a time-consuming and costly endeavour, despite the existence of advanced software capabilities designed to facilitate programming activities.

Learning from Demonstration (LfD), or Programming by Demonstration (PbD), (Perez-Villeda et al. 2023; Grollman and Billard 2012; Xi et al. 2019) is a new paradigm in robotics and robot programming. It allows expert operators to use their practical skills in operating tools and moving them relative to objects being processed, without a robot, to build a control program (Englert and Toussaint 2018; Pignat et al. 2022). Unlike classical programming, where the successive positions of the robot are recorded in Point-To-Point (PTP) or Continuous-Path (CP) mode (Craig 2004; Murray et al. 2017), LfD techniques enable quick programming of the robot trajectory with minimal knowledge of programming. The program can be reproduced and edited by adjusting the robot path within a given range of motion, using the same technique as used to create the program (Billard and Grollman 2013; Xie et al. 2020). Approaches to creating robot programs from demonstration fall into three categories: kinesthetic teaching, teleoperation, and passive observation (Ravichandar et al. 2020). Collecting and processing user-demonstrated movements require accurate and low-latency pose tracking (Xiang et al. 2018), which creates a demand for accessible tracking systems. In this regard, synergy can be found with research and advancements in Virtual Reality (VR) and Augmented Reality (AR), where a range of off-the-shelf tracking systems is already available (Lan et al. 2022).

The Mimic Kit system from Nordbo Robotics (Robotics n.d) is a notable approach for translating operator movement skills into a ready-to-use robot program. The Mimic Kit system is based on the HTC VIVE Tracker (Vive n.d), a widely used 6-DoF pose tracker for VR (Virtual Reality) and AR (Augmented Reality) tracking. The tracker uses structured infrared light emitted by the base station to determine its pose. The VIVE system consists of one or more trackers and one or more base stations (Valve n.d). The base stations are arranged in the working area, such as to cover the workspace. The trackers are designed to be attached to the moving objects in that space. The tracker is a small (ca. 8 x 8 x 5 cm) device that contains 20 IR receivers, an IMU, a microcontroller, and a built-in battery, enabling a 7.5 h operating period. The declared range of the tracking system is from 0.5 m to a maximum of 7 m, and the accuracy declared by the producer is in the submillimeter range, while in literature it is reported in the range of several millimeters to a few centimeters, depending on the application and the environmental conditions (Peer et al. 2018; Astad et al. 2019; Kulozik and Jarrassé 2025). The base station contains an IR light pattern emitter. The tracker's pose is determined using the Time-Difference-of-Arrival (TDoA) principle across the diode array and is further augmented with data from the IMU. The tracking system's source code is proprietary, and its pose estimation algorithm is not publicly available.

Some research has been published on the topic of HTC VIVE tracker accuracy and calibration issues. In (Sansone et al. 2021), the accuracy of the earlier version of the system has been tested. In (Holzwarth et al. 2021), the accuracy of the SteamVR tracking system has been compared to the accuracy of the Oculus 2 tracking. In (Bauer et al. 2021), the accuracy of the VR tracking system with one or more base stations has been thoroughly tested in the large workspace. Similar assessment has been performed by (Niehorster et al. 2017). In (Borges et al. 2018), the authors perform a static and dynamic test of VIVE tracking system accuracy and present an open-source set of algorithms for improving the system performance. In (Weber et al. 2023), authors improve the accuracy of the HTC VIVE tracker and Inertia Measurement Units (IMU) up to several millimeters by using the Unscented Kalman Filter as opposed to the Extended Kalman Filter algorithm, increasing the number of base stations, increasing the number of trackers, and fusing IMUs. The paper (Astad et al. 2019) discusses specific issues of the lighthouse tracking ofthe HTC VIVE tracking system and describes automated calibration procedures for centimetric measurement error in the robot cell. In the paper (Wu et al. 2020), the authors proposed a linearized error model for the HTC VIVE tracking system to improve the tracking accuracy. In (Soares et al. 2021), the authors compared the accuracy and repeatability of Microsoft's HoloLens 2 and HTC VIVE device using an OptiTrack system. In (Peer et al. 2018), the authors proposed a simple method for aligning and correcting the tracking of the HTC Vive system, enabling quick coordinate frame calibration and improved positioning accuracy in VR applications. In (Astad et al. 2019), the authors presented a rapid robot cell calibration approach using the HTC Vive tracking system to estimate robot and workspace poses with minimal setup time. In (Zana and Zelei 2021), the authors developed a feedback motion control method for a spatial double-pendulum manipulator based on pose estimation obtained from a swept-laser measurement system.

1.1 | Neural Networks for Calibration and Accuracy Improvement

There is a growing number of neural network applications for calibration and for enhancing position accuracy. Since neural networks can estimate unknown functions, they meet the requirements for such error refinement in measurement models, where the nature of the measurement error is either unknown or difficult to derive using classical mathematical descriptions. With appropriate learning procedures, neural networks can outperform other approaches and solve problems that were previously unknown. For example, numerous neural network approaches are available for vision-based measurement systems. In (Zhang et al. 2022), authors present TAB-IOL, a novel deep neural network-based method for real-time posture tracking of multiple multi-joint fish-like robots. The paper (Lan et al. 2022) presents a comprehensive review of deep learning-based Human Pose Estimation (HPE), covering 2D and

3D methods, applications, challenges, and research trends. The work (Bilal et al. 2022) develops an eye-to-hand camera-based pose estimation system for robotic machining, enhancing accuracy using Long Short-Term Memory (LSTM) neural networks and sparse regression. Through experimental validation on a KUKA KR240 R2900 ultra robot, the proposed methods outperform the Extended Kalman Filter (EKF), significantly improving pose estimation accuracy; the sparse regression approach yields the best results. An introduction to PoseCNN is presented in (Xiang et al. 2018). It's a convolutional neural network for 6D object pose estimation, which predicts 3D translation by localizing object centers and regresses 3D rotation using a quaternion representation. A novel loss function enables handling symmetric objects, and the approach is validated on the newly proposed YCB-Video dataset.

1.2 | Contribution

In general, system accuracy has been assessed across the full range of the tracker's operation. It has been found that due to the closed nature of the tracker software, the systematic errors are hard to predict and correct. Furthermore, due to the apparent utilization of the IMU data in the pose estimation algorithm, the system is susceptible to errors arising from the unpredictable movement of the tracker (e.g., vibration). Another important source of errors is the issue of the Line of Sight (LoS) (Ciężkowski et al. 2020), where the system has to reinitialize upon reacquiring the sight of the base station. For these reasons, in applications that use the VIVE tracker, such as the Mimic Kit by Nordbo Robotics, the user is typically required to perform a calibration procedure whenever undertaking a new demonstration of the movement. In the case of the Mimic Kit, this is required for each separate trajectory demonstration. Furthermore, the system has to be reinitialized every time a line-of-sight breach incident occurs. Since the trajectory demonstration obtained with the HTC VIVE tracker is never perfect, the system provides a mode for the trajectory correction, where the user has a chance to correct the waypoints through teleoperation. The use of the HTC VIVE tracker to accurately estimate a robot's pose in three-dimensional space, given its non-industrial primary function, requires improving its accuracy through various representations and neural networks. In this paper, we focus on the utilization of the HTC VIVE tracker system in an industrial context in small workspaces. We aim to establish a straightforward calibration procedure with a simple error-correction model to minimize the user participation required when demonstrating movements within a limited area. We conduct an HTC VIVE tracker static accuracy assessment and present a method to improve accuracy using a Polynomial Model Correction (PMC) and a Neural Network (NN). The proposed methodology in the article allows for a 17-fold improvement in the accuracy of the HTC VIVE tracking system. It should be noted that the proposed method was validated using a single Universal Robots UR5 manipulator and one HTC Vive tracking setup. Consequently, the presented results do not allow for a statistical assessment of variability across different UR5 units or HTC devices of the same type. Manufacturing tolerances, calibration differences, and device-specific characteristics may influence the achieved accuracy. Therefore, the reported performance should be interpreted as representative for the tested configuration only, and broader generalization would require experiments on multiple robots and tracking systems.

1.3 | Paper Structure

The content of the paper is organized as follows: the first section—Introduction—defines the basic problems of programming industrial and service robots and introduces the reader to the subject of programming robotic systems using LfD techniques, and in particular using the Mimic Kit system from Nordbo Robotics, based on the HTC VIVE tracker sensor. It also defines the thesis's purpose and scope, based on a review of the relevant literature.

The second section, Materials and Methods, presents the theoretical basis of selected methods for describing the representation of rotations and displacements of a material point in three-dimensional space. The methods discussed were also applied in the training of neural networks to improve the accuracy of the HTC VIVE Tracker system's representation of the robot's pose. It also discusses the setup of the lab bench used in the tests with Universal Robots' UR5e cobot, which had the VIVE Tracker sensor attached to its wrist. It describes the cobot control structure adopted, using a specially created RobWorkStudio (Ellekilde and Jorgensen 2010) and ROS (Robot Operating System) environment to explore the digital twin. It also defined the conditions and methodology for carrying out three experiments, from which data were obtained that was used to teach and calibrate neural networks. We have made the dataset publicly available[1].

The third section, Results, presents the experimental results in tabular form. Based on the analysis of the results, conclusions were drawn, and the results discussed with the aim of improving the accuracy of the HTC VIVE Tracker system. We used data collected during an experiment conducted in the laboratory of Białystok University of Technology. We did not find any external data on the VIVE tracker from other researchers that fit the considered scenarios.

The fourth section—Conclusion formulates the final conclusions regarding the improvement of the accuracy and repeatability of the HTC VIVE Tracker and indicates possible directions for further research in the presented topic.

The paper is completed by a list of references used in the preparation of the article and Appendix A1.

2 | Materials and Methods

2.1 | Representation of Orientation

In this section, we present the selected representations for describing the relative orientation between a pair of frames of reference. This property is also often referred to in the literature as rotation—or attitude.

Many representations of the orientation are known, utilized and existent in the literature (Craig 2004; Murray et al. 2017; Spong

et al. 2006). In this study, we focus on several popular approaches with the objective of assessing their suitability in the implementation of a neural network-based system for improving the static accuracy of the IR tracking system. The selected representations and the conversions between them are shown in Figure 1. In the interest of clarity, we introduce these representations in the sections below. The considered representations are (with names in brackets indicating the shorthand for how they will be referred to):

- rotation matrix (R),
- axis-angle representation with a compacted Equivalent Axis-Angle version (EAA),
- quaternion representation (Quat),
- continuous 6D representation introduced in (Zhou et al. 2019) (Zhou).

2.1.1 | *Rotation Matrix*

The rotation matrix, denoted as ${}^{A}_{B}\mathbf{R}$, is the most direct representation of relative orientation between two frames of reference: A and B. In three dimensions it is a 3x3 matrix, the columns of which correspond to the coordinates of the three versors of the B frame of reference expressed in the A coordinate system. Since the versors are necessarily of length 1, from the property of vector dot product, the rotation matrix can also be said to contain the so called direction cosines, representing the angles between the pairs of versors of frames A and B. Therefore, the rotation matrix is sometimes referred to as a Direction Cosine Matrix (DCM).

$$
\begin{aligned}
{}^{A}_{B}\mathbf{R} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \\
&= \begin{bmatrix} {}^{A}\hat{x}_B & {}^{A}\hat{y}_B & {}^{A}\hat{z}_B \end{bmatrix} = \\
&= \begin{bmatrix} \cos(\hat{x}_A, \hat{x}_B) & \cos(\hat{x}_A, \hat{y}_B) & \cos(\hat{x}_A, \hat{z}_B) \\ \cos(\hat{y}_A, \hat{x}_B) & \cos(\hat{y}_A, \hat{y}_B) & \cos(\hat{y}_A, \hat{z}_B) \\ \cos(\hat{z}_A, \hat{x}_B) & \cos(\hat{z}_A, \hat{y}_B) & \cos(\hat{z}_A, \hat{z}_B) \end{bmatrix}
\end{aligned}
\tag{1}
$$

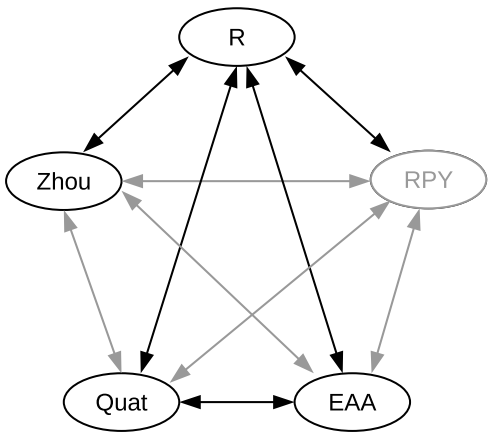

FIGURE 1 | Orientation representations and the conversions between them used in the study. Black indicates the preferred conversions, as described in detail in the paper. Conventions and conversions not used in the study are highlighted in gray.

Since operations with rotation matrices involve matrix multiplication, this representation is particularly well-suited for processing by modern computers, which are typically designed to perform such calculations efficiently.

It should be noted that while in three dimensions the rotation matrix $\mathbf{R}$ consists of 9 elements, these are not independent, since the orientation can be said to only have three degrees of freedom. Any valid rotation matrix $\mathbf{R}$ must meet the orthogonality criterion and the elements of the rotation matrix are therefore redundant. This may cause several issues, such as (a) it requires more memory to store orientation expressed as a rotation matrix, and (b) the representation is more vulnerable to rounding errors due to redundant elements.

In practice, very often rotation matrices are used when direct calculations are necessary, and another representation method is chosen for the storage.

2.1.2 | *Axis-Angle Representation*

The axis-angle representation exploits the fact that any performed sequence of N rotation transformations can be reduced to a single rotation $\mathbf{R}'$ of angle θ around one axis k:

$$
\mathbf{R}_1 \cdot \mathbf{R}_2 \cdot \ldots \cdot \mathbf{R}_N = \mathbf{R}' = \mathrm{AA}([k_x, k_y, k_z], \theta) \tag{2}
$$

The axis-angle representation of orientation $\mathrm{AA}([k_x, k_y, k_z], \theta)$ requires the provision of 4 parameters: x, y and z which are the coordinates of the axis of the revolution and θ which is the angle of the rotation.

The axis-angle representation is still somewhat redundant, and it is very common to compact it further by utilizing the fact that the axis of the rotation can be expressed as a versor, and the angle can be encoded in the length of the vector. Such reduced representation is called the *rotation vector* representation or *Equivalent Axis Angle (EAA)* representation:

$$
\mathrm{AA}([k_x, k_y, k_z], \theta) = \mathrm{EAA}([\theta\hat{k}_x, \theta\hat{k}_y, \theta\hat{k}_z]) \tag{3}
$$

The EAA (and AA) representation has an advantage of being compact and easily intuitively understood. While possible (e.g., by utilizing the Rodrigues formula), it is inconvenient to be directly employed in performing transformation computations. It is therefore often used in storage and data transfer. It is also useful when a direct access to the parameters for the individual degrees of freedom of the orientation is desired.

2.1.3 | *Quaternion Representation*

Quaternion representation of orientation uses an extended complex number system with three distinct imaginary units to define a rotation. Quaternions q can be expressed as:

$$
q = [q_w, q_x, q_y\ q_z]^T = q_w + iq_x + jq_y + kq_z \tag{4}
$$

Quaternions can be used directly in transforming vectors by rotations. Let us consider a sample vector v rotated by a

quaternion q. The resultant transformed vector v' can be computed as:

$$v' = q^{-1} v_q q \tag{5}$$

where v_q is a quaternion obtained from the vector v as $v_q = iv_x + jv_y + kv_z$.

The quaternion representation forms a close relationship with the axis-angle representation, since the imaginary part of the quaternion encodes the axis of the rotation and the real part - the angle of the rotation. Given the axis-angle representation $\mathrm{AA}([k_x, k_y, k_z], \theta)$ the corresponding quaternion can be constructed as:

$$q_x = k_x \sin\frac{\theta}{2} \tag{6}$$

$$q_y = k_y \sin\frac{\theta}{2} \tag{7}$$

$$q_z = k_z \sin\frac{\theta}{2} \tag{8}$$

$$q_w = \cos\frac{\theta}{2} \tag{9}$$

Quaternions offer a compact representation suited for storage and some special operations (such as interpolation) that at the same time does not suffer from the singularities particular to the Euler angle representation (e.g., *gimbal lock*).

2.1.4 | *Zhou Representation*

In (Zhou et al. 2019), a certain novel approach for orientation representation based on a compact rotation matrix was introduced. It was intended particularly for use with neural networks and to deal with singularities and discontinuities present in certain other representations. The authors of the cited research argue and prove that in three dimensions no representation of orientation 4 or fewer parameters is continuous. They introduce certain 5D and 6D representations that show promise in being well conditioned for training neural networks.

In this work, we adopt the 6D representation mentioned above and we will refer to this representation as the *Zhou representation*.

The Zhou representation exploits the redundancy exhibited in the rotation matrix. The representation only requires six parameters, which are obtained directly from the matrix by means of dropping the third column of $\mathbf{R}$. Let us assume that a certain orientation is expressed with a rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{10}$$

The Zhou representation of this orientation can be written as:

$$\mathbf{R} = \mathrm{Zhou}(\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_3, \mathcal{Z}_4, \mathcal{Z}_5, \mathcal{Z}_6) \tag{11}$$

where the $\mathcal{Z}_i$ are the Zhou parameters such that:

$$\begin{bmatrix} \mathcal{Z}_1 & \mathcal{Z}_4 \\ \mathcal{Z}_2 & \mathcal{Z}_5 \\ \mathcal{Z}_3 & \mathcal{Z}_6 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ r_{31} & r_{32} \end{bmatrix} \tag{12}$$

The Equations (11) and (12) define the conversion from the rotation matrix representation to the Zhou representation. The conversion from the Zhou representation $(\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_3, \mathcal{Z}_4, \mathcal{Z}_5, \mathcal{Z}_6)$ to the rotation matrix representation can be achieved through the following procedure. Let us define vectors a_1 and a_2:

$$a_1 = [\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_3]^T \tag{13}$$

$$a_2 = [\mathcal{Z}_4, \mathcal{Z}_5, \mathcal{Z}_6]^T \tag{14}$$

Let us now compute the vectors $b_{1,2,3}$:

$$b_1 = N(a_1) \tag{15}$$

$$b_2 = N(a_2 - (b_1 \cdot a_2) b_1) \tag{16}$$

$$b_3 = b_1 \times b_2 \tag{17}$$

The recovered rotation matrix $\mathbf{R}$ can now be assembled as:

$$\mathbf{R} = [b_1 \quad b_2 \quad b_3] \tag{18}$$

2.2 | Experimental Setup

To assess the accuracy of the VIVE tracker in positioning objects within a limited area of the workcell, we used a reference system consisting of a UR5e robot arm. The UR5e manipulator (Robots n.d) is a well-known industrial arm capable of lifting up to 5 kg with a reach of 850 mm and has a declared accuracy of ±0.1 mm. Since the accuracy of the robot is much higher than the expected accuracy of the tracker, we treat the pose measured by the robot as the ground truth in the further analysis.

The diagram of the experimental setup as visualized in RobWorkStudio (Ellekilde and Jorgensen 2010) is shown in Figure 2. The real experimental setup is shown in Figure 3. The UR5e and the VIVE tracker are both placed on a solid steel table at a distance of ~2 m between each other. The base station is mounted on a tripod at the height of 500 mm above the table and it is directed towards the robot.

The volume of the workspace where the tracker position is acquired is shown in Figure 2 as a red wire cube. The experimental volume is in the form of a cube, the center of which is located at (0, −0.6, 0.5) m in the frame of reference attached to the base of the UR5e robot. This position was chosen to ensure the feasibility of IK solutions and a constant robot posture (elbow up) across the entire volume. A constant robot posture is required in order to provide a stable line of sight to the tracker base station.

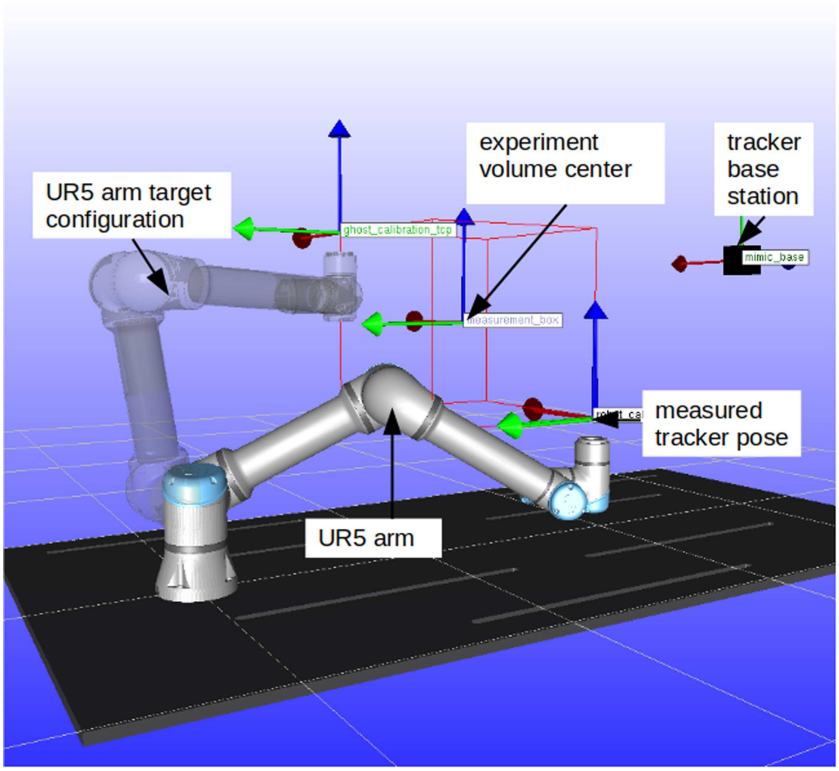

FIGURE 2 | The experimental setup visualization and user interface in RobWorkStudio. [Color figure can be viewed at wileyonlinelibrary.com]

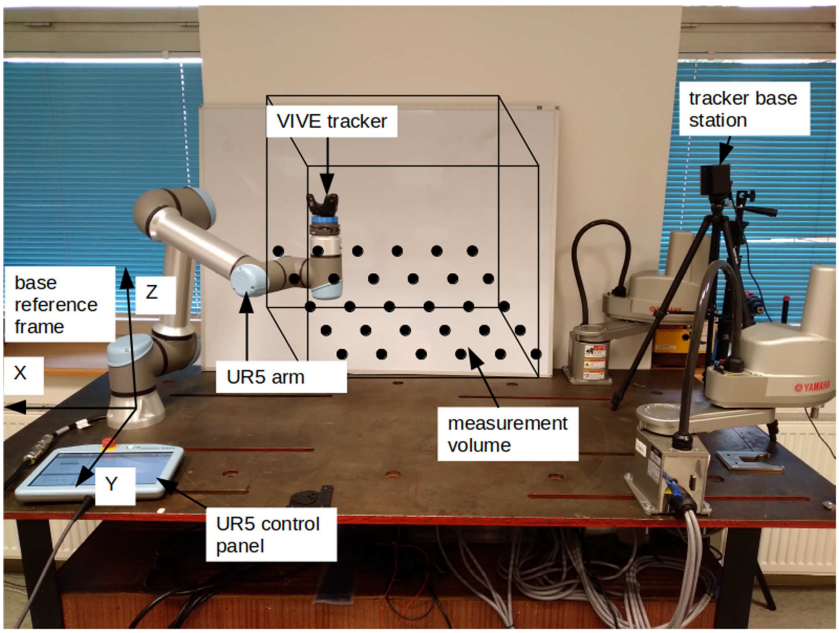

FIGURE 3 | The experimental setup. [Color figure can be viewed at wileyonlinelibrary.com]

We have devised and performed three experiments in which we collected three sets of reference poses and measured poses. For each of these experiments, a number of predetermined sensor placements were performed by the UR5e robot. For each of these positions, the sensor was left in constant pose for 10 s before the recording was done. The experiments were designed to form a series of increasingly more difficult and general positioning tasks:

Experiment I. in this experiment (see Figure 4), the sensor was placed in a series of positions arrayed in a 11 x 11 x 11 lattice within a cube with the side length of 0.4 m. The sensor was kept in a constant upright orientation, and 11 x 11 x 11 = 1331 poses were recorded. This experiment was designed to provide the baseline data for a typical case of translation-only pose calibration.

Experiment II. in this experiment (see Figure 5), the sensor was placed in a series of positions arrayed in a 5 x 5 x 5 lattice within a cube with the side length of 0.4 m. The sensor orientation was upright, but for each position in the lattice it was varied by rotating it around the vertical axis in 13 regular increments in the range of $[0; 2\pi]$. $5 \times 5 \times 5 \times 13 = 1625$ poses were recorded. This experiment was designed to provide a simple dataset involving sensor orientation in addition to translational displacement. We have used this dataset to select the most promising NN architecture for pose calibration.

Experiment III. in this experiment (see Figure 6), the sensor was placed in a series of positions arrayed in a 5 x 5 x 5 lattice within a cube with the side length of 0.2 m. For each of these positions, the sensor orientation was varied around all three axes with the rotation induced using the RPY (Roll-Pitch-Yaw angles) representation. For each of the rotations, we chose the range of $[-30^o; +30^o]$, and each rotation parameter was changed in 5 regularly spaced increments. Thus, $3 \times 3 \times 3 \times 5 \times 5 \times 5 = 3375$ poses were recorded. The size of the measurement volume and the rotation ranges were selected deliberately to avoid collisions and singularities in the workspace of the robot. The number of samples was limited due to the availability of the robot (the experiment took around 10 h to perform). The experiment was designed to provide a more general dataset involving both the translational and the rotational displacements, which was used for the final verification of the tested calibration methods.

