Kinematic Motion Retargeting via Neural Latent Optimization for Learning Sign Language

Haodong Zhang, Weijie Li, Jiangpin Liