SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2024, Vol. 67, Iss. 12, 222202:1–222202:15
https://doi.org/10.1007/s11432-023-4107-3

Graph-geometric message passing via a graph convolution transformer for FKP regression

Huizhi ZHU[1], Wenxia XU[1*], Jian HUANG[2] & Baocheng YU[1]

[1]*Hubei Key Laboratory of Intelligent Robot, Wuhan Institute of Technology, Wuhan* 430205*, China;*
[2]*School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan* 430074*, China*

Received 12 June 2023/Revised 31 May 2024/Accepted 18 June 2024/Published online 11 November 2024

Abstract In this paper, the forward kinematics problem (FKP) of the Gough-Stewart platform (GSP) with six degrees of freedom (6 DoFs) is estimated via deep learning. We propose a graph convolution transformer model by systematically analyzing some challenges encountered with using deep learning regression on large-scale data. We attempt to leverage the graph-geometric message as input and singular value decomposition (SVD) orthogonalization for SO(3) manifold learning. This study is the first in which a robot with a sophisticated closed-loop mechanism is described by a graph structure and a specific deep learning model is proposed to solve the FKP of the GSP. Qualitative and quantitative experiments on our dataset demonstrate that our model is feasible and superior to other methods. Our method can guarantee error percentages of translation and rotation less than 1 mm and 1° of 81.9% and 96.7%, respectively.

Keywords deep learning, graph-structured learning, graph convolution transformer, forward kinematics problem, Gough-Stewart platform

1 Introduction

Solving forward kinematics problems (FKPs) is very important for obtaining a valuable mapping relationship between the joint space and the pose of an end-effector, which is necessary for motion control, modeling, and algorithm verification of robots. It is widely known that the FKP of a robot with rigid links can be expressed mathematically. However, it is intractable to calculate the FKP for complex mechanisms with six degrees of freedom (6 DoFs), e.g., the Gough-Stewart platform (GSP) [1, 2].

The FKP of the GSP has previously been solved using numerical or analytical methods [3–8]. However, this problem cannot be adequately addressed using these approaches. The multilink, closed-loop mechanism architecture of the platform and the strong nonlinearity introduced by the mapping relationship result in the platform lacking an FKP solution [2]. Deep learning technology is currently being used to build an end-to-end model that can adapt to this nonlinear spatial mapping relationship to facilitate the transplantation of various mechanisms and the incorporation of numerical algorithms. Existing neural network (NN)-based methods consist of MLPs and use Euler angles as the output representation [2, 9–12]. However, a brief structure and improper rotation representation produce an output with poor accuracy, rendering this design insufficient to meet the requirements of the robot mission and hindering end-to-end output. Numerical methods are typically used to optimize the estimated values of the NN for a single pose [2, 9, 12, 13] but require a large computation time and cannot ensure a high success rate.

On the basis of the empirical studies theory of some specialized visualization tasks, w.r.t., camera relocalization [14–17], 6D object pose estimation (6D OPE) [18–21], and 3D human pose estimation (3D HPE) [22–26], we contend that it is challenging to design and train an end-to-end NN model for this task for two primary reasons. (i) It is challenging to use the end-to-end model to represent and learn the rotation of nonEuclidean space. In particular, learning problems are pervasive in 3D computer vision and robotics [15, 18, 27, 28], such as 3D rotations. In an Euclidean space, $\mathbb{R}^n$, feature vectors from deep neural networks are commonly generated. However, Euclidean spaces are unsuitable for representing the

* Corresponding author (email: xuwenxia@wit.edu.cn)

outcome of many tasks, such as regressing a target on a specified manifold $\mathcal{M}$ (3D transformation groups), i.e., SO(3), SE(3) [29]. SO(3) is not homeomorphic to the output space of an NN, which is an enormous obstacle to deep rotation regression in an end-to-end model. (ii) The decoupled representation of the input data greatly reduces the diversity of model design and impedes learning strategies. Ordinarily, six joint displacements are used as NN inputs. This 6D representation is in Euclidean space but differs from typical image, voice or text input information. The front and back data have limited connectivity, and there is no semantic or multiscale context information. Therefore, the input data representation problem makes it impractical to generalize and learn the network, while complicating the design of the network architecture. To address the aforementioned problems, we employ the following four strategies to reconstruct and optimize the NN for FKP regression.

Selection of a good differentiable rotation representation mapping. Zhou et al. [30] showed that all rotation representations that reside in Euclidean space are discontinuous in four or fewer dimensions. This discontinuous representation makes it difficult for the NN output to approximate the ground truth. Zhou et al. proposed a 6D continuous representation and Gram-Schmidt orthogonalization. Subsequently, Peretroukhin et al. [31] adopted a 10D symmetric matrix representation, whereas Levinson et al. [32] utilized a 9D singular value decomposition (SVD) orthogonalization representation. Brégier [33] systematically analyzed the properties of good differentiable rotation representations, among which SVD orthogonalization usually exhibited the best performance. 9D SVD orthogonalization is also the foundation of the related rotation representation mapping of our model's output.

A graph-geometric message of the mechanism is implemented as the input. "Graph-geometric message" is introduced as a novel term in this paper. A graph structure is a natural and direct way to represent the configuration information for robot mechanisms [34–37]. Kim et al. [37] utilized a graph to represent a robot's structure and a graph neural network (GNN) for learning; however, this learning method was carried out on a robot with a simple linear structure rather than a complex mechanism. The input to the graph convolution network (GCN) is the feature information of the vertices, which is represented by physical coordinates, e.g., 3D HPE [23,26], that is not available for the FKP, for which the known information is instead mainly the geometric linkage information of the GSP. Therefore, the geometric message of the relevant neighbor edges serves as a representation for the vertex information that we input, namely, the graph-geometric message.

Building an advanced GNN model. GNNs are capable of learning the data representation of a nonEuclidean space [38]. To meet the task requirements of the FKP, we design a reasonable graph convolution transformer model. We propose a Chebyshev graph atrous convolution (ChebGAC) to fully extract the multiscale contextual information of the graph-geometric message and retain its related graph-structured information as the network propagates forward. We utilize the appropriate graph convolution and graph transformer to actively support long-range relationship learning for graph-geometric messages while capturing global information without sacrificing the specifics of the graph-structured information.

Designing a suitable loss function. Deep learning techniques generally parameterize a pose using a description that separates rotation from translation [14–17,39]. A matching weighted loss function is established by computing the $\mathscr{L}_p$ norm between the two datasets and their ground truths. Quaternions are traditionally employed as rotation representations to determine the Euclidean distance. However, the output of our rotation focuses on learning for SO(3), i.e., the Frobenius norm between the estimated value $\boldsymbol{R}$ and the ground truth $\boldsymbol{R}_{\rm gt}$.

In this study, we propose an end-to-end model for FKP regression via a graph convolution transformer network[1)] (GCT-FKP). To the best of our knowledge, this study is the first in which a robot with a highly complex closed-loop mechanism is represented by a graph structure and a specific deep learning model is proposed to solve the FKP of the GSP. The contributions of this study are summarized below.

- We propose a novel learning representation as input, namely, the graph-geometric message, which is mainly represented by a graph structure.
- We propose a graph-structured multiscale learning method that leverages multiscale fusion mechanisms, namely, ChebGAC, which can provide global graph-geometric message for subsequent propagation.
- We propose an FKP regression model based on a graph convolution transformer, namely, the GCT-FKP. Our GCT-FKP model is capable of achieving end-to-end high-precision output while facilitating the integration of numerical algorithms to obtain higher-precision poses.

1) Codes. https://github.com/FLAMEZZ5201/GCT-FKP.

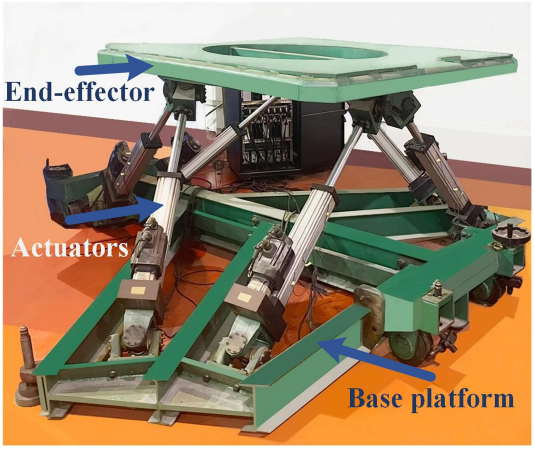

Figure 1 (Color online) Experimental 6-UCU GSP.

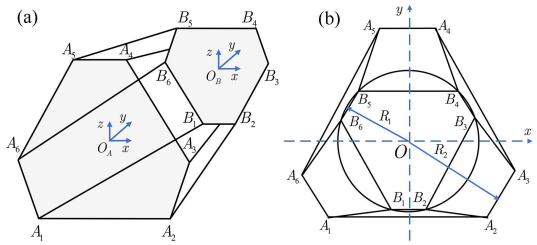

Figure 2 (Color online) Illustration of the 6-UCU GSP. (a) Initial coordinate systems; (b) perspective planar view.

2 Kinematics model and problem statement

2.1 Kinematics model

In this subsection, we use a quaternion representation to build an inverse kinematics model of the GSP. Figure 1 illustrates the employed 6-UCU (U-universal joint; C-cylinder joint) GSP mechanism.

Figure 2 is a schematic of the GSP. The bottom and top hinge points are denoted as A_i and B_i, respectively. The connection between A_i and B_i is defined as $\boldsymbol{l}_i$, $i \in (1,\ldots,6)$. In the GSP system, O_A and O_B denote the origins of the coordinate systems for the base platform and end-effector, respectively.

Determining the end-effector pose $\boldsymbol{p}$ to obtain the joint variable $\boldsymbol{l}_{oi}$ is an inverse kinematics problem (IKP) of the GSP. The displacements of the ith actuator and end-effector pose are as follows:

$$\begin{aligned}\boldsymbol{l}_{oi} &= \begin{pmatrix} l_{o1} & l_{o2} & l_{o3} & l_{o4} & l_{o5} & l_{o6} \end{pmatrix},\\ \boldsymbol{p} &= \begin{pmatrix} \boldsymbol{x} & \boldsymbol{q} \end{pmatrix} = \begin{pmatrix} x & y & z & q_0 & q_1 & q_2 & q_3 \end{pmatrix},\end{aligned} \tag{1}$$

where $\boldsymbol{l}_{oi} \in \mathbb{R}^6$, $\boldsymbol{x} \in \mathbb{R}^3$, and $\boldsymbol{q} \in \mathcal{S}^3$. The mapping relationship between the joint space and the pose space in the IKP is ψ: $\boldsymbol{p} \to \boldsymbol{l}_{oi}$. The IKP of the GSP is represented as follows:

$$\psi \ : \ \boldsymbol{p} \to \boldsymbol{l}_{oi},\ \boldsymbol{l}_{oi} = \overline{l}_j - \overline{l}_i, \tag{2}$$

$$\overline{l}_i = \|\boldsymbol{b}_i - \boldsymbol{a}_i\|_2,\ \overline{l}_j = \|\boldsymbol{c}_i - \boldsymbol{a}_i\|_2, \tag{3}$$

$$\boldsymbol{c}_i = \boldsymbol{T}\overline{b},\ \boldsymbol{T} = \begin{pmatrix} \boldsymbol{R}(q) & \boldsymbol{x} \\ 0 & 1 \end{pmatrix},\ \overline{b} = \begin{pmatrix} \boldsymbol{b}_i & 1 \end{pmatrix}^{\mathrm{T}}, \tag{4}$$

where $\overline{l}_i$ is the length of the actuator in the initial state and $\overline{l}_j$ is the length of the actuator in the current state, $j \in (1,\ldots,6)$; $\boldsymbol{R}(q)$ denotes the rotation matrix $\boldsymbol{R} \in \mathrm{SO}(3)$ obtained by transformation using the quaternion $\boldsymbol{q}$; $\boldsymbol{T}$ is a 4×4 homogeneous transformation matrix; and $\boldsymbol{a}_i = (A_{ix}\ A_{iy}\ A_{iz})$, $\boldsymbol{b}_i = (B_{ix}\ B_{iy}\ B_{iz})$, $\boldsymbol{c}_i = (C_{ix}\ C_{iy}\ C_{iz})$.

2.2 Problem statement

Formally, given a series of joint displacement variables $\boldsymbol{l}_{oi} \in \mathbb{R}^6$, the FKP of the GSP is defined as finding the end-effector pose $\boldsymbol{T} \in \mathrm{SE}(3)$. The MLP [2, 9, 11] has previously been employed to establish this nonlinear mapping relationship as $\mathbb{R}^6 \to \mathbb{R}^6$. Here, we use a neural network to build this mapping relationship to learn a regression function $\mathcal{F}^*$ and minimize the error over a dataset containing an end-effector pose $\boldsymbol{T}$ as follows:

$$\mathcal{F}^* = \arg\min_{\mathcal{F}} \mathcal{L}(\mathcal{F}(\boldsymbol{X}), \boldsymbol{T}), \tag{5}$$

where $\boldsymbol{X} \in \mathbb{R}^{12\times3}$ is the graph-geometric message obtained using the joint displacement variables $\boldsymbol{l}_{oi}$. The mapping relationship finally constructed by the neural network is $\mathbb{R}^{12\times3} \to \mathrm{SE}(3)$.

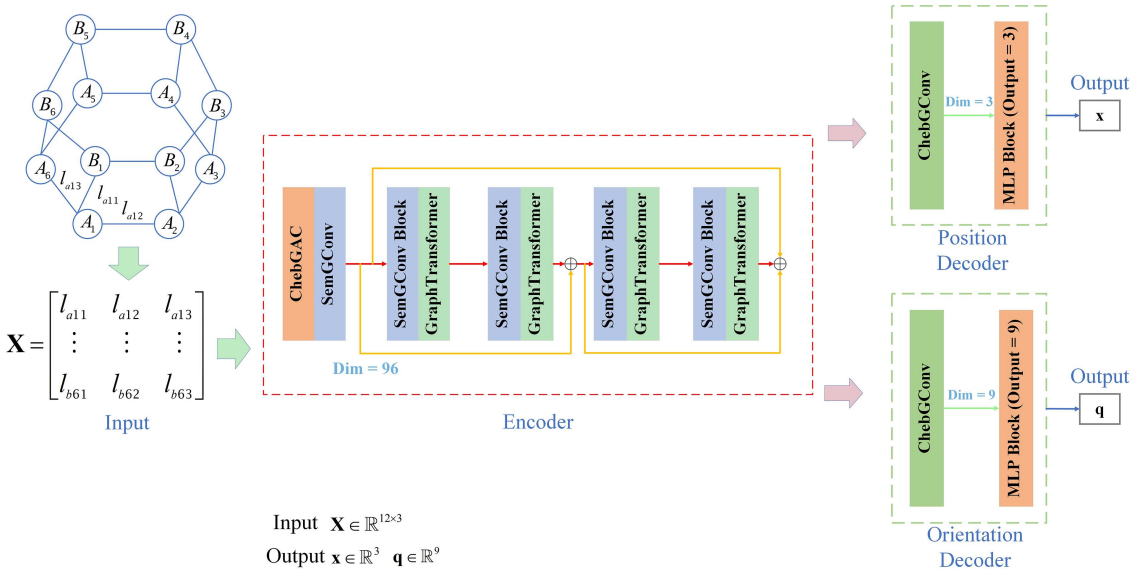

Figure 3 (Color online) Graph convolution transformer network for forward kinematics problem (GCT-FKP). Input information is construsted by "Graph-Geometric message", $\boldsymbol{X} \in \mathbb{R}^{12\times3}$, which first passes through our ChebGAC to get related multiscale contextual information of graph structure. ChebGAC in the encoder produced 12 feature dimensions, whereas ChebGConv in the decoder produced 3 and 9 feature dimensions, respectively. The message passes through the MLP blocks of the various decoders, including the position and orientation decoders, to linearly output estimated values of characteristic data $\boldsymbol{x} \in \mathbb{R}^3$, $\boldsymbol{q} \in \mathbb{R}^9$.

3 Proposed method

3.1 Graph-geometric message

Using a graph structure to record data for the robot configuration data is a natural and direct approach [34–37]. In this study, we utilize the graph structure defined on the GSP given below.

Graph definition. The vertex $\mathcal{V}$ and edge $\mathcal{E}$ of the graph are represented by the 12 hinge joint nodes and their 18 physical connections of the GSP, respectively. The graph structure of the GSP is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{12\times12}$.

In the GNN-based method, the vertex feature vector is generally represented as physical coordinates, which are not known for the FKP of the GSP. However, other geometric information is available, i.e., the geometric length of all connections, that could be used in this method. Each vertex has neighboring edges, where the geometric lengths of three edges can be represented by a corresponding vertex, which is an intriguing aspect of the geometrically symmetric topology of the GSP. One connection variable l_{xy1} and two connection constants $\{l_{xy2}, l_{xy3}\}$ can make up the feature of a vertex, where x denotes the top or bottom hinge vertex and y denotes the order of the vertex. The connection variable can be obtained through the IKP, which is given in (3). The feature $\boldsymbol{X}$ formed by all vertices is called the "graph-geometric message", which serves as the model input. Figure 3 illustrates the input process, where the specific form of $\boldsymbol{X}$ is as follows:

$$\boldsymbol{X} = \begin{pmatrix} l_{a11} & l_{a12} & l_{a13} \\ \vdots & \vdots & \vdots \\ l_{b61} & l_{b62} & l_{b63} \end{pmatrix}_{12\times3}, \tag{6}$$

where a denotes the bottom hinge vertex and b denotes the top vertex.

3.2 ChebGConv and ChebGAC

Defferrard et al. [40] proposed a Chebyshev graph convolution (ChebGConv) to process graph-structured data. ChebGConv requires more computational resources than vanilla GCN [41]. The graph-geometric message we constructed has few vertices because of the reduced complexity of constructing the graph, enabling ChebGConv to be used for feature extraction in the early stages of the network to provide strong

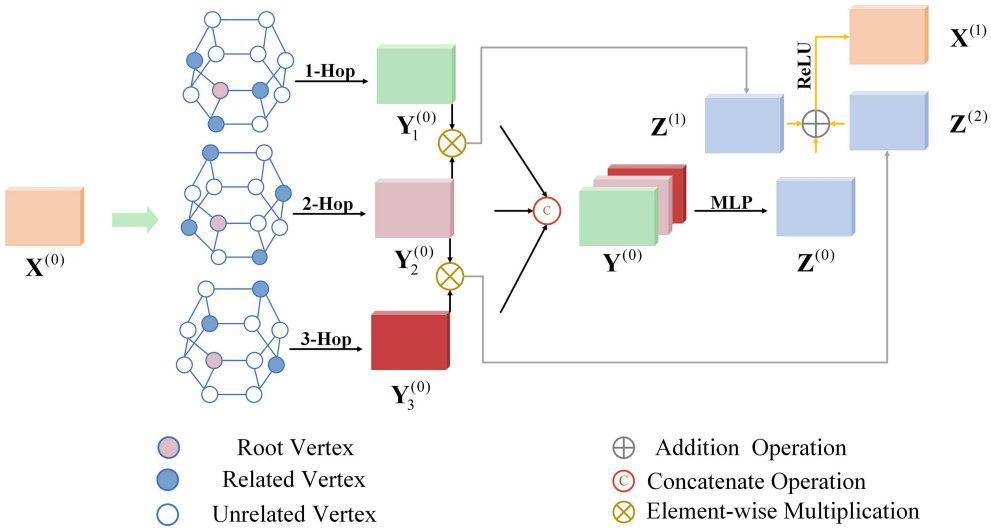

Figure 4 (Color online) Diagram of ChebGAC's parallel convolution for the graph-geometric message.

graph-structured information for subsequent processing. We formally define ChebGConv as follows:

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}, \ \tilde{\boldsymbol{L}} = 2\boldsymbol{L}/\lambda_{\max} - \boldsymbol{I}, \tag{7}$$

$$\boldsymbol{X}^{(l+1)} = \sum_{k=0}^{K-1} \boldsymbol{T}_k(\tilde{\boldsymbol{L}})\boldsymbol{X}^{(l)}\boldsymbol{W}, \tag{8}$$

where $\boldsymbol{I}$ is the identity matrix; $\boldsymbol{D}$ represents the degree matrix of the graph; $\tilde{\boldsymbol{L}} \in \mathbb{R}^{12\times 12}$ is the scaled Laplacian; $\lambda_{\max}$ is the maximum eigenvalue of $\boldsymbol{L}$; $\boldsymbol{T}_k(x)$ denotes the kth-order Chebyshev polynomial; $\boldsymbol{T}_k(x) = 2x\boldsymbol{T}_{k-1}(x) - \boldsymbol{T}_{k-2}(x)$; $\boldsymbol{T}_0(x) = 1$ and $\boldsymbol{T}_1(x) = x$; $\boldsymbol{X}$ is the input feature; and $\boldsymbol{W}$ is a weight matrix.

Traditional GCN [41] only convolves 1-hop neighbors through limited filters and neglects the multiscale contextual information of higher-order neighbors. In response, Zhu et al. [23] proposed GAC, a technique from image segmentation that employs convolution operations with different dilation factors in parallel. However, the multiscale fusion mechanism of GAC is a concatenation-based linear operation that could potentially reduce the representation ability of the model. To address this issue, we modified the multiscale fusion mechanism to provide stronger graph-structured information. Additionally, we replaced vanilla GCN with the 2nd-order ChebGConv to obtain a larger receptive field.

ChebGAC. We propose a Chebyshev graph atrous convolution (ChebGAC) to learn the multiscale contextual information of the graph-geometric message. Unlike GAC [23], our ChebGAC is represented by only 1-hop, 2-hop, and 3-hop neighbors, reducing redundant computations and parameters in the overall ChebGAC. In this study, the dilation factors are defined as the graph distance between vertices. The ChebGAC process for the input graph-geometric message is shown in Figure 4. We formally define ChebGAC as follows:

$$\left[\tilde{\boldsymbol{A}}_k\right]_{i,j} = \begin{cases} 1, \ \mathrm{dist}(v_i, v_j) = k, \\ 1, \ \mathrm{dist}(v_i, v_j) = 0, \\ 0, \ \text{otherwise}, \end{cases} \tag{9}$$

$$\underbrace{\boldsymbol{Y}_k^{(0)} = \sum_{z=0}^{1} \boldsymbol{T}_z(\tilde{\boldsymbol{L}}_k)\boldsymbol{X}^{(0)}\boldsymbol{W}, \ \boldsymbol{Y}^{(0)} = \Big\|_{k=1}^{3} \boldsymbol{Y}_k^{(0)}}_{\text{multiscale contextual information}}, \tag{10}$$

where $\tilde{\boldsymbol{A}}_k$ is the self-loop adjacency matrix with k-hop; $\mathrm{dist}(v_i, v_j)$ is the distance between v_i and v_j corresponding to the shortest path on the graph; $\tilde{\boldsymbol{L}}_k$ represents the rescaled Laplacian matrix with k-hop obtained through different adjacency matrices $\boldsymbol{A}_k$; and $\|$ represents the concatenation operation. The application of ChebGConv changes the number of feature dimensions to 12.

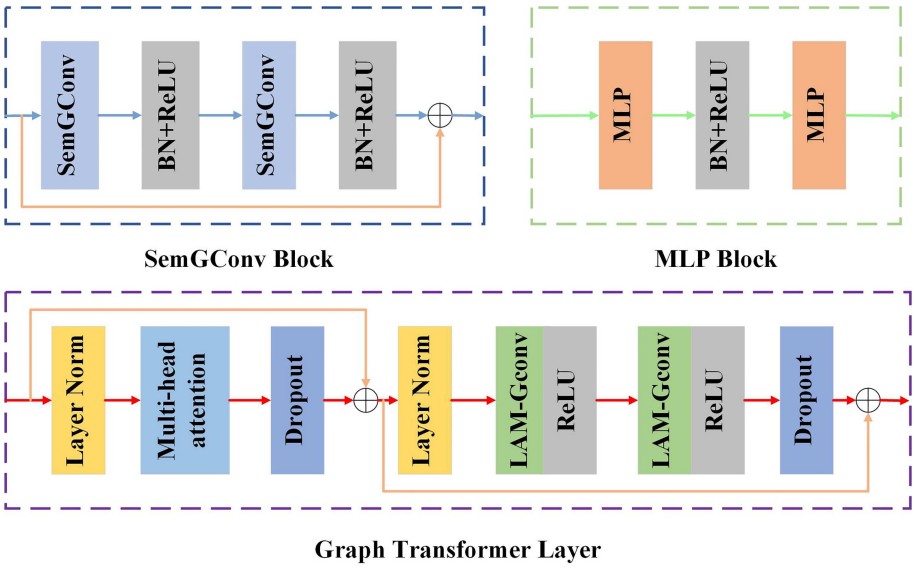

Figure 5 (Color online) Illustration of the related blocks and graph transformer in our GCT-FKP.

Finally, we propose the following multiscale fusion mechanism to aggregate multiscale and global graph-structured features:

$$\begin{cases} \boldsymbol{Z}^{(0)} = \boldsymbol{Y}^{(0)}\boldsymbol{W}, \\ \boldsymbol{Z}^{(1)} = \boldsymbol{Y}_1^{(0)} \otimes \boldsymbol{Y}_2^{(0)}, \\ \boldsymbol{Z}^{(2)} = \boldsymbol{Y}_2^{(0)} \otimes \boldsymbol{Y}_3^{(0)}, \end{cases} \tag{11}$$

$$\boldsymbol{X}^{(1)} = \sigma(\boldsymbol{Z}^{(0)} \oplus \boldsymbol{Z}^{(1)} \oplus \boldsymbol{Z}^{(2)}), \tag{12}$$

where $\otimes$ represents elementwise multiplication; $\oplus$ denotes the addition operation; and $\sigma(\cdot)$ is the ReLU activation function.

3.3 GCT-FKP network

The overall architecture of our GCT-FKP is an encoder to a dual-decoder, as depicted in Figure 3. We introduce each layer of the message passing in this subsection.

SemGConv. After introducing graph structures with multiscale information, we further process the graph via a SemGConv block, which fuses the multiscale information across different layers. SemGConv [22] is a spatial-based graph convolution method that can effectively extract the semantic features of graph structures with low computational effort. SemGConv maps the number of feature dimensions from 12 to 96. Figure 5 illustrates the SemGConv block, where the SemGConv operation is defined as

$$\boldsymbol{X}^{(l+1)} = \sigma(\rho(\boldsymbol{M} \otimes \boldsymbol{A})\boldsymbol{X}^{(l)}\boldsymbol{W}), \tag{13}$$

where ρ represents the Softmax operation, $\boldsymbol{M}$ is a learnable weight matrix, and $\boldsymbol{A}$ is an adjacency matrix that serves as a mask matrix in this equation.

Graph transformer. Although GCN is a powerful method for graph learning, some problems are well-recognized, such as a restricted receptive field and oversmoothing in continuous multilayer GCN, which affect GCN performance. It is widely recognized that the best GCN performance can be achieved with only two layers. That is, a multilayer GCN is not suitable for learning long-range relationships and the GCN performance does not improve with depth. We can mitigate this issue by acquiring multiscale information, which, however, does not enable GCN to adapt to long-range relationship learning. Therefore, we input the semantic features obtained through each SemGConv block into the graph transformer layer. This layer captures long-range relationship information by participating in learning global features rather than identifying local physical connections of the GSP. This step is inspired by 3D HPE [22,23,26], which focuses on the nonlocal features of the graph after graph convolution. The transformer is primarily composed of a feedforward network and a multihead attention module, as proposed by Vaswani et

al. [42]. Our graph transformer is the GraAttention block in GraFormer, as shown in Figure 5. LAM-Gconv [26] is employed by GraAttention to replace the MLP of the transformer, which enhances the training performance and enables robust nonlocal feature information to be captured without losing the graph-structured details. The self-attention mechanism and LAM-Gconv are formally defined as follows:

$$\boldsymbol{H}^{(l)} = \rho\left(\frac{\boldsymbol{Q}^{(l)} \cdot \boldsymbol{K}^{(l)^{\mathrm{T}}}}{\sqrt{d_k}}\right)\boldsymbol{V}^{(l)}, \tag{14}$$

$$\boldsymbol{X}^{(l+1)} = \sigma(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{X}^{(l)}\boldsymbol{W}),\ \hat{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I}, \tag{15}$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n\times d_k}$ are the query, key, and value matrices, respectively; d_k is the dimension of the key vectors; '$\cdot$' is a dot-product operation; and $\hat{\boldsymbol{A}}$ denotes a learnable self-loop adjacency matrix.

Dual-decoder. We designed a dual-decoder structure, namely, a decoder for position and orientation to facilitate learning these two separate data characteristics. The decoders in our model consist of a ChebGConv block and an multilayer perceptron (MLP) block, where the feature dimensions are appropriately converted in the position and orientation decoders. Initially, ChebGConv captures the graph-structured information obtained from the encoder and generates the associated data dimension space; e.g., the feature dimension is converted from dim = 96 to dim = 3, and $\mathbb{R}^{12\times 96} \to \mathbb{R}^{12\times 3}$ in the position decoder. Finally, the graph-structured feature is input into the MLP block (shown in Figure 5) as the linear output of the related dimensions.

3.4 Loss function

In this subsection, we introduce a 9D rotation mapping representation for manifold learning in the GCT-FKP model and our weighted loss function for learning the pose of the end-effector.

SVD orthogonalization. Deep learning-based methods generally generate a high-dimensional Euclidean space that is not suitable for learning an embedded low-dimensional manifold space. Therefore, a differentiable rotation representation is needed for SO(3) regression. Levinson et al. [32] employed SVD orthogonalization, which uses SVD to translate a 9D rotation representation into SO(3), where SVD is also suitable for backpropagation learning. The raw 9D output $\boldsymbol{q}$ is translated to a rotation matrix by $\pi_{9\mathrm{D}}(M(\boldsymbol{q}))$. The manifold mapping $\pi : M(\boldsymbol{q}) \to \boldsymbol{R}$ is defined as follows:

$$\pi_{9D}(M(\boldsymbol{q})) : \mathrm{SVD}^{+}(\boldsymbol{M}) = \boldsymbol{U}\boldsymbol{\Sigma}'\boldsymbol{V}^{\mathrm{T}},\ M(\boldsymbol{q}) : \boldsymbol{q} \in \mathbb{R}^9 \to \boldsymbol{M} \in \mathbb{R}^{3\times 3}, \tag{16}$$

where $\boldsymbol{\Sigma}'$ is $\mathrm{diag}(1, 1, \det(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}))$ and $M(\boldsymbol{q})$ denotes the vector-to-matrix operator.

Learning position and orientation. In regression for pose estimation, translation and rotation are typically separated and the Euclidean distance between the estimated value and the ground truth is calculated as described in previous visualization studies [14–16]. In this study, more emphasis is placed on manifold learning of the differentiable homeomorphic SO(3) mapping. A continuous, smooth, and injective loss is constructed in Euclidean space to learn the position and orientation of the end-effector.

$$\mathcal{L}_{\mathrm{trans}} = \|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{gt}}\|_2, \tag{17}$$

$$\mathcal{L}_{\mathrm{rotation}} = \|\boldsymbol{R} - \boldsymbol{R}_{\mathrm{gt}}\|_F. \tag{18}$$

The objective function is

$$\mathcal{L}_p = \mathcal{L}_{\mathrm{trans}} + \beta\mathcal{L}_{\mathrm{rotation}}, \tag{19}$$

where $\|\cdot\|_2$ denotes the $\mathscr{L}_2$-norm, $\boldsymbol{x}$ is the estimated value, and $\boldsymbol{x}_{\mathrm{gt}}$ is its ground truth; $\|\cdot\|_F$ denotes the Frobenius norm, $\boldsymbol{R}$ is the estimated value, and $\boldsymbol{R}_{\mathrm{gt}}$ is its ground truth; $\mathcal{L}_{\mathrm{rotation}}$ is equal to $4 - 4\cos(\langle \boldsymbol{R}, \boldsymbol{R}_{\mathrm{gt}}\rangle)$; $\langle \boldsymbol{R}, \boldsymbol{R}_{\mathrm{gt}}\rangle$ is the angle between $\boldsymbol{R}$ and $\boldsymbol{R}_{\mathrm{gt}}$; and β is a balancing factor that regulates the trade-off between the penalties associated with position and orientation.

4 Experiments

4.1 Dataset

In this study, we obtain related geometry parameters from the GSP, as shown in Figure 1. The dataset is acquired by establishing the inverse kinematics model. Initially, we obtain the position information of

Table 1 Motion range of the end-effector

Pose variable	Generated dataset size	Motion range	Unit
(x, y, z)	50×10^4	$[-50, 50]$	mm
(rx, ry, rz)	50×10^4	$[-30, 30]$	°

Table 2 Positions of hinge points

Coordinate	Hinge point numbers (mm)					
	1	2	3	4	5	6
A_x	−250.0	250.0	1350.0	1100.0	−1100.0	−1350.0
A_y	1414.5	1414.5	−490.7	−923.7	923.7	−490.7
A_z	0.0	0.0	0.0	0.0	0.0	0.0
B_x	−525.0	525.0	625.0	100.0	−100.0	−625.0
B_y	418.5	418.5	245.3	−663.9	−663.9	245.3
B_z	800.0	800.0	800.0	800.0	800.0	800.0

the hinge points to build the related inverse kinematics model. We randomly generate meaningful pose information for the end-effector of the GSP in a motion-range space. The motion range of the end-effector is presented in Table 1. Instead of immediately determining the displacement of the connections, we use the inverse kinematics algorithm presented in Subsection 2.1 to compute the length of the connections, which facilitates the construction of a graph-geometric message. Table 2 presents the position of the hinge points of the GSP. The aforementioned process creates a large-scale dataset[2)] with a size of 50×10^4. The training and test sets have sizes of 40×10^4 and 10×10^4, respectively, corresponding to a size ratio of 8 : 2.

4.2 Metrics

The evaluation metrics for model performance are typically evaluated separately from those for pose estimation using the deep learning technique. We specifically design two evaluation metrics for the regression of the FKP, E-trans and E-rot, to estimate the performance of translation and rotation, respectively. E-trans corresponds to the average Euclidean distance between the estimated value and ground truth value in the test set. E-trans is formally defined as follows:

$$\text{E-trans} = \frac{1}{N} \sum_{\boldsymbol{x} \in \mathbb{R}^3}^{N} \left\| \boldsymbol{x} - \boldsymbol{x}_{\text{gt}} \right\|_2, \tag{20}$$

where N is the number of samples in the test set, $\boldsymbol{x}$ is the estimated translation vector, $\boldsymbol{x}_{\text{gt}}$ is the ground truth translation vector, and $\|\cdot\|_2$ denotes the $\mathscr{L}_2$-norm.

E-rot corresponds to the average geodesic distance between the estimated value and ground truth value in the test set. E-rot is formally defined as follows:

$$\text{E-rot} = \frac{1}{N} \sum_{\boldsymbol{R} \in \text{SO}(3)}^{N} \left\| \text{Log}(\boldsymbol{R}\boldsymbol{R}_{\text{gt}}^{\text{T}}) \right\|_F, \tag{21}$$

where N is the number of samples in the test set, $\boldsymbol{R}$ is the estimated rotation matrix, $\boldsymbol{R}_{\text{gt}}$ is the ground truth rotation matrix, $\text{Log}(\cdot)$ is the matrix logarithm mapping function, and $\|\cdot\|_F$ denotes the Frobenius norm.

4.3 Implement details

Our GCT-FKP model consists of one ChebGAC layer, four SemGConv blocks, four graph transformer layers with four heads, and a dual-decoder. Our polynomial graph Laplacian is the 2nd order in ChebGConv, and the scale factor β is 200 (although we have not conducted many experiments to identify the optimal β in this study, various values of β will impact the learning outcomes as a whole). The Adam optimizer is chosen as our optimizer, and the learning rate is 0.001. We train our model for 800 epochs with a batch size of 4000. The experiments are carried out using a single RTX 3090 GPU and the PyTorch deep learning framework.

2) Dataset. https://www.kaggle.com/datasets/huizhizhu/gsp-fkdataset.

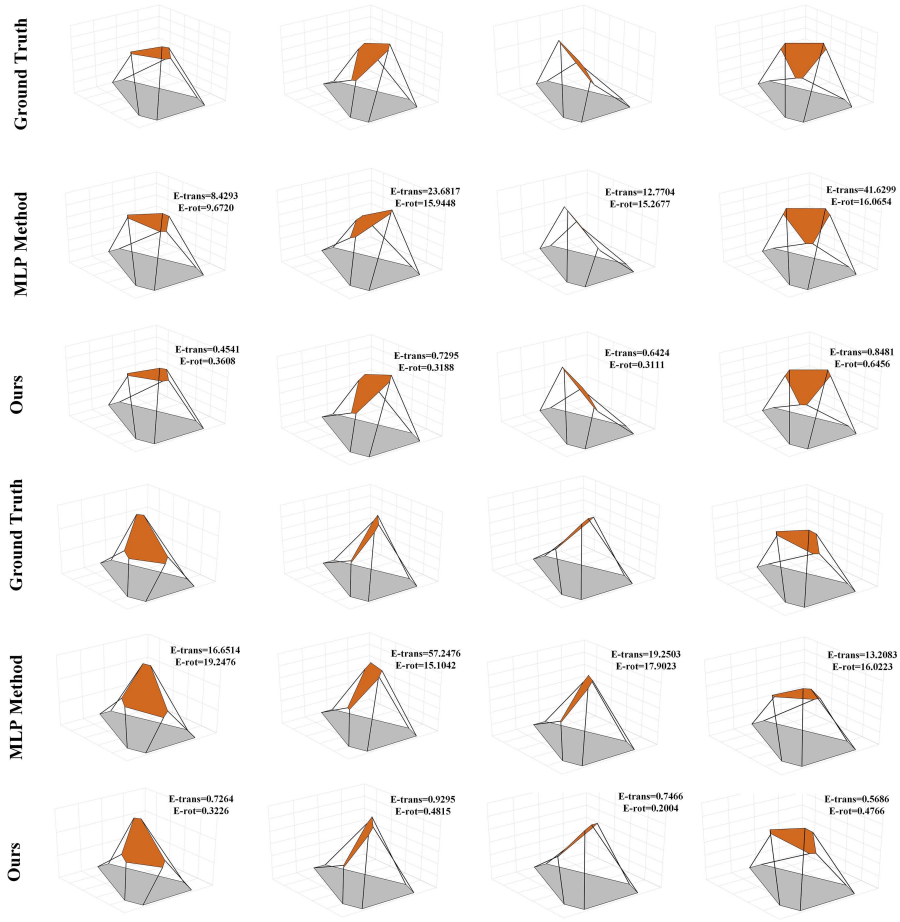

Figure 6 (Color online) Qualitative illustration and comparison for the poses of the GSP's end-effector. The results are obtained by MLP and our GCT-FKP on the test set.

4.4 Experimental results

We perform four types of analyses: qualitative and quantitative analyses, a comparative analysis of different methods, an ablation study, and a comparative analysis of rotation representations. In this study, the primary benchmark used in the comparative analysis is the existing MLP [2,9,11,12] consisting of an MLP model and Euler-angle representation. The MLP method employs mean square error (MSE) to learn the output 6D vector with respect to the ground truth values. In the comparative analysis on different methods, a numerical algorithm (Newton-Raphson (NR) method) is employed as a traditional method and different methods are used to generate the initial values. We evaluate the success rates of the solutions to compare the performance of the methods. We calculate useful comparison metrics, such as the runtime and number of iterations, to accurately assess the feasibility of the methods. Ablation experiments are used to verify the effectiveness of the proposed modules, and rotation representations are compared to identify the optimal regression-friendly rotation representation for the existing representation method.

(a) Qualitative and quantitative analyses. Figure 6 shows representative visualizations of the poses of the GSP's end-effector for a total of eight different poses. We qualitatively compare the end-effector poses of the GSP obtained by applying the MLP (our MLP structure: $\mathbb{R}^6 \to$ (MLP+BN +ReLU)$\times N \to \mathbb{R}^6$, with the one hidden layer $N = 1$, as was consistently used in our previous work) as well as our proposed GCT-FKP method to the test set. The poses of the GSP's end-effector predicted by the GCT-FKP are considerably more accurate than the ground truth annotations. The MLP also obtains poses that are close to the ground truth but are less accurate than GCT-FKP, with visible deviations from the ground truth poses.