The experimental setups described above are presented for clarity in Figures 4–6.

The UR5e robot and the tracker are controlled using a ROS stack, the structure of which is shown in Figure 7. We have a custom ROS driver for the UR5e robot arm[2] and a custom trajectory planning and interpolation library[3] to execute the robot movements. The robot stopped at every sampled point in the workspace for 5 s before a pose estimate was recorded. We have used a custom ROS driver to record the pose measurement[4]. In Figure 7 the ovals with `/ur5`, `/mimic_node` and `/kair_vive_plugin` correspond to nodes for the control of the UR5 robot, HTC Vive tracker and the data collection respectively. The rectanlges correspond to the ROS topics and actions, of which `/ur/joint_states` is used to collect the data from the robot and the `/ur5/follow_joint_trajectory` is used to command the robot movement. The remaining interfaces of the UR5 controller are not utilized.

The diagram of the reference frames used in the experiment is shown in Figure 8. The *base* frame is the frame attached to the robot base. The tracker is attached to the robot end at *tracker* frame of reference. The tracker base station is labeled with

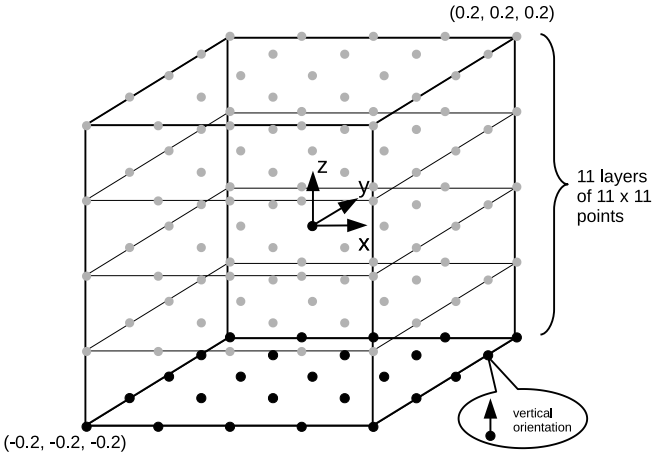

FIGURE 4 | Reference points setup for the experiment I.

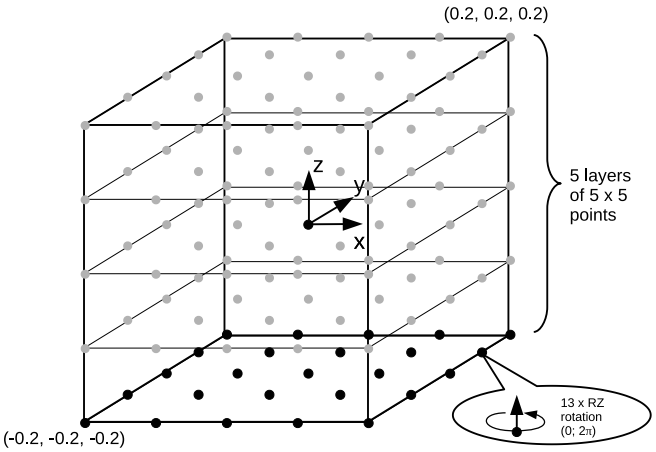

FIGURE 5 | Reference points setup for the experiment III.

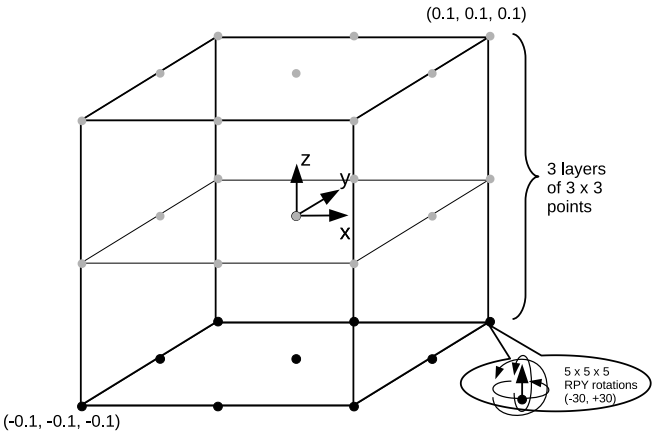

FIGURE 6 | Reference points setup for the experiment II.

station frame. The raw pose measurement obtained from the VIVE system is the ${}^{\text{station}}_{\text{tracker}}T$ pose. We acquire and compare two data sets of poses ${}^{\text{base}}_{\text{tracker}}T$ where the reference value is obtained through the robot kinematics:

$$ {}^{\text{base}}_{\text{tracker}}T_{\text{ref}} = {}^{\text{base}}_{\text{end}}T(q) \cdot {}^{\text{end}}_{\text{tracker}}T \tag{19} $$

The measured pose is obtained through the tracking system:

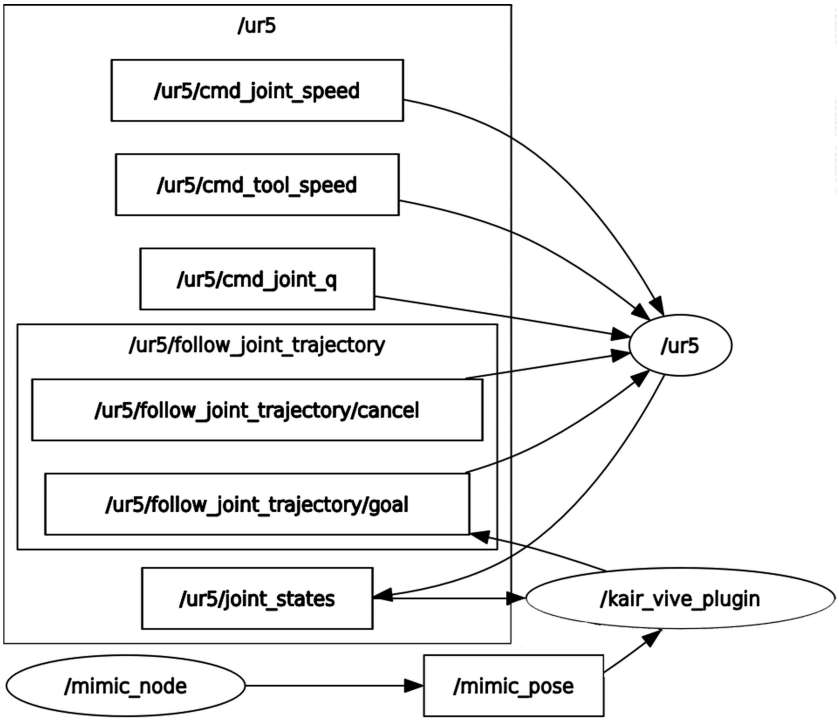

FIGURE 7 | Diagram of the ROS system used for data acquisition.

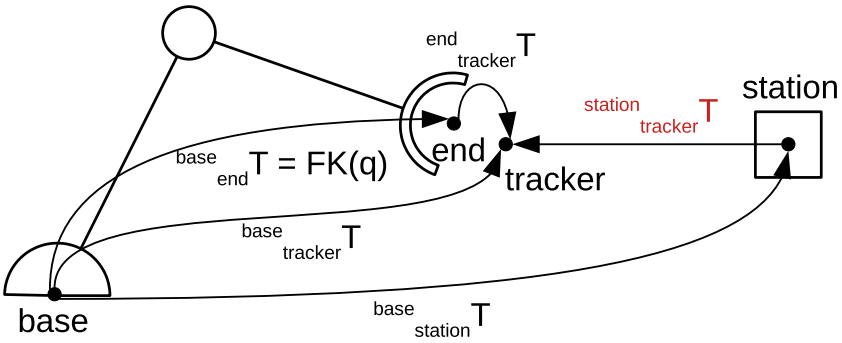

FIGURE 8 | Diagram of the frames of reference used in the experiment. [Color figure can be viewed at wileyonlinelibrary.com]

$$ {}^{\text{base}}_{\text{tracker}}T_{\text{mea}} = {}^{\text{base}}_{\text{station}}T \cdot {}^{\text{station}}_{\text{tracker}}T \tag{20} $$

The tracker is placed in the measurement volume in an array of reference points, where the robot configuration is obtained as:

$$ q = IK\left({}^{\text{base}}_{\text{end}}T \right) = IK\left({}^{\text{base}}_{\text{tracker}}T_{\text{ref}} \cdot {}^{\text{end}}_{\text{tracker}}T^{-1} \right) \tag{21} $$

where IK indicates the inverse kinematics solution for the UR5e robot. The ${}^{\text{end}}_{\text{tracker}}T$ is known due to the structure of the adapter and is equal to:

$$ {}^{\text{end}}_{\text{tracker}}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.042 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{22} $$

The calibration of the pose of the tracker base station wrt. to the robot base is performed on the basis of single measurement before the experiment is launched. We collect the reference pose of the tracker wrt. to the robot base according to the robot kinematics, measured tracker pose according to the base station measurement and the ${}^{\text{base}}_{\text{station}}T$, and the raw pose measurement wrt the tracker base station. The calibration transform ${}^{\text{base}}_{\text{station}}T$ has been found as:

$$ {}^{\text{base}}_{\text{station}}T = \begin{bmatrix} 0.9981 & 0.0125 & 0.0601 & 0.0764 \\ 0.0599 & 0.0160 & -0.9981 & -1.5668 \\ -0.0134 & 0.9998 & 0.0152 & 0.6615 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23} $$

2.3 | Polynomial Model Correction

A Polynomial Model Correction (PMC) method can be used to improve the accuracy of the measurement system. PMC method maps the raw data into the reference set by means of a given model. Different models may be used to fulfil this task. The proposed PMC approach is conceptually related to polynomial-

based error compensation methods used in robot calibration, such as those discussed by (Hollerbach and Wampler 1996), where systematic positioning errors are modeled and reduced using parametric correction functions.

Let's assume the calibrated parameters are given by X. For the purpose of position and orientation rectification, we can state following general model of the PMC method:

$$X' = \mathbf{A}_k \cdot X^T \cdot X + B_k \cdot X + c_k \tag{24}$$

where X' is the rectified vector of the calibrated parameters, $\mathbf{A}_k$ is the coefficient matrix and B_k is the coefficient vector, and c_k is a coefficient constant, and " $\cdot$ " denotes classical matrix multiplication.

The $\mathbf{A}_k$ matrix includes the calibration coefficients related with a second order part of the model mentioned in (24). Aforementioned matrix is defined as following:

$$\mathbf{A}_k = \begin{bmatrix} a_{k11} & a_{k12} & a_{k13} & \dots & a_{k16} \\ 0 & a_{k22} & a_{k23} & \dots & a_{k26} \\ 0 & 0 & a_{k33} & \dots & a_{k36} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{kij} \end{bmatrix} \tag{25}$$

where a_{kij} is a coefficient to calibrate a k-th parameter in relation to i-th and j-th measured parameter.

Moreover the B_k vector is defined as follows:

$$B_k = \begin{bmatrix} b_{k1} \\ b_{k2} \\ b_{k3} \\ \vdots \\ b_{ki} \end{bmatrix} \tag{26}$$

where b_{ki} is a coefficient to calibrate a k-th parameter in relation to i-th measured parameter.

The aforementioned PMC coefficients may be estimated through the least-square approach by means of the Equation (24).

In this research, four different angle representations are employed: Euler Angle Axis (EAA), quaternion, rotation matrix (R), and Zhou representation. Each of these representations incorporates a distinct set of parameters to describe angles. The subsequent subsection introduces the five PMC models employed to rectify position alone and both position and orientation across these four diverse representations.

2.3.1 | *Position Rectification*

The model represented by Equations (29) is used to rectify the position estimations and improve the resulting accuracy. In the case where only the position is rectified, the vector of the calibrated parameters takes the simplest form of:

$$X = [xyz] \tag{27}$$

where $\hat{x}, \hat{y}, \hat{z}$ are the measured coordinates. The indices i, j, and k in the rectification of the position values only change in the following range:

$$i = j = k = 1 \dots 3 \tag{28}$$

We can write out then three separate equations, based on the general PMC model given in (24) and coefficients matrices (25, 26), that describe way the respective coordinates are calibrated:

$$\begin{aligned} x' &= a_{111}x^2 + a_{122}y^2 + a_{133}z^2 + \\ &+ a_{112}xy + a_{113}xz + a_{123}yz + \\ &+ b_{11}x + b_{12}y + b_{13}z + c_1 \\ y' &= a_{211}x^2 + a_{222}y^2 + a_{233}z^2 + \\ &+ a_{212}xy + a_{213}xz + a_{223}yz + \\ &+ b_{21}x + b_{22}y + b_{23}z + c_2 \\ z' &= a_{311}x^2 + a_{322}y^2 + a_{333}z^2 + \\ &+ a_{312}xy + a_{313}xz + a_{323}yz + \\ &+ b_{31}x + b_{32}y + b_{33}z + c_3 \end{aligned} \tag{29}$$

where x, y, z are the measured coordinates, x', y', z', are the corrected values, and a_{kij}, b_{ki}, c_k are the calibration coefficients as described previously.

2.3.2 | *Rectification of Position and Orientation (EAA)*

In the EAA representation, three specific angles are utilized to define a particular orientation. These angles correspond to rotations around the x, y, and z axes required to attain a final orientation, assuming the initial orientation is aligned with the x-axis. We can name them in the following way: $\theta\hat{k}_x$, $\theta\hat{k}_y$ and $\theta\hat{k}_z$, as presented in the section Axis-angle representation.

Therefore, we can assume that measured parameters can be noted in the following way:

$$\hat{X} = [xyz\theta\hat{k}_x\theta\hat{k}_y\theta\hat{k}_z] \tag{30}$$

where x, y, z are the measured coordinates, $\hat{k}_x, \hat{k}_y, \hat{k}_z$ are the coordinates of the rotation axis, and θ is the rotation angle.

In the EAA representation, the indices i, j and k are defined in the range:

$$i = j = k = 1, \dots, 6 \tag{31}$$

2.3.3 | *Rectification of Position and Orientation (Quaternion)*

According to the detailed description of the quaternion representation presented in section *Quaternion representation*, there are 4 distinctive values to specify an orientation, namely: q_w, q_x, q_y, q_z. In that case, parameters that are going to be calibrated are:

$$\hat{X} = [xyzq_wq_xq_yq_z] \tag{32}$$

Therefore, for quaternion representation the indices i, j and k will change in the range:

$$i = j = k = 1, ..., 7 \tag{33}$$

2.3.4 | *Rectification of Position and Orientation (R)*

The R representation involves the usage of the 3×3 matrix containing values of the nine cosines of the specific angles. The R matrix can be noted in the short form as presented in Equation (10).

Therefore, nine parameters are needed to represent an orientation using the R representation. In fact, the vector of the calibrated parameters takes the following form:

$$\hat{X} = [xyzr_{11}r_{12}r_{13}r_{21}r_{22}r_{23}r_{31}r_{32}r_{33}] \tag{34}$$

In this representation, the indices i, j and k for the PMC method can be specified as follows:

$$i = j = k = 1, ..., 12 \tag{35}$$

2.3.5 | *Rectification of Position and Orientation (Zhou)*

Zhou representation requires 6 parameters to specify an orientation. Taking into account notations presented in section *Zhou representation*, the vector of calibrated parameters is stated as follows:

$$\hat{X} = [\hat{x}\hat{y}\hat{z}\hat{Z}_1\hat{Z}_2\hat{Z}_3\hat{Z}_4\hat{Z}_5\hat{Z}_6] \tag{36}$$

Utilizing the Zhou representation the indices i, j, and k, required to specify the matrices needed for the PMC method, change in the range of:

$$i = j = k = 1, ..., 9 \tag{37}$$

2.4 | Neural Network Calibration

Neural networks are a class of soft computing techniques in artificial intelligence that are based on parallel processing inspired by the human brain. In this study, a fully connected feedforward artificial neural network, specifically a Multilayer Perceptron (MLP), is employed. The strong computational capability of this network arises from a large number of processing units, commonly referred to as neurons, organized into distinct layers: an input layer, one or more hidden layers, and an output layer. The network architecture is defined by the number of layers and the number of neurons in each layer. Each neuron in a given layer is fully connected to every neuron in the subsequent layer via weighted connections, forming a fully connected feedforward network (Rumelhart et al. 1986).

One of the most essential characteristics of neural networks is their ability to learn and generalize acquired knowledge based on the available experimental data (Petrović et al. 2021). In this paper, we use a backpropagation algorithm for supervised learning of neural networks, based on gradient descent. In particular, the learning process involves computing the gradient of the error function with respect to the expected and actual outputs of the neural network, and modifying the connection weights between neurons accordingly. The error is therefore propagated through the network, and the training process is iterated until the error is minimized. For more details about the backpropagation algorithm, the reader is referred to the literature (Rumelhart et al. 1986) and (Hecht-Nielsen 1992).

To identify an appropriate neural network for the problem addressed in this paper, a systematic hyperparameter-tuning procedure is conducted. Widely used software tools such as MATLAB and Python (Pedregosa et al. 2011) are employed for model development, implementation, and evaluation, as they provide extensive libraries and frameworks that facilitate efficient design, training, and testing of neural networks. The tuning process considers key design choices that influence convergence behavior and generalization capability, including the learning algorithms, activation functions and network architectures. Hyperparameters are selected through a series of preliminary experiments to identify configurations that ensure stable convergence and satisfactory predictive performance. In particular, 12 different learning algorithms listed in Table 1 are evaluated, as they encompass a broad range of training paradigms with distinct optimization characteristics. In addition, eight activation functions, namely compet, hardlim, logsig, purelin, radbas, satlin, softmax, and tansig, are investigated in the hidden and output layers to examine their effectiveness in modeling nonlinear relationships and accommodating different output representations. Moreover, several network architectures with varying numbers of hidden layers and neurons are evaluated, and the final configuration, as shown in Table 1, is selected based on performance in preliminary experiments. These architectures include from one-layered to four-layered network architectures with the numbers of neurons in each layer varied from 5 to 20. The total number of architectures used in Experiments I, II, and III is 16. In total, 1536 different neural network models are tested in Experiment I (12 learning algorithms, eight activation functions, and 16 architectures), 384 in Experiment II (three learning algorithms, eight activation functions, and 16 architectures), and 96 in Experiment III (two learning algorithms, three activation functions, and 16 architectures). The remaining hyperparameters are set as follows: the learning rate is set to 0.01, the maximum number of epochs is set to 1000, and the RMSE error for training is set to 10^{-3}. The neural network models are implemented in a Matlab software environment and launched on a laptop computer with an Intel™ i7-5500U 2.4GHz processor with 8GB of RAM. To collect the best value, average value, and standard deviation, the experiments are run 30 times with randomly selected training and validation datasets (ratios 70% and 15%, respectively), while the test dataset (15%) is kept fixed and used only for final evaluation.

2.5 | Method for Pose Estimation Accuracy Quantification

In our work, we aim to quantify the accuracy of the VIVE tracker subjected to various methods of pose rectification. Typically, when two- or three-dimensional position data is considered, the pose estimation accuracy is computed as the root mean square of the error, i.e. In three-dimensional case:

TABLE 1 | The configuration space of neural network architectures, activation functions, and learning algorithms investigated.

Activation function	No.	Architecture	No.	Learning algorithm
compet	1	5	1	Levenberg-Marquardt backpropagation (LM)
hardlim	2	10	2	Bayesian regularization (BR)
logsig	3	15	3	Resilient backpropagation (RP)
purelin	4	20	4	Scaled conjugate gradient backpropagation (SCG)
radbas	5	5-5	5	BFGS quasi-Newton backpropagation (BFG)
satlin	6	10-10	6	Gradient descent backpropagation (GD)
softmax	7	15-15	7	Gradient descent with adaptive learning rule backpropagation (GDA)
tansig	8	20-20	8	Gradient descent with momentum backpropagation (GDM)
	9	5-5-5	9	Gradient descent with momentum and adaptive learning rule backpropagation (GDMA)
	10	10-10-10	10	Powell-Beale conjugate gradient backpropagation (PB)
	11	15-15-15	11	Fletcher-Powell conjugate gradient backpropagation (FP)
	12	20-20-20	12	Polak-Ribiére conjugate gradient backpropagation (PR)
	13	5-5-5-5		
	14	10-10-10-10		
	15	15-15-15-15		
	16	20-20-20-20		

Note: Specific neural network instances are created based on a combination of choices from each group (e.g., radbas 9/1 is a 5-5-5 network with radbas activation trained using the LM algorithm).

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i' - x_i\right)^2 + \left(y_i' - y_i\right)^2 + (z_i' - z_i)^2} \quad (38)$$

This approach is sometimes extended in the literature to also quantify the orientation errors. In such case, the orientation error is expressed in a representation of choice, and the RMSE is computed on a dataset of enlarged dimensionality (6 dimensions in the case of RPY representation). This approach does not account for the difference in units.

Alternatively, two separate error measures are computed: one for the position and one for the orientation error. This approach works well enough for small errors, where the non-linearity of the orientation representation can be neglected.

In our work, we measure the pose error separately for the position and for the orientation in the following way. We measure the error in the translation as a standard RMSE:

$$\mathrm{RMSE}_{\mathrm{pos}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|t_i - t_i'\|^2} \quad (39)$$

where t and t' are respectively the vectors representing the positions of the reference and the measured points.

Since we want our orientation error to be independent of the choice of representation, we compute the accuracy of orientation using the following procedure. an orientation difference between the rotation matrix $\mathbf{R}$ representing the reference orientation and the matrix R′ representing the measured orientation can be calculated as:

$$\Delta\mathbf{R} = \mathbf{R} \cdot \mathbf{R}'^{-1} \quad (40)$$

The magnitude of the difference in orientation $\Delta\mathbf{R}$ can be computed through the conversion of the resultant matrix $\Delta\mathbf{R}$ to the EAA representation as the θ angle:

$$\|\Delta\mathbf{R}\| = |\theta| = \left|\arccos\frac{\mathrm{tr}(\Delta\mathbf{R}) - 1}{2}\right| \quad (41)$$

We compute the RMSE of orientation based on the magnitudes of orientation errors thus:

$$\mathrm{RMSE}_{\mathrm{rot}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|\Delta\mathbf{R}_i\|^2} \quad (42)$$

Furthermore, for easy overall comparison of different calibration methods, we use the following quantity to represent the general pose estimation accuracy, including both the translation and the orientation errors:

$$\mathrm{RMSE}_{\mathrm{total}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(w_{\mathrm{pos}} \cdot \|t_i - t_i'\| + w_{\mathrm{rot}} \cdot \|\Delta\mathbf{R}_i\|\right)^2} \quad (43)$$

In the Equation (43), the $w_{\mathrm{pos}} = 1$ [1/m] and $w_{\mathrm{rot}} = 1$ [1] are the weights for the translation error and the orientation error respectively.

3 | Results

In this section, we discuss the results obtained from the three experiments described in the section Experimental setup. All

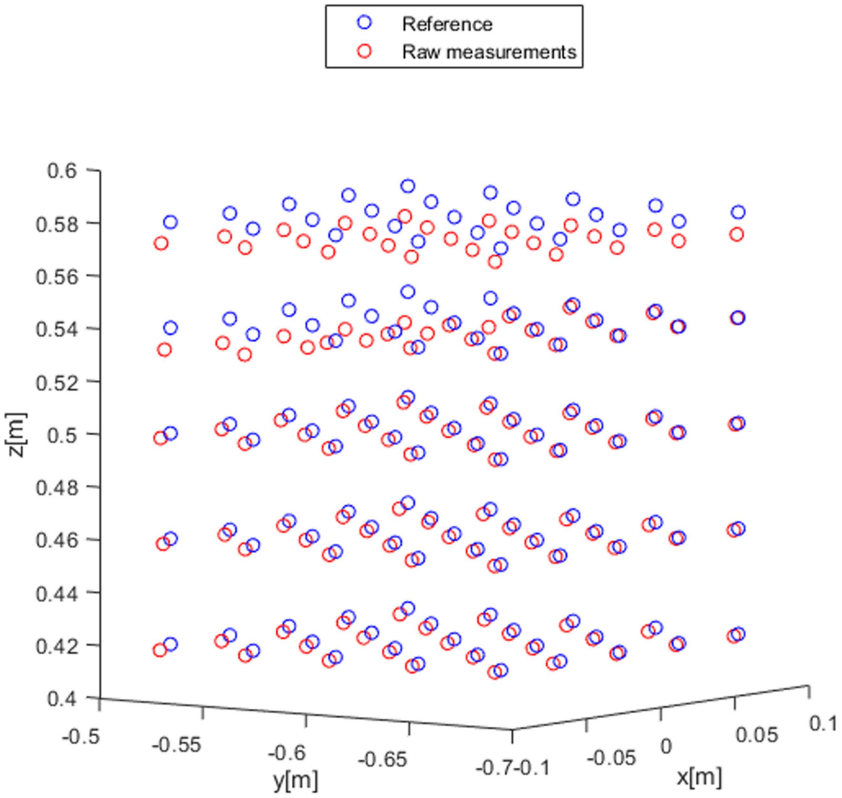

FIGURE 9 | Error distribution of the measured raw position in experiment I. [Color figure can be viewed at wileyonlinelibrary.com]

the results presented in this section are shown for the test dataset in each of the experiments.

3.1 | Experiment I

In this experiment, we have only considered the displacement in translation between the reference and the measured points arranged in a three-dimensional lattice. Since the orientation of the sensor was kept constantly upright, only the error in position was recorded (Figure 9). Consequently, the measured values of the RMSE are only provided for the accuracy in the position of the sensor.