Moreover, we quantitatively demonstrate the accuracy under different errors, as shown in Figure 7. We

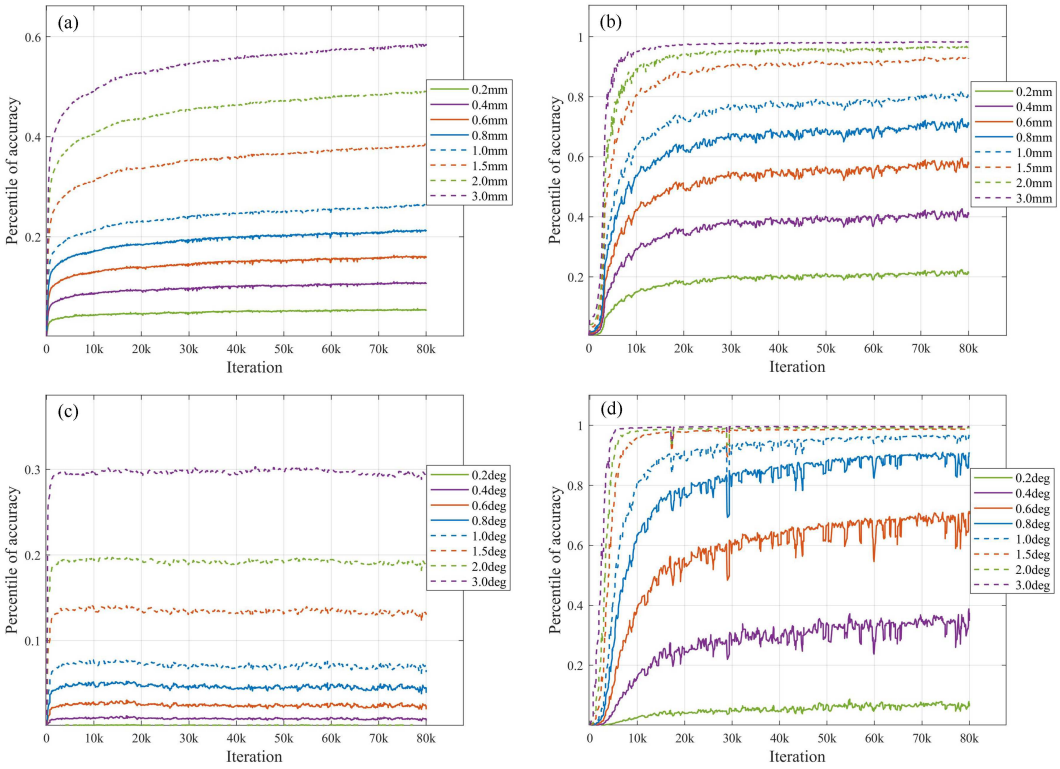

Figure 7 (Color online) Accuracy of test errors at different iterations during training. Each curve refers to a different error. The x-axis represents the number of current iterations, and the y-axis represents the percentage of the test set that is less than this error during the current iteration. (a) The accuracy of translation (MLP method); (b) the accuracy of translation (GCT-FKP method); (c) the accuracy of rotation (MLP method); (d) the accuracy of rotation (GCT-FKP method).

present a comparison between our proposed method and the existing MLP method at eight different levels of accuracy, from the perspective of both translation and rotation errors. To evaluate the performance of the method, the translation error is computed as the absolute error between the estimated translation $\boldsymbol{x}$ and the ground truth translation $\boldsymbol{x}_{\rm gt}$. Similarly, the rotation error is measured by the geodesic distance between the estimated rotation matrix $\boldsymbol{R}$ and the ground truth rotation matrix $\boldsymbol{R}_{\rm gt}$. The accuracy at a given error threshold is obtained by calculating the percentage of values with errors less than the threshold. Obviously, our method can guarantee that the pose errors generated from the dataset are frequently quite small, with the translation errors less than 1 mm and the rotation errors less than 1° accounting for 81.9% and 96.7%, respectively. However, it should be noted that the accuracy of the MLP method is generally low, with only 26.3% of translation errors less than 1 mm and 7.3% of rotation errors less than 1°. Moreover, from the comparison results, we observe that we can achieve higher training saturation with the proposed GCT-FKP method. In contrast, we get a low overall accuracy when using the MLP method, particularly in terms of rotation accuracy represented using Euler angles, which may not meet the requirements of high-precision robotic tasks.

(b) Method comparison. Traditional methods for solving problems can be categorized as analytical [6–8] or numerical [4,5]. Analytical methods are prone to multiple solutions and have complex derivations, which makes these methods unsuitable for obtaining real-time solutions. The Newton-Raphson [43] method is a numerical algorithm that is computationally efficient but limited by the choice of initial values, a low success rate, and high time-consumption. Figure 8 is a comparison of the iterative calculations with different initial values (randomly given (`torch.randn`), MLP-based, and GCT-FKP-based initial values), demonstrating that our proposed method combined with a numerical algorithm has a higher success rate than the MLP for a lower average number of iterations. The Newton-Raphson experiments were conducted on an Intel(R) Core(TM)i7-11800H @ 2.30 GHz in Python for a maximum iteration count of 50 and precision levels set to 10^{-1}, 10^{-2}, 10^{-3}, and 10^{-4}.

Table 3 is a statistical comparison of the error rates for the test set obtained using the MLP with varying numbers of hidden layers and GCT-FKP. Increasing the number of layers ($N = 1, 2, 3, 4$) only improves the MLP performance slightly while the accuracy remains low. In particular, there is a trivial improvement in

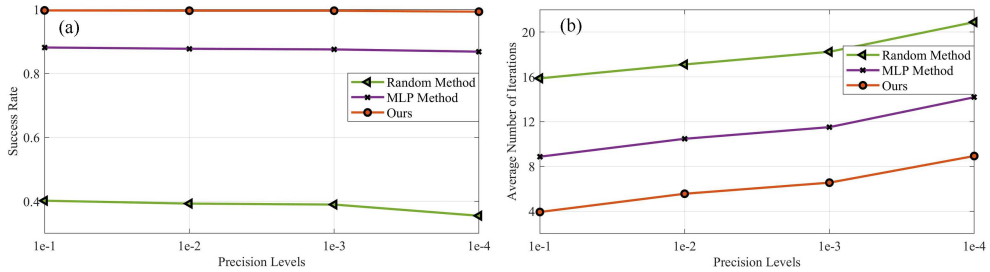

Figure 8 (Color online) Results of providing different initial values to the Newton-Raphson algorithm using various methods on the test set. (a) Success rate; (b) average number of iterations.

Table 3 Results for the statistical comparison of error rates on test set[a)]

Method	Translation error				Rotation error			
	Md↓ (mm)	Mn↓ (mm)	Acc (0.5 mm)↑ (%)	Acc (1 mm)↑ (%)	Md↓ (°)	Mn↓ (°)	Acc (0.5°)↑ (%)	Acc (1°)↑ (%)
MLP (1-layer)	2.33	3.17	11.9	26.3	5.51	6.32	2.1	7.3
MLP (2-layer)	1.72	2.45	16.7	35.1	5.35	6.14	2.2	7.4
MLP (3-layer)	1.34	1.93	21.3	45.8	5.34	6.03	2.3	7.5
MLP (4-layer)	0.95	1.55	28.1	54.8	5.31	6.03	2.3	7.5
MLP (5-layer)	1.08	1.64	23.5	49.6	5.42	6.19	2.2	7.4
GCT-FKP (small)	0.59	0.83	42.7	75.5	0.51	0.58	52.1	91.8
GCT-FKP (ours)	**0.47**	**0.70**	**53.4**	**81.9**	**0.44**	**0.52**	**57.6**	**96.7**

a) The overall best results are presented in bold. Md, Mn, and Acc are abbreviations of median, mean, and accuracy.

Table 4 Results for the performance comparison between different models on test set

Method	Params	Runtime (ms)	Average time (μs)	Average NR runtime (ms)	E-trans	E-rot	Success rate (%)
Random	–	–	–	28.6/50.1/68.4/96.2	–	–	35.5
MLP (1-layer)	0.10M	20.1	5.0	17.5/25.8/35.6/58.7	5.3	6.3	79.8
MLP (2-layer)	0.18M	26.5	6.6	15.8/23.4/34.8/55.9	4.5	6.1	81.2
MLP (3-layer)	0.27M	31.3	7.8	15.1/22.5/33.9/55.1	3.3	6.0	81.9
MLP (4-layer)	0.36M	35.9	8.9	14.8/21.9/33.1/54.6	2.6	6.0	82.7
GCT-FKP (small)	0.29M	235.3	63.3	6.9/14.1/21.6/29.9	1.5	0.58	98.5
GCT-FKP (ours)	0.52M	382.2	95.5	6.1/13.4/19.2/26.7	1.1	0.52	99.3

the accuracy of rotation estimation. For $N = 5$, the network shows signs of overfitting after 800 training epochs, resulting in a decrease in the overall performance compared with the model with four hidden layers. We compared the results obtained using a lightweight version of the GCT-FKP, which has fewer parameters and only includes half of the encoder used for the full GCT-FKP. Specifically, this GCT-FKP version includes one ChebGAC, two SemGConv blocks, two graph transformer layers with four heads, and a dual-decoder. This GCT-FKP version can effectively learn the pose of the end-effector, achieving 75.5% accuracy based on translation errors less than 1 mm and 91.8% accuracy based on rotation errors less than 1°. For a 3-layer MLP with similar parameters, the mean translation error decreases by 56.9%, and the accuracy of the 1-mm error improves by 64.8%. Our method can effectively learn rotation by using a non-Euclidean representation with SVD orthogonalization, considerably outperforming traditional MLP-based methods based on an Euler angle representation. In (d), we compare the results of experiments using different rotation representations.