In experiment I, we recorded the position accuracy of the raw dataset (10), the position accuracy after rectification using the PMC method, and the position accuracy of rectification using different NN architectures and training algorithms. The neural network inputs for Experiment I are 1331 measured/recorded sensor positions, and the outputs are coresponding reference points shown in Figure 4, where the sensor was placed. The list of the NN architecture types, activation function types, and learning algorithms choices referred to throughout the experiments is presented in Table 1. The concrete neural network instances are created by combining the options presented in the Table 1.

In Experiment I, we have tested the PMC correction, as well as all different architectures and learning algorithms for each of the activation functions. For each of these tests, 30 rounds of training were performed. The results of the fitting are presented in Table 2, and visualized in Figure 10. In this, and all subsequent tables, the improvement ratio is calculated as the ratio of the RMSE of the raw dataset over the minimum RMSE achieved with the compared method. We calculate such improvement rates for position, rotation, and total RMSE separately, but consider method the best when it achieves the highest improvement ratio in RMSE total.

Improvement in positioning accuracy has been achieved using PMC and almost all the tested NNs, with the exception of the NNs using the *compet* activation function. The best improvement of around 81 times was achieved using a NN with the *tansig* activation function, with the 15-15-15-15 architecture and using the LM training algorithm. The mentioned NN achieved the positional accuracy of $\mathrm{RMSE}_{\mathrm{total}} = 0.0005$, as compared to the positional accuracy $\mathrm{RMSE}_{\mathrm{total}} = 0.0440$ of the raw dataset. The accuracy obtained using PMC rectification was around 4 times worse, with $\mathrm{RMSE}_{\mathrm{total}} = 0.0021$.

3.2 | Experiment II

In Experiment II, we aimed to consider the influence of the choice of the orientation representation, as well as the rectification method on the improvement of the accuracy of the sensor positioning. We have measured a set of 1625 sensor poses in a cubical lattice, where we have varied the orientation of the sensor in the vertical axis. The neural network inputs for Experiment II are 1625 measured/recorded sensor poses, and the outputs are coresponding reference poses shown in Figure 5.

In this experiment, we have tested 8 activation functions, 16 NN architectures, and 3 learning algorithms (LM, BR, RP), as well as 4 orientation representations (EAA, quaternion, R, and

TABLE 2 | Results for pose accuracy improvement in Experiment I.

	Position RMSE [m]				Rotation RMSE [rad]				Total RMSE			
Method	**min**	**avg**	**max**	**imp**	**min**	**avg**	**max**	**imp**	**min**	**avg**	**max**	**imp**
raw data	0.0440	—	—	—	—	—	—	—	0.0440	—	—	—
PMC	0.0021	0.0023	0.0024	20.71	—	—	—	—	0.0021	0.0023	0.0024	20.71
compet 4/3	0.0692	0.0816	0.0964	0.64	—	—	—	—	0.0692	0.0816	0.0964	0.64
hardlim 4/3	0.0364	0.0410	0.0482	1.21	—	—	—	—	0.0364	0.0410	0.0482	1.21
logsig 16/1	0.0006	0.0010	0.0013	71.81	—	—	—	—	0.0006	0.0010	0.0013	71.81
purelin 12/3	0.0014	0.0016	0.0017	31.43	—	—	—	—	0.0014	0.0016	0.0017	31.43
radbas 15/1	0.0007	0.0011	0.0014	62.45	—	—	—	—	0.0007	0.0011	0.0014	62.45
satlin 4/2	0.0006	0.0009	0.0012	77.47	—	—	—	—	0.0006	0.0009	0.0012	77.47
softmax 16/1	0.0008	0.0186	0.0780	53.39	—	—	—	—	0.0008	0.0186	0.0780	53.39
tansig 15/1	0.0005	0.0009	0.0012	81.21	—	—	—	—	**0.0005**	0.0009	0.0012	81.21

Note: The best rectification method for each representation is marked in bold based on the minimum value of the total RMSE found for all runs (the column highlighted in gray). The positional accuracy improvement rates are highlighted in green (for positive improvement) or red (for negative improvement). Method is indicated either as PMC or activation function-architecture number/algorithm number. Only translational error is considered in this experiment; therefore, the position and total RMSE values are the same.

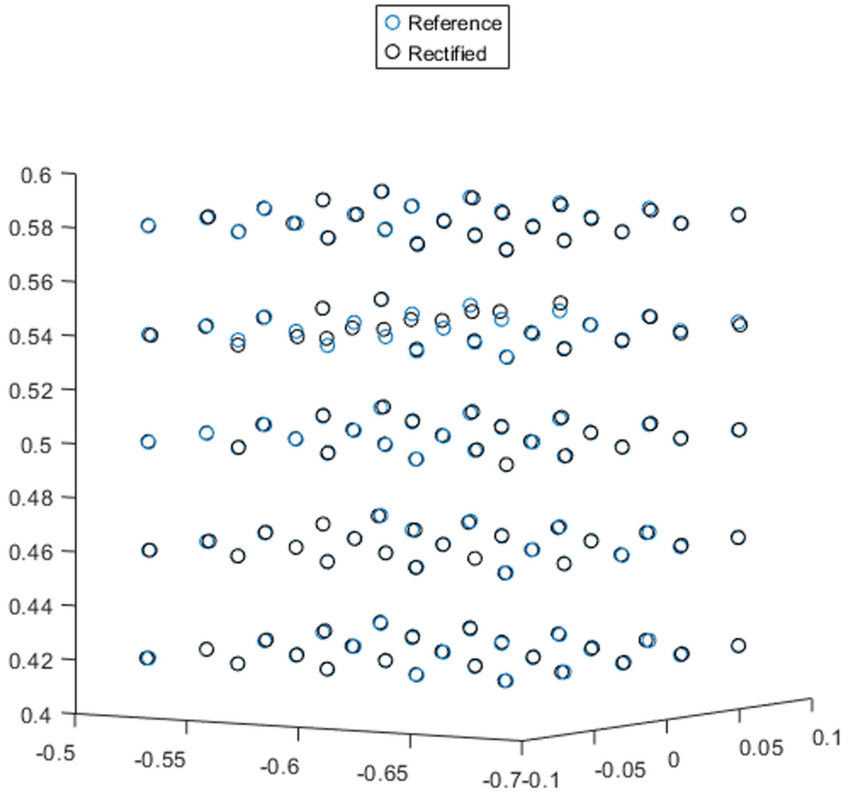

FIGURE 10 | Error distribution of the position rectified with a tansig 15/1 NN in experiment I. [Color figure can be viewed at wileyonlinelibrary.com]

Zhou). We have also included the results for PMC rectification for the comparison. The results for the Experiment II are presented in the Table 3.

The pose estimation accuracy of the raw data for this experiment was calculated to be $\text{RMSE}_{\text{total}} = 0.0167$. We have achieved no improvement in the pose accuracy using PMC and NN rectification with the use of the EAA representation. In fact, with the EAA representation, the pose accuracy was drastically degraded, as compared even to the raw dataset.

Similar outcomes were observed when employing the quaternion representation, with both the PMC and a majority of NN structures showing poor performance. Only the NN using the *radbas* and *tansig* have achieved modest pose estimation accuracy improvement.

TABLE 3 | Results for pose accuracy improvement in Experiment II.

	Position RMSE [m]				Rotation RMSE [rad]				Total RMSE			
Method	**min**	**avg**	**max**	**imp**	**min**	**avg**	**max**	**imp**	**min**	**avg**	**max**	**imp**
raw data	0.0058	—	—	—	0.0115	—	—	—	0.0167	—	—	—
EAA representation												
PMC	0.0027	0.0029	0.0030	2.16	0.4324	0.5156	0.6296	0.03	0.4339	0.5169	0.6308	0.04
compet 1/1	0.1734	0.2009	0.2285	0.03	1.0909	1.4767	1.7137	0.01	1.2626	1.6409	1.8891	0.01
hardlim 4/2	0.1063	0.1198	0.2061	0.05	0.6191	0.8311	1.0836	0.02	0.7177	0.9253	1.1766	0.02
logsig 10/1	0.1026	0.1527	0.2071	0.06	0.2450	0.4784	0.6864	0.05	0.3244	0.5728	0.7887	0.05
purelin 8/1	0.0028	0.0041	0.0131	2.07	0.2760	0.3666	0.4522	0.04	0.2778	0.3687	0.4538	0.06
radbas 15/1	0.0764	0.2076	0.2420	0.08	0.2734	0.5692	0.7563	0.04	0.3092	0.7154	0.9217	0.05
satlin 16/2	0.0448	0.0865	0.1394	0.13	0.2642	0.5130	0.6761	0.04	0.2868	0.5669	0.7286	0.06
softmax 4/2	0.1175	0.1528	0.2097	0.05	0.1511	0.4618	0.5927	0.08	0.2690	0.5491	0.6659	0.06
tansig 7/2	0.0215	0.1400	0.2491	0.27	0.1942	0.5009	0.6629	0.06	**0.2097**	0.5897	0.7349	0.08
Quaternion representation												
PMC	0.0026	0.0028	0.0030	2.22	0.0786	0.1053	0.1956	0.15	0.0802	0.1069	0.1968	0.21
compet 3/3	0.1651	0.2219	0.2553	0.04	0.8193	1.4546	1.8299	0.01	0.9635	1.6355	2.0399	0.02
hardlim 4/1	0.1002	0.1946	0.2460	0.06	0.5017	1.2180	1.7809	0.02	0.6018	1.3733	1.9719	0.03
logsig 11/2	0.0074	0.0804	0.2529	0.78	0.0143	0.1758	0.5395	0.80	0.0252	0.2324	0.6684	0.66
purelin 15/2	0.0027	0.0136	0.2040	2.12	0.0615	0.1110	0.4341	0.19	0.0636	0.1211	0.4755	0.26
radbas 15/1	0.0022	0.0988	0.2484	2.63	0.0047	0.2752	1.5110	2.44	**0.0065**	0.3462	1.7020	2.57
satlin 11/2	0.0069	0.0937	0.2543	0.84	0.0459	0.2335	1.8527	0.25	0.0701	0.3005	2.0539	0.24
softmax 4/2	0.0106	0.1685	0.5198	0.55	0.0175	0.2789	1.8593	0.66	0.0293	0.4087	2.0698	0.57
tansig 7/2	0.0027	0.0765	0.2528	2.11	0.0097	0.1796	0.4940	1.19	0.0109	0.2336	0.6293	1.53
R representation												
PMC	0.0023	0.0026	0.0033	2.56	0.0034	0.0041	0.0050	3.36	0.0054	0.0062	0.0077	3.10
compet 4/2	0.1846	0.2053	0.2227	0.03	0.4057	0.7267	1.2721	0.03	0.5375	0.8405	1.3759	0.03
hardlim 4/1	0.1238	0.1392	0.1574	0.05	0.2985	0.3745	0.5511	0.04	0.3886	0.4630	0.6263	0.04
logsig 11/2	0.0017	0.0020	0.0029	3.34	0.0092	0.0195	0.0274	1.26	0.0100	0.0206	0.0292	1.67
purelin 4/2	0.0028	0.0472	0.1136	2.07	0.0001	0.0065	0.0448	115.35	0.0100	0.0520	0.1136	1.67
radbas 15/1	0.0017	0.0020	0.0026	3.48	0.0161	0.0239	0.0287	0.72	0.0171	0.0250	0.0298	0.98
satlin 16/2	0.0018	0.0641	0.2526	3.22	0.0172	0.0258	0.0559	0.67	0.0180	0.0855	0.2710	0.93
softmax 4/2	0.0022	0.0671	0.2016	2.68	0.0239	0.0429	0.0801	0.48	0.0251	0.1014	0.2569	0.67
tansig 16/2	0.0016	0.0020	0.0023	3.62	0.0089	0.0143	0.0212	1.30	**0.0098**	0.0153	0.0227	1.71
Zhou representation												
PMC	0.0024	0.0026	0.0028	2.43	0.0036	0.0041	0.0048	3.22	0.0056	0.0061	0.0069	3.01
compet 3/3	0.1880	0.1996	0.2217	0.03	0.4939	0.7634	1.0333	0.02	0.6727	0.9167	1.1748	0.02
hardlim 4/1	0.1294	0.1313	0.1348	0.04	0.2639	0.3385	0.4766	0.04	0.3745	0.442	0.5774	0.04
logsig 11/2	0.0016	0.0018	0.0022	3.62	0.0014	0.0023	0.0034	8.24	0.0028	0.0038	0.005	5.98
purelin 15/2	0.0029	0.0031	0.0033	2.00	0.0036	0.0046	0.0055	3.20	0.0058	0.0071	0.008	2.89
radbas 15/1	0.0017	0.0020	0.0028	3.40	0.0015	0.0029	0.004	7.69	0.0029	0.0045	0.0064	5.77
satlin 11/2	0.0017	0.0020	0.0026	3.40	0.0019	0.0027	0.0034	6.07	0.0034	0.0044	0.0055	4.92
softmax 4/2	0.0022	0.0026	0.0043	2.63	0.0027	0.0040	0.0095	4.27	0.0044	0.0061	0.0136	3.80
tansig 7/2	0.0016	0.0018	0.0021	3.62	0.0014	0.0023	0.0036	8.24	**0.0028**	0.0037	0.0052	5.98

Note: The best rectification method for each representation is marked in bold based on the minimum value of the total RMSE found for all runs (the column highlighted in gray). The pose accuracy improvement rates are highlighted in green (for positive improvement) or red (for negative improvement). Method is indicated either as PMC or activation function-architecture number/algorithm number.

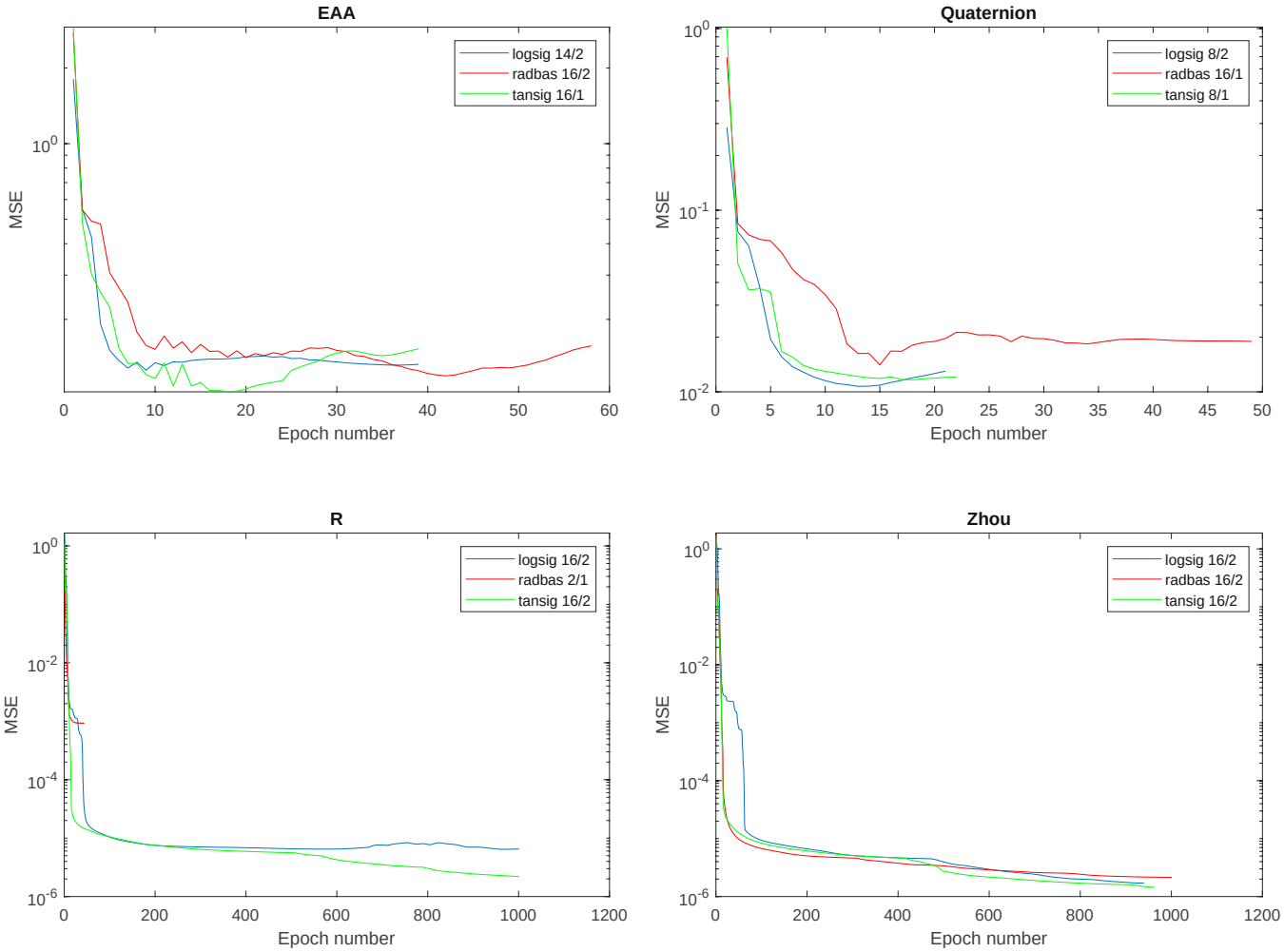

FIGURE 11 | Learning curves for the best NNs in individual representations in the experiment III. [Color figure can be viewed at wileyonlinelibrary.com]

The PMC was able to achieve pose estimation accuracy improvement using only the R and Zhou representations. a good apose estimation accuracy improvement rate of around $5-6$ was only achieved by some NNs with the Zhou representation, where the best result of $\text{RMSE}_{\text{total}} = 0.0028$ and the accuracy improvement of ~6 was achieved using the *tansig* activation function, 15-15 architecture and the BR training algorithm.

It can be noted that the different calibration methods are able to achieve consistent improvement rate when only a position is considered, while they struggle in the rotational aspect of the rectification problem.

3.3 | Experiment III

In Experiment III we have used a dataset that contained the poses measured in a 20 x 20 x 20 cm cube, where the sensor orientation varied in all three axes with the range of $\pm 30^{o}$. This test was meant to be the ultimate test for the performance of the various rectification methods developed throughout Experiment I and II. The neural network inputs for Experiment III are 3375 measured/recorded sensor poses, and the outputs are coresponding reference poses shown in Figure 6.

Because of the large dataset ($3 \times 3 \times 3 \times 5 \times 5 \times 5 = 3375$ data points) and the associated computational cost, in Experiment III we have considered only two learning algorithms (LM and BR), 16 architectures, and three best activation function combinations (logsig, radbas, tansig) for each of the four representations, according to the results from Experiment II. Nevertheless, for each of these architectures, the computation time ranged from 50 to 100 h across the 30 training sessions.

The resulting learning curves for the best solutions of the NNs in each of the representations are presented in the Figure 11, which represent the MSE of the NNs for the test dataset. As shown part of the learning processes finished earlier, because they met the stop criterion.

In Experiment III the raw data pose accuracy was found as $\text{RMSE}_{\text{total}} = 0.0718$. No improvement was recorded using the PMC or the NNs with the EAA and quaternion representations.

Moderate improvement in pose estimation accuracy was achieved using PMC for both the R ($\text{RMSE}_{\text{total}} = 0.0138$ and the

TABLE 4 | Results for pose accuracy improvement in Experiment III.

	Position RMSE [m]				Rotation RMSE [rad]				Total RMSE			
Method	**min**	**avg**	**max**	**imp**	**min**	**avg**	**max**	**imp**	**min**	**avg**	**max**	**imp**
raw data	0.0222	—	—	—	0.0508	—	—	—	0.0718	—	—	—
EAA representation												
PMC	0.0079	0.0084	0.0087	2.80	0.3515	0.3884	0.4427	0.14	0.3574	0.3943	0.4484	0.20
logsig 14/2	0.1234	0.1411	0.1450	0.18	0.3530	0.5400	0.6602	0.14	0.4458	0.6284	0.7465	0.16
radbas 16/2	0.0486	0.1277	0.1404	0.46	0.3124	0.6270	1.0604	0.16	**0.3352**	0.7183	1.1708	0.21
tansig 16/1	0.0759	0.1007	0.1203	0.29	0.3281	0.5575	0.7377	0.15	0.3772	0.6259	0.8142	0.19
Quaternion representation												
PMC	0.0055	0.0057	0.0059	4.06	0.1049	0.1198	0.1343	0.48	**0.1093**	0.1241	0.1385	0.66
logsig 8/2	0.0167	0.0234	0.0314	1.33	0.1198	0.2767	0.3603	0.42	0.1364	0.2888	0.3692	0.53
radbas 16/1	0.0200	0.0712	0.1058	1.11	0.1368	0.3234	0.4783	0.37	0.1488	0.3737	0.5280	0.48
tansig 8/2	0.0166	0.0237	0.0303	1.34	0.1195	0.2718	0.3658	0.43	0.1346	0.2844	0.3792	0.53
R representation												
PMC	0.0036	0.0037	0.0039	6.24	0.0105	0.0110	0.0118	4.85	**0.0138**	0.0143	0.0153	5.21
logsig 16/2	0.0020	0.0023	0.0032	11.13	0.0197	0.0252	0.0294	2.58	0.0208	0.0266	0.0312	3.45
radbas 2/1	0.0090	0.0168	0.0393	2.47	0.0174	0.1136	0.1313	2.91	0.0223	0.1252	0.1452	3.23
tansig 16/2	0.0020	0.0023	0.0030	11.33	0.0217	0.0254	0.0309	2.34	0.0230	0.0268	0.0322	3.12
Zhou representation												
PMC	0.0048	0.0049	0.0051	4.67	0.0128	0.0132	0.0138	3.95	0.0172	0.0177	0.0183	4.17
logsig 16/2	0.0018	0.0074	0.0816	12.33	0.0026	0.005	0.0117	19.54	**0.0042**	0.0118	0.0900	17.10
radbas 16/2	0.0019	0.0022	0.0027	11.68	0.0027	0.0046	0.0090	18.81	0.0044	0.0064	0.0108	16.32
tansig 16/2	0.0019	0.0021	0.003	11.68	0.0025	0.0045	0.0147	20.32	**0.0042**	0.0063	0.0171	17.10

Note: The best rectification method for each representation is marked in bold based on the minimum value of the total RMSE found for all runs (the column highlighted in gray). The estimation accuracy improvement rates are highlighted in green (for positive improvement) or red (for negative improvement). Method is indicated either as PMC or activation function-architecture number/algorithm number.

Zhou representations ($\text{RMSE}_{\text{total}} = 0.0172$) corresponding to the improvement rates of 5.21 and 4.17 respectively.

Best pose accuracy improvement rate of 17.10 was recorded using *logsig* and *tansig* activation functions paired with the Zhou representation, where the $\text{RMSE}_{\text{total}} = 0.0042$ was achieved. Overall, the improvement using the NN was much greater with the Zhou representation than with the R representation.

Nearly all tried rectification methods were able to achieve pose estimation accuracy improvement when only the position of the sensor was considered, with the exception of the EAA representation of the rotation (rectification in position is still calculated using all inputs—orientation included).

The results for Experiment III are presented in Table 4.

In Experiment III, as explained above, we compared rectification outcomes across several selected methods and network architectures for each representation (see the column *method* in Table 4).

We have conducted further analysis to verify that the outcomes of the presented method are statistically significant. Let $M = 16$ denote the number of evaluated methods and $N = 30$ the number of tests per method. Let $r_{i,j}$ be the rank of method $j \in \{1, ..., M\}$ on test run $i \in \{1, ..., N\}$ based on its total RMSE.

The null hypothesis is defined as

$$H_0 : \quad \mathbb{E}[r_{i,1}] = \mathbb{E}[r_{i,2}] = \cdots = \mathbb{E}[r_{i,M}], \tag{44}$$

which states that all methods have equal expected performance.

We have rejected the null hypothesis H_0 based on a Friedman test with $p < 0.001$.

Next, we have performed a Dunn–Šidák test to establish how the analysed methods compare against each other pair-wise. In Table A1 in the Appendix we present the relative mean rank difference between pairs of methods, along with the confidence interval bounds and the p-values.

Finally, based on the pair-wise comparisons and their associated rank differences and p-values, we sort the analysed methods into a tier list, such that methods in tier i all statistically significantly outperform methods in subsequent tier $i + 1$, and there are no statistically significant difference among the methods in a single tier. The tier list is presented in Table 5 below.

TABLE 5 | Tier list of methods used in Experiment III according to Dunn–Šidák test.

Tier	Methods					
1	R-PMC	Zhou-PMC	Zhou-logsig-16/2	Zhou-radbas-16/2	Zhou-tansig-16/2	
2	Quaternion-PMC	R-logsig-16/2	R-radbas-2/1	R-tansig-16/2		
3	EAA-PMC	EAA-logsig-14/2	EAA-tansig-16/1	Quaternion-logsig-8/2	Quaternion-radbas-16/1	Quaternion-tansig-8/2
4	EAA-radbas-16/2					

4 | Discussion

The evaluation phase involved the use of the UR5e robot as a precise reference system. Three distinct experiments, referred to as Experiment I, Experiment II, and Experiment III, were designed to encompass diverse measurement scenarios. Each experiment generated a dataset, with the largest dataset obtained from Experiment III. Employing the PMC algorithm and neural networks, we aimed to improve the static position and orientation estimations provided by the VIVE tracker.

In Experiment I, we exclusively assessed and calibrated the position measurements. Experiment II involved an examination of both position and orientation changes along specific axes. Based on the evaluation of numerous neural networks, we selected a subset that demonstrated promising results in Experiment II as rectification tools for Experiment III. Experiment III encompassed changes in both position and orientation angles, yielding the most extensive dataset of over 3300 probes, despite a reduction in position-set variability. Notably, the training time for certain neural networks in this stage reached approximately 130–150 h.