Considering the requirements for task solving in real time, we compare different methods from other perspectives, e.g., in terms of the runtime and number of parameters. An essential feature of neural networks is the ability to perform parallel computations. To demonstrate the computational efficiency of our method, we conduct parallel computations for a batch size instead of solving for a single pose, enabling us to obtain a short runtime. Table 4 compares the performance of the different models. We set the batch size to 4000 to determine the runtime for different network-based methods. The MLP runtime increases in proportion to the number of parameters, where a 3-layer MLP has an 86.6% shorter runtime than the lightweight CGT-FKP with similar parameters. Our method may not have a shorter runtime than the MLP but can ensure high-precision pose output. This result means that many FK solutions can be directly output through end-to-end parallel computation, which can reduce the computational time of

Table 5 Results for the performance comparison between different methods on test set

Method	Batch size	Runtime	Success rate (%)	Acc (%)	Multiple solutions?	Precision level	Devices
NR [3]	4000	≈ 89.5 s	41.2	–	No	<1 mm and <1°	CPU
MLP [11]	4000	≈ 26.5 ms	–	7.3	No	<1 mm and <1°	GPU
MLP+NR [2]	4000	≈ 63.2 s	89.7	–	No	<1 mm and <1°	GPU+CPU
Ours	4000	≈ 382.2 ms	–	81.9	No	<1 mm and <1°	GPU
Ours+NR	4000	≈ 18.3 s	100.0	–	No	<1 mm and <1°	GPU+CPU

Table 6 Ablation study on the dataset[a)]

Method	Translation error				Rotation error			
	Md↓ (mm)	Mn↓ (mm)	Acc (0.5 mm)↑ (%)	Acc (1 mm)↑ (%)	Md↓ (°)	Mn↓ (°)	Acc (0.5°)↑ (%)	Acc (1°)↑ (%)
w/o ChebGAC	0.49	0.74	51.5	80.1	0.46	0.53	56.2	95.3
w/o Dual-decoder	0.48	0.72	51.9	80.6	0.45	0.53	57.0	95.8
Ours (GCT-FKP)	**0.47**	**0.70**	**53.4**	**81.9**	**0.44**	**0.52**	**57.6**	**96.7**

a) The overall best results are presented in bold. Md, Mn, and Acc are abbreviations of median, mean and accuracy.

Table 7 Rotation representation comparison for FKP regression[a)]

Method	Rotation error			
	Md↓ (°)	Mn↓ (°)	Acc (0.5°)↑ (%)	Acc (1°)↑ (%)
Eluer (baseline) [2]	5.05	5.53	3.3	8.6
Quaternion [44]	1.65	1.90	6.5	24.7
6D [30]	0.62	0.67	43.4	92.1
10D [31]	0.51	0.56	47.8	94.8
Ours (9D)	**0.44**	**0.52**	**57.6**	**96.7**

a) The overall best results are presented in bold. Md, Mn, and Acc are abbreviations of median, mean and accuracy.

high-precision poses for individual points via a numerical algorithm. Compared with the 3-layer MLP, the lightweight CGT-FKP reduces E-trans and E-rot by 54.4% and 90.3%, respectively. We determine the average runtime for using NR at different precision levels with initial values provided by different methods and consider the success rate at a precision level of 10^{-4} as an additional metric. Our method ensures an average NR runtime interval of 6.1 ms to 26.7 ms for precision levels ranging from 10^{-1} to 10^{-4} with a 99.3% success rate at a precision level of 10^{-4}. Compared with strategies that use random generation or an MLP for providing initial values, our method considerably reduces the required computation time of NR and improves the success rate. That is, our method achieves end-to-end high-precision prediction while performing ultrahigh-precision pose estimation with good compatibility with numerical algorithms. Finally, Table 5 compares the performance of the different methods in terms of the runtime, success rate, and accuracy.

(c) Ablation study. We perform ablation experiments on the modules proposed in Section 3, i.e., ChebGAC and the dual-decoder module. To verify these modules, we train two variants of GCT-FKP: a version without the ChebGAC module and a version that uses only one layer of the MLP to output the features of the encoder, where the dimensions change from $\mathbb{R}^{12\times 96}$ to $\mathbb{R}^{12}$. All the models are trained on the architecture after 800 epochs. The results are shown in Table 6. The full model exhibits the best performance. Adding ChebGAC increases the error accuracy (for 1 mm and 1°) by 1.8% and 1.4%, respectively, indicating that the ChebGConv and multiscale fusion mechanism effectively complement the multiscale graph-structured information in the graph-geometric message. Employing a dual-decoder increases the error accuracy (1 mm and 1°) by 1.6% and 0.9%, respectively.

(d) Comparison of rotation representations. Table 7 [44] compares several rotation representations that are based on the GCT-FKP model. The 9D representation model achieves the best performance for orientation regression on the test set, whereas the Euclidean representation has considerable difficulty with learning the rotation characteristics. The mean error of the 9D representation is 90.5% lower than that of the Euler angles.

5 Discussion and conclusion

Simple structure or complex structure? The advantages and disadvantages of simple and complex structures should be considered in selecting a model structure. Simple models, such as MLPs, have fewer parameters and lower computational complexity, and are therefore easy to implement and faster to run. However, simple models have limited expressive power and are usually suitable for tasks with simple model requirements. For the FKP of a robot, it is challenging to train models to predict the pose of the end-effector. The pose of the end-effector in space is infinite, and it is difficult to obtain a model with generalization ability and strong performance on a limited dataset (it is often challenging to acquire high-quality, large-scale datasets for robot learning, and few datasets are publicly available). Simple models have insufficient power to meet the requirement of high-precision output of robot poses, as demonstrated in Section 4. Although IKP can be used to generate the dataset for the GSP infinitely, the performance of the simple model may not be improved infinitely (the use of MLPs often produces problems such as gradient vanishing, underfitting, and overfitting (no residual connections)). Training on infinitely large datasets is impractical because immense computational power is needed to accommodate and train GPUs. By contrast, complex structures generally have better expressive power and performance, are suitable for tasks that require processing large-scale and high-dimensional data, and have higher requirements for model accuracy, which is more appropriate for the FKP of a robot. For example, graph representation is a natural and direct way for robots to learn, and GNNs typically have strong expressive power and generalization ability [45–47]. The high accuracy requirements of robot kinematics make models with complex structures more suitable for solving the FKP while ensuring that the computational time of the model is not excessive.

Limitations and future work. We propose a novel approach for solving the forward kinematics of complex robots from a learning-based perspective. Our method ensures high accuracy in predicting most of the poses and outperforms both the MLP and traditional numerical methods. However, our approach has two limitations. (i) Our approach lacks a learning-based solution system to meet the requirements of different robot tasks. Our novel approach to solving the FKP has not been tailored to a specific robot task. Neural network inference requires high-performance GPUs, making parallel computing with large-scale data suitable for the proposed method; however, high-performance computing resources would be wasted on tasks that only require a single FKP solution to be output. Therefore, it is essential to design a reasonable and universal learning-based solution system. (ii) To meet the requirements of different real-time tasks, a network with a shorter runtime and high-precision output of poses is needed. Although a short runtime and high accuracy were achieved in this study using parallel computing, further research needs to be performed on applying the proposed method to high-demand tasks, such as efficient and ultrahigh-precision control.

Our proposed method for solving the FKP of complex robots is very interesting, different from traditional methods, and has unique advantages and more potential. Implementing a learning-based end-to-end model not only helps to solve the FKP of complex mechanisms but also helps to apply and design new complex mechanisms. We believe that our learning-based kinematics method can be transplanted to different robot structures and can solve more complex kinematics problems in the future.