An intriguing insight emerges from the obtained results, particularly with respect to the various orientation representation systems used in Experiment III. The Zhou representation stands out as the most effective among the options considered. Conversely, both the EAA and quaternion representations, employed during the calibration stage, exhibited inferior accuracy in representing orientation compared to the raw data itself. Although the use of the R representation led to a notable improvement, it still fell short when compared to the performance of the Zhou representation.

These observations may be attributed to the inherent nonlinearities present in EAA and quaternion representations, making them less amenable to modeling by neural networks. It is noteworthy that R and Zhou representations are based on the same principle, with the former utilizing nine parameters as opposed to the latter's six. Therefore, providing sufficient computational time and power for training neural networks could narrow or even eliminate the performance gap between R and Zhou representations. The advantage of the Zhou representation over the R representation results from the lower number of utilized parameters, leading to a less redundant input vector for the neural network.

The representations can be sorted according to their performance with neural network calibration from the worst to the best as follows: EAA, quaternion, R, Zhou. The EAA performs worst due to discontinuity and singularities in its parameter space and the double-cover property. The double-cover property also limits the performance of the quaternion representation. The two best representations do not have these limitations, with the superiority of the Zhou representation over the R representation explainable by its less redundant structure.

The PMC algorithm demonstrated commendable performance in comparison to neural networks. Its improvement was four times lower than the best-performing neural network in Experiment I, twice as inferior in Experiment II, and three times lower in Experiment III. Despite the lower pose estimation accuracy, PMC still yields a substantial improvement over raw data, achieving enhancements of 20, 3, and 5 times in Experiments I, II, and III,

respectively. Notably, the computational time required to derive the polynomial model parameters was significantly shorter, taking only a few minutes, in contrast to the neural network training, which required hundreds of times more computational time. Hence, PMC emerges as an intriguing solution, particularly when relatively fast calibration is essential, especially for applications focused solely on positioning.

The high discrepancy between the minimum and maximum values of RMSE for some orientation representations and activation functions results from underfitting. This occurs when linear activation functions, such as purelin 4/2 presented in Table 3 for the R representation, are too simplistic to capture nonlinear relationships and data complexity. To address this issue, various neural network architectures incorporating additional layers, increased numbers of neurons, different learning algorithms, and nonlinear activation functions were investigated. Overall, nonlinear activation functions, particularly those related to trigonometric behavior (logsig, radbas, tansig), appear best suited to reflect the complexities inherent in orientation representation.

We posed a null hypothesis that the proposed methods exhibit the same level of performance and rejected it using the Friedman test. Subsequently, the Dunn–Šidák test was applied to establish statistically significant performance differences. The results indicate that the Zhou representation is best suited for pose rectification and is tied with the polynomial correction method based on the rotation matrix representation.

Multiple neural network architectures were evaluated across the experiments, including shallow and deep feedforward networks with varying numbers of hidden layers, neurons, and activation functions. The tested configurations incorporated both linear and nonlinear activation functions such as `purelin`, `tansig`, `logsig`, and `radbas`, as well as different learning algorithms. The results indicate that nonlinear activation functions and deeper architectures generally provided better performance, particularly for orientation calibration in Experiment III, where the data exhibited stronger nonlinear characteristics. Simpler architectures showed reduced computational requirements but were more prone to underfitting, especially when applied to complex orientation representations.

5 | Conclusion

In this work we have considered the problem of the choice of calibration method and orientation representation best suited for the improvement of the tracking accuracy of the HTC VIVE tracker. We have collected an extensive dataset of pose samples measured in a three static scenarios while using an UR5 collaborative robot arm as a reference setup. We have compared the accuracy of the multiple Neural Network based calibration methods using the Polynomial Model Correction as a baseline while using four different representations of the measured pose orientation. The obtained results show that the choice of the orientation representation plays an important role in the accuracy improvement and the proper choice of representation and the rectification method offers a potential for a remarkable and statistically significant 17-times improvement in pose estimation accuracy in combined position and orientation RMSE. In future work, this analysis should be extended to include the dynamic performance of the system, as well as to assess a variance between individual instances of the device. Moreover, the considerable computational requirements of neural network training constitute a significant limitation and should be addressed in future work.

The PMC algorithm offers a valuable alternative when fast calibration is required, providing significant accuracy improvements with minimal computational cost. While neural networks achieved the highest accuracy, their training required substantial computational time, reaching up to 150 h in Experiment III. Therefore, the choice between neural networks and polynomial correction depends on the required trade-off between accuracy and computational efficiency. Furthermore, the choice of orientation representation plays a crucial role, with the Zhou representation consistently achieving the best performance due to its compact and non-redundant structure. The obtained results confirm that both the representation and neural network architecture significantly influence calibration effectiveness.

It is also worth noting that the HTC VIVE tracker operates as a stand-alone system, preventing access to its internal algorithms. If modification of these internal algorithms were possible, further improvements in pose estimation accuracy could potentially be achieved while reducing computational demands. The results were obtained using a single Universal Robots UR5 and one HTC VIVE Tracker setup, which limits statistical generalization. The achieved accuracy may depend on unit-to-unit variability and manufacturer repeatability; therefore, further validation on additional systems is encouraged, and the proposed method can be readily verified by other researchers.

Acknowledgments

The research leading to these results has received funding from the commissioned task entitled "VIA CARPATIA Universities of Technology Network named after the President of the Republic of Poland Lech Kaczyński" contract no. MEiN/2022/DPI/2577 action entitled "In the neighborhood—inter-university research internships and study visits. The research has also been supported by the Science Fund of the Republic of Serbia, Grant No. 17801, Cybersecurity of Motion Control Systems in Industry 4.0—MCSecurity, as well as by the Ministry of Science, Technological Development, and Innovations of the Serbian Government under Contract No. 451-03-34/2026-03/200105.

Data Availability Statement

The data that support the findings of this study are openly available in jfr_vive_htc at https://gitlab.com/kair_robotics/jfr_vive_htc.

Endnotes

[1] https://gitlab.com/kair_robotics/jfr_vive_htc.

[2] https://gitlab.com/kair_robotics/kair_universalrobot.

[3] https://gitlab.com/kair_robotics/kair_trajectory.

[4] https://gitlab.com/kair_robotics/kair_vive.

References

Astad, M. A., M. H. Arbo, E. I. Grotli, and J. T. Gravdahl. 2019. "Vive for Robotics: Rapid Robot Cell Calibration." In *7th International Conference on Control, Mechatronics and Automation (ICCMA)*, 151–156.

Bauer, P., W. Lienhart, and S. Jost. 2021. “Accuracy Investigation of the Pose Determination of a VR System.” *Sensors* 21, no. 5: 1622

Bi, Z. M., M. Luo, Z. Miao, B. Zhang, W. J. Zhang, and L. Wang. 2021. “Safety Assurance Mechanisms of Collaborative Robotic Systems in Manufacturing.” *Robotics and Computer-Integrated Manufacturing* 67: 102022.

Bilal, D., M. Unel, T. Tunc, and B. Gonul. 2022. “Development of a Vision Based Pose Estimation System for Robotic Machining and Improving Its Accuracy Using LSTM Neural Networks and Sparse Regression.” *Robotics and Computer-Integrated Manufacturing* 74: 102262.

Billard, A., and D. Grollman. 2013. “Robot Learning by Demonstration.” *Scholarpedia* 8: 3824.

Borges, M., A. Symington, B. Coltin, T. Smith, and R. Ventura. 2018. “Htc vive: Analysis and Accuracy Improvement.” In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, 2610–2615. IEEE.

Ciężkowski, M., S. Romaniuk, and A. Wolniakowski. 2020. “Apparent Beacon Position Estimation for Accuracy Improvement in Lateration Positioning System.” *Measurement* 153: 107400.

Craig, J. J. 2004. “Introduction to Robotics: Mechanics and Control 3Rd.” *Prentice Hall*.

Ellekilde, L. -P., and J. Jorgensen. 2010. “Robwork: A Flexible Toolbox for Robotics Research and Education.” In *SR/ROBOTIK 2010, Proceedings for the joint conference of ISR 2010 (41st Internationel Symposium on Robotics) und ROBOTIK 2010 (6th German Conference on Robotics)*, 1–7.

Englert, P., and M. Toussaint. 2018. “Learning Manipulation Skills From a Single Demonstration.” *International Journal of Robotics Research* 37, no. 1: 137–154.

Grollman, D. H., and A. G. Billard. 2012. “Robot Learning From Failed Demonstrations.” *International Journal of Social Robotics* 4: 331–342.

Hecht-Nielsen, R. 1992. “Theory of the Backpropagation Neural Network.” In *Neural Networks for Perception*, 65–93. Elsevier.

Hollerbach, J., and C. Wampler. 1996. “The Calibration Index and Taxonomy for Robot Kinematic Calibration Methods.” *International Journal of Robotics Research* 15: 573–591.

Holzwarth, V., J. Gisler, C. Hirt, and A. Kunz. 2021. “Comparing the Accuracy and Precision of Steamvr Tracking 2.0 and Oculus Quest 2 in a Room Scale Setup.” In *Proceedings of 5th International Conference on Virtual and Augmented Reality Simulations*.

Kulozik, J., and N. Jarrassé. 2025. “Evaluating the Precision of the HTC VIVE Ultimate Tracker With Robotic and Human Movements Under Varied Environmental Conditions.”

Lan, G., Y. Wu, F. Hu, and Q. Hao. 2022. “Vision-Based Human Pose Estimation Via Deep Learning: A Survey.” In *IEEE Transactions on Human-Machine Systems*, 1–16. IEEE.

Middleton, R. H., G. C. Goodwin, and R. W. Longman. 1989. “A Method for Improving the Dynamic Accuracy of a Robot Performing a Repetitive Task.” *International Journal of Robotics Research* 8, no. 5: 67–74.

Murray, R. M., Z. Li, and S. S. Sastry. 2017. *A Mathematical Introduction to Robotic Manipulation*. CRC Press.

Niehorster, D., L. Li, and M. Lappe. 2017. “The Accuracy and Precision of Position and Orientation Tracking in the HTC Vive Virtual Reality System for Scientific Research.” *i-Perception* 8: 204166951770820.

Pedregosa, F., G. Varoquaux, A. Gramfort, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825–2830.

Peer, A., P. Ullich, and K. Ponto. 2018. “Vive Tracking Alignment and Correction Made Easy.” In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 653–654. IEEE.

Perez-Villeda, H., J. Piater, and M. Saveriano. 2023. “Learning and Extrapolation of Robotic Skills Using Task-Parameterized Equation Learner Networks.” *Robotics and Autonomous Systems* 160: 104309.

Petrović, M., M. Ciężkowski, S. Romaniuk, A. Wolniakowski, and Z. Miljković. 2021. “A Novel Hybrid nn-Abpe-Based Calibration Method for Improving Accuracy of Lateration Positioning System.” *Sensors* 21, no. 24: 8204.

Pignat, E., J. Silvério, and S. Calinon. 2022. “Learning From Demonstration Using Products of Experts: Applications to Manipulation and Task Prioritization.” *The International Journal of Robotics Research* 41, no. 2: 163–188.

Ravichandar, H., A. S. Polydoros, S. Chernova, and A. Billard. 2020. “Recent Advances in Robot Learning From Demonstration.” *Annual Reviews* 3: 297–330.

Robotics, N. n.d. “Mimic Kit.” https://www.nordbo-robotics.com/mimic-1. [Online; accessed: Dec. 30, 2025].

Robots, U. n.d. “Ur5 Technical Specifications.” https://www.universal-robots.com/media/50588/ur5_en.pdf. [Online; accessed: Dec. 30, 2025].

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. “Learning Representations by Back-Propagating Errors.” *Nature* 323, no. 6088: 533–536.

Sansone, L., R. Stanzani, M. Job, S. Battista, A. Signori, and M. Testa. 2021. “Robustness and Static-Positional Accuracy of the Steamvr 1.0 Virtual Reality Tracking System.” *Virtual Reality* 26: 903–924

Soares, I., R. B. Sousa, M. Petry, and A. P. Moreira. 2021. “Accuracy and Repeatability Tests on Hololens 2 and Htc Vive.” *Multimodal Technologies and Interaction* 5: 47.

Spong, M. W., S. Hutchinson, and M. Vidyasagar. 2006. *Robot Modeling and Control* (2nd edition), Vol. 26. John. Wiley & Sons, Inc.

Valve. n.d. “Steamvr.” https://store.steampowered.com/app/250820/SteamVR/. [Online; accessed: Dec. 30, 2025].

Vive. n.d. “Vive Tracker.” https://www.Vive.com/us/accessory/tracker3/. [Online; accessed: Dec. 30, 2025].

Weber, M., R. Hartl, M. F. Zäh, and J. Lee. 2023. “Dynamic Pose Tracking Accuracy Improvement Via Fusing HTC Vive Trackers and Inertia Measurement Units.” *International Journal of Precision Engineering and Manufacturing* 24: 1671–1624.

Wu, R., J. Pandurangaiah, G. M. Blankenship, et al. 2020. “Evaluation of Virtual Reality Tracking Performance for Indoor Navigation.” In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 1311–1316. ION.

Xi, B., S. Wang, X. Ye, Y. Cai, T. Lu, and R. Wang. 2019. “A Robotic Shared Control Teleoperation Method Based on Learning From Demonstrations.” *International Journal of Advanced Robotic Systems* 16: 1–13.

Xiang, Y., T. Schmidt, V. Narayanan, and D. Fox. 2018. “Posecnn: A Convolutional Neural Network for 6d Object Pose Estimation in Cluttered Scenes.” *Robotics: Science and Systems* 14.

Xie, Z. W., Q. Zhang, Z. N. Jiang, and H. Liu. 2020. “Robot Learning From Demonstration for Path Planning: A Review.” *Science China Technological Sciences* 63: 1325–1334.

Zana, R. R., and A. Zelei. 2021. “Feedback Motion Control of a Spatial Double Pendulum Manipulator Relying on Swept Laser Based Pose Estimation.” *International Journal of Optomechatronics* 15, no. 1: 32–60.

Zhang, T., R. Tian, H. Yang, et al. 2022. “From Simulation to Reality: A Learning Framework for Fish-Like Robots to Perform Control Tasks.” *IEEE Transactions on Robotics* PP: 1–18.

Zhou, Y., C. Barnes, L. Jingwan, Y. Jimei, and L. Hao. 2019. “On the Continuity of Rotation Representations in Neural Networks.” In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5738–5746. IEEE.

Appendix A

Appendix

See Table A1

TABLE A1 | Multiple comparison Dunn–Šidák test results.

Method 1	Method 2	CI lower	Mean rank difference	CI upper	*p*-value
eaa-pmc	eaa-logsig-14-2	−6.397	−2.067	2.264	1.000
eaa-pmc	eaa-radbas-16-2	−7.097	−2.767	1.564	0.948
eaa-pmc	eaa-tansig-16-1	−6.464	−2.133	2.197	1.000
eaa-pmc	quat-pmc	0.070	4.400	8.730	0.040
eaa-pmc	quat-logsig-8-2	−2.497	1.833	6.164	1.000
eaa-pmc	quat-radbas-16-1	−3.864	0.467	4.797	1.000
eaa-pmc	quat-tansig-8-2	−2.464	1.867	6.197	1.000
eaa-pmc	R-pmc	4.536	8.867	13.197	<0.001
eaa-pmc	R-logsig-16-2	2.003	6.333	10.664	<0.001
eaa-pmc	R-radbas-2-1	−0.464	3.867	8.197	0.181
eaa-pmc	R-tansig-16-2	2.003	6.333	10.664	<0.001
eaa-pmc	zhou-pmc	3.536	7.867	12.197	<0.001
eaa-pmc	zhou-logsig-16-2	5.470	9.800	14.130	<0.001
eaa-pmc	zhou-radbas-16-2	5.936	10.267	14.597	<0.001
eaa-pmc	zhou-tansig-16-2	6.336	10.667	14.997	<0.001
eaa-logsig-14-2	eaa-radbas-16-2	−5.030	−0.700	3.630	1.000
eaa-logsig-14-2	eaa-tansig-16-1	−4.397	−0.067	4.264	1.000
eaa-logsig-14-2	quat-pmc	2.136	6.467	10.797	<0.001
eaa-logsig-14-2	quat-logsig-8-2	−0.430	3.900	8.230	0.166
eaa-logsig-14-2	quat-radbas-16-1	−1.797	2.533	6.864	0.992
eaa-logsig-14-2	quat-tansig-8-2	−0.397	3.933	8.264	0.152
eaa-logsig-14-2	R-pmc	6.603	10.933	15.264	<0.001
eaa-logsig-14-2	R-logsig-16-2	4.070	8.400	12.730	<0.001
eaa-logsig-14-2	R-radbas-2-1	1.603	5.933	10.264	<0.001
eaa-logsig-14-2	R-tansig-16-2	4.070	8.400	12.730	<0.001
eaa-logsig-14-2	zhou-pmc	5.603	9.933	14.264	<0.001
eaa-logsig-14-2	zhou-logsig-16-2	7.536	11.867	16.197	<0.001
eaa-logsig-14-2	zhou-radbas-16-2	8.003	12.333	16.664	<0.001
eaa-logsig-14-2	zhou-tansig-16-2	8.403	12.733	17.064	<0.001
eaa-radbas-16-2	eaa-tansig-16-1	−3.697	0.633	4.964	1.000
eaa-radbas-16-2	quat-pmc	2.836	7.167	11.497	<0.001
eaa-radbas-16-2	quat-logsig-8-2	0.270	4.600	8.930	0.022
eaa-radbas-16-2	quat-radbas-16-1	−1.097	3.233	7.564	0.642
eaa-radbas-16-2	quat-tansig-8-2	0.303	4.633	8.964	0.019
eaa-radbas-16-2	R-pmc	7.303	11.633	15.964	<0.001
eaa-radbas-16-2	R-logsig-16-2	4.770	9.100	13.430	<0.001
eaa-radbas-16-2	R-radbas-2-1	2.303	6.633	10.964	<0.001
eaa-radbas-16-2	R-tansig-16-2	4.770	9.100	13.430	<0.001

(Continues)

TABLE A1 | (Continued)

Method 1	Method 2	CI lower	Mean rank difference	CI upper	*p*-value
eaa-radbas-16-2	zhou-pmc	6.303	10.633	14.964	<0.001
eaa-radbas-16-2	zhou-logsig-16-2	8.236	12.567	16.897	<0.001
eaa-radbas-16-2	zhou-radbas-16-2	8.703	13.033	17.364	<0.001
eaa-radbas-16-2	zhou-tansig-16-2	9.103	13.433	17.764	<0.001
eaa-tansig-16-1	quat-pmc	2.203	6.533	10.864	<0.001
eaa-tansig-16-1	quat-logsig-8-2	−0.364	3.967	8.297	0.140
eaa-tansig-16-1	quat-radbas-16-1	−1.730	2.600	6.930	0.985
eaa-tansig-16-1	quat-tansig-8-2	−0.330	4.000	8.330	0.128
eaa-tansig-16-1	R-pmc	6.670	11.000	15.330	<0.001
eaa-tansig-16-1	R-logsig-16-2	4.136	8.467	12.797	<0.001
eaa-tansig-16-1	R-radbas-2-1	1.670	6.000	10.330	<0.001
eaa-tansig-16-1	R-tansig-16-2	4.136	8.467	12.797	<0.001
eaa-tansig-16-1	zhou-pmc	5.670	10.000	14.330	<0.001
eaa-tansig-16-1	zhou-logsig-16-2	7.603	11.933	16.264	<0.001
eaa-tansig-16-1	zhou-radbas-16-2	8.070	12.400	16.730	<0.001
eaa-tansig-16-1	zhou-tansig-16-2	8.470	12.800	17.130	<0.001
quat-pmc	quat-logsig-8-2	−6.897	−2.567	1.764	0.989
quat-pmc	quat-radbas-16-1	−8.264	−3.933	0.397	0.152
quat-pmc	quat-tansig-8-2	−6.864	−2.533	1.797	0.992
quat-pmc	R-pmc	0.136	4.467	8.797	0.033
quat-pmc	R-logsig-16-2	−2.397	1.933	6.264	1.000
quat-pmc	R-radbas-2-1	−4.864	−0.533	3.797	1.000
quat-pmc	R-tansig-16-2	−2.397	1.933	6.264	1.000
quat-pmc	zhou-pmc	−0.864	3.467	7.797	0.439
quat-pmc	zhou-logsig-16-2	1.070	5.400	9.730	<0.001
quat-pmc	zhou-radbas-16-2	1.536	5.867	10.197	<0.001
quat-pmc	zhou-tansig-16-2	1.936	6.267	10.597	<0.001
quat-logsig-8-2	quat-radbas-16-1	−5.697	−1.367	2.964	1.000
quat-logsig-8-2	quat-tansig-8-2	−4.297	0.033	4.364	1.000
quat-logsig-8-2	R-pmc	2.703	7.033	11.364	<0.001
quat-logsig-8-2	R-logsig-16-2	0.170	4.500	8.830	0.030
quat-logsig-8-2	R-radbas-2-1	−2.297	2.033	6.364	1.000
quat-logsig-8-2	R-tansig-16-2	0.170	4.500	8.830	0.030
quat-logsig-8-2	zhou-pmc	1.703	6.033	10.364	<0.001
quat-logsig-8-2	zhou-logsig-16-2	3.636	7.967	12.297	<0.001
quat-logsig-8-2	zhou-radbas-16-2	4.103	8.433	12.764	<0.001
quat-logsig-8-2	zhou-tansig-16-2	4.503	8.833	13.164	<0.001
quat-radbas-16-1	quat-tansig-8-2	−2.930	1.400	5.730	1.000
quat-radbas-16-1	R-pmc	4.070	8.400	12.730	<0.001
quat-radbas-16-1	R-logsig-16-2	1.536	5.867	10.197	<0.001
quat-radbas-16-1	R-radbas-2-1	−0.930	3.400	7.730	0.495
quat-radbas-16-1	R-tansig-16-2	1.536	5.867	10.197	<0.001
quat-radbas-16-1	zhou-pmc	3.070	7.400	11.730	<0.001

(Continues)

TABLE A1 | (Continued)

Method 1	Method 2	CI lower	Mean rank difference	CI upper	*p*-value
quat-radbas-16-1	zhou-logsig-16-2	5.003	9.333	13.664	<0.001
quat-radbas-16-1	zhou-radbas-16-2	5.470	9.800	14.130	<0.001
quat-radbas-16-1	zhou-tansig-16-2	5.870	10.200	14.530	<0.001
quat-tansig-8-2	R-pmc	2.670	7.000	11.330	<0.001
quat-tansig-8-2	R-logsig-16-2	0.136	4.467	8.797	0.033
quat-tansig-8-2	R-radbas-2-1	−2.330	2.000	6.330	1.000
quat-tansig-8-2	R-tansig-16-2	0.136	4.467	8.797	0.033
quat-tansig-8-2	zhou-pmc	1.670	6.000	10.330	<0.001
quat-tansig-8-2	zhou-logsig-16-2	3.603	7.933	12.264	<0.001
quat-tansig-8-2	zhou-radbas-16-2	4.070	8.400	12.730	<0.001
quat-tansig-8-2	zhou-tansig-16-2	4.470	8.800	13.130	<0.001
R-pmc	R-logsig-16-2	−6.864	−2.533	1.797	0.992
R-pmc	R-radbas-2-1	−9.330	−5.000	−0.670	0.006
R-pmc	R-tansig-16-2	−6.864	−2.533	1.797	0.992
R-pmc	zhou-pmc	−5.330	−1.000	3.330	1.000
R-pmc	zhou-logsig-16-2	−3.397	0.933	5.264	1.000
R-pmc	zhou-radbas-16-2	−2.930	1.400	5.730	1.000
R-pmc	zhou-tansig-16-2	−2.530	1.800	6.130	1.000
R-logsig-16-2	R-radbas-2-1	−6.797	−2.467	1.864	0.996
R-logsig-16-2	R-tansig-16-2	−4.330	0.000	4.330	1.000
R-logsig-16-2	zhou-pmc	−2.797	1.533	5.864	1.000
R-logsig-16-2	zhou-logsig-16-2	−0.864	3.467	7.797	0.439
R-logsig-16-2	zhou-radbas-16-2	−0.397	3.933	8.264	0.152
R-logsig-16-2	zhou-tansig-16-2	0.003	4.333	8.664	0.050
R-radbas-2-1	R-tansig-16-2	−1.864	2.467	6.797	0.996
R-radbas-2-1	zhou-pmc	−0.330	4.000	8.330	0.128
R-radbas-2-1	zhou-logsig-16-2	1.603	5.933	10.264	<0.001
R-radbas-2-1	zhou-radbas-16-2	2.070	6.400	10.730	<0.001
R-radbas-2-1	zhou-tansig-16-2	2.470	6.800	11.130	<0.001
R-tansig-16-2	zhou-pmc	−2.797	1.533	5.864	1.000
R-tansig-16-2	zhou-logsig-16-2	−0.864	3.467	7.797	0.439
R-tansig-16-2	zhou-radbas-16-2	−0.397	3.933	8.264	0.152
R-tansig-16-2	zhou-tansig-16-2	0.003	4.333	8.664	0.050
zhou-pmc	zhou-logsig-16-2	−2.397	1.933	6.264	1.000
zhou-pmc	zhou-radbas-16-2	−1.930	2.400	6.730	0.998
zhou-pmc	zhou-tansig-16-2	−1.530	2.800	7.130	0.937
zhou-logsig-16-2	zhou-radbas-16-2	−3.864	0.467	4.797	1.000
zhou-logsig-16-2	zhou-tansig-16-2	−3.464	0.867	5.197	1.000
zhou-radbas-16-2	zhou-tansig-16-2	−3.930	0.400	4.730	1.000