Conclusion. In this study, deep learning-based regression is carried out for the forward kinematics of sophisticated mechanisms using a Gough-Stewart platform as an example. We analyze the main challenge of using deep learning to solve this novel regression problem and propose matching strategies. Among these strategies, we describe the graph-structured data of the parallel robot by generating a "graph-geometric message" of the mechanism to facilitate the implementation of an advanced graph-convolution-transformer neural network for pose learning. We recommend employing regression-friendly rotation representations for learning SO(3) manifolds based on 9D SVD orthogonalization.

Acknowledgements This work was supported by National Natural Science Foundation Youth Fund of China (Grant No. 61803286) and Innovation Fund Project of Hubei Key Laboratory of Intelligent Robot (Grant No. HBIRL202210).

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Dasgupta B, Mruthyunjaya T S. The Stewart platform manipulator: a review. Mech Mach Theor, 2000, 35: 15–40
2 Parikh P J, Lam S S Y. A hybrid strategy to solve the forward kinematics problem in parallel manipulators. IEEE Trans Robot, 2005, 21: 18–25

3 Nguyen C C, Zhou Z L, Antrazi S S, et al. Efficient computation of forward kinematics and Jacobian matrix of a Stewart platform-based manipulator. In: Proceedings of the SOUTHEASTCON'91, 1991. 2: 869–874
4 Merlet J P. Solving the forward kinematics of a gough-type parallel manipulator with interval analysis. Int J Robot Res, 2004, 23: 221–235
5 Nanua P, Waldron K J, Murthy V. Direct kinematic solution of a Stewart platform. IEEE Trans Robot Automat, 1990, 6: 438–444
6 Ji P, Wu H T. A closed-form forward kinematics solution for the 6-6/sup p/ Stewart platform. IEEE Trans Robot Automat, 2001, 17: 522–526
7 Huang X G, Liao Q Z, Wei S M. Closed-form forward kinematics for a symmetrical 6-6 Stewart platform using algebraic elimination. Mech Mach Theor, 2010, 45: 327–334
8 Lee T Y, Shim J K. Forward kinematics of the general 6-6 Stewart platform using algebraic elimination. Mech Mach Theor, 2001, 36: 1073–1085
9 Zhu H Z, Xu W X, Yu B C, et al. A novel hybrid algorithm for the forward kinematics problem of 6 DOF based on neural networks. Sensors, 2022, 22: 5318
10 Zubizarreta A, Larrea M, Irigoyen E, et al. Real time direct kinematic problem computation of the 3PRS robot using neural networks. Neurocomputing, 2018, 271: 104–114
11 Yee C S, Lim K B. Forward kinematics solution of Stewart platform using neural networks. Neurocomputing, 1997, 16: 333–349
12 Parikh P J, Lam S S. Solving the forward kinematics problem in parallel manipulators using an iterative artificial neural network strategy. Int J Adv Manuf Technol, 2009, 40: 595–606
13 Yahia I B, Merlet J P, Papegay Y. Mixing neural networks and the Newton method for the kinematics of simple cable-driven parallel robots with sagging cables. In: Proceedings of IEEE International Conference on Advanced Robotics (ICAR), 2021. 241–246
14 Kendall A, Grimes M, Cipolla R. PoseNet: a convolutional network for real-time 6-DOF camera relocalization. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2015. 2938–2946
15 Kendall A, Cipolla R. Geometric loss functions for camera pose regression with deep learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 5974–5983
16 Li Q, Zhu J S, Cao R, et al. Relative geometry-aware siamese neural network for 6DOF camera relocalization. Neurocomputing, 2021, 426: 134–146
17 Walch F, Hazirbas C, Laura L T, et al. Image-based localization using LSTMs for structured feature correlation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 627–637
18 Xiang Y, Schmidt T, Narayanan V, et al. PoseCNN: a convolutional neural network for 6D object pose estimation in cluttered scenes. In: Proceedings of Robotics: Science and Systems (RSS), 2018
19 Peng S D, Liu Y, Huang Q X, et al. PVNet: pixel-wise voting network for 6DoF pose estimation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 4561–4570
20 Kehl W, Manhardt F, Tombari F, et al. SSD-6D: making RGB-based 3D detection and 6D pose estimation great again. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2017. 1521–1529
21 Tyree S, Tremblay J, To T, et al. 6-DoF pose estimation of household objects for robotic manipulation: an accessible dataset and benchmark. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS), 2022. 13081–13088
22 Zhao L, Peng X, Tian Y, et al. Semantic graph convolutional networks for 3D human pose regression. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 3425–3435
23 Zhu Y R, Xu X, Shen F M, et al. PoseGTAC: graph transformer encoder-decoder with atrous convolution for 3D human pose estimation. In: Proceedings of International Joint Conferences on Artificial Intelligence, 2021. 1359–1365
24 Toshev A, Szegedy C. DeepPose: human pose estimation via deep neural networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014. 1653–1660
25 Zou Z M, Liu K K, Le W, et al. High-order graph convolutional networks for 3D human pose estimation. In: Proceedings of British Machine Vision Conference, 2020
26 Zhao W X, Wang W Q, Tian Y J. GraFormer: graph-oriented transformer for 3D pose estimation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 20438–20447
27 Omran M, Lassner C, Gerard P M, et al. Neural body fitting: unifying deep learning and model based human pose and shape estimation. In: Proceedings of IEEE International Conference on 3D Vision (3DV), 2018. 484–494
28 Chen J Y, Yin Y D, Birdal T, et al. Projective manifold gradient layer for deep rotation regression. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022. 6646–6655
29 Teed Z, Deng J. Tangent space backpropagation for 3D transformation groups. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021. 10338–10347
30 Zhou Y, Barnes C, Lu J W, et al. On the continuity of rotation representations in neural networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 5745–5753
31 Peretroukhin V, Giamou M, Greene W N, et al. A smooth representation of belief over SO(3) for deep rotation learning with uncertainty. In: Proceedings of Robotics: Science and Systems (RSS), 2020
32 Levinson J, Esteves C, Chen K F, et al. An analysis of SVD for deep rotation estimation. In: Proceedings of Conference on Neural Information Processing Systems, 2020. 33: 22554–22565
33 Brégier R. Deep regression on manifolds: a 3D rotation case study. In: Proceedings of IEEE International Conference on 3D Vision (3DV), 2021. 166–174
34 Zhao A, Xu J, Konaković-Luković M, et al. RoboGrammar: graph grammar for terrain-optimized robot design. ACM Trans Graph, 2020, 39: 1–16
35 Marić F, Giamou M, Hall A W, et al. Riemannian optimization for distance-geometric inverse kinematics. IEEE Trans Robot, 2022, 38: 1703–1722
36 Baca G, Jose A, Yerpes A, et al. Modelling of modular robot configurations using graph theory. In: Proceedings of the 3rd International Workshop on Hybrid Artificial Intelligence Systems, 2008. 649–656
37 Kim J T, Park J, Choi S, et al. Learning robot structure and motion embeddings using graph neural networks. 2021. ArXiv:2109.07543
38 Bronstein M M, Bruna J, LeCun Y, et al. Geometric deep learning: going beyond Euclidean data. IEEE Signal Process Mag, 2017, 34: 18–42

39 Kendall A, Cipolla R. Modelling uncertainty in deep learning for camera relocalization. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2017. 4762–4769

40 Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of Conference on Neural Information Processing Systems, 2016

41 Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of International Conference on Learning Representations (ICLR), 2017

42 Vaswani A, Shazeer N, Parmar N. Attention is all you need. In: Proceedings of Conference on Neural Information Processing Systems, 2017

43 Ypma T J. Historical development of the Newton-Raphson method. SIAM Rev, 1995, 37: 531–551

44 Zhu H Z, Xu W X, Wu J. Deep regression on quaternion: a forward kinematics neural network for study six-DoF pose. In: Proceedings of IEEE International Conference on Cyborg and Bionic Systems (CBS), 2023. 432–437

45 Xu K, Zhang M Z, Li J L, et al. How neural networks extrapolate: from feedforward to graph neural networks. In: Proceedings of International Conference on Learning Representations (ICLR), 2021

46 Xu K, Hu W, Leskovec J, et al. How powerful are graph neural networks? In: Proceedings of International Conference on Learning Representations (ICLR), 2019

47 Li P, Leskovec J. The expressive power of graph neural networks. In: Graph Neural Networks: Foundations, Frontiers, and Applications. Singapore: Springer, 2022. 63–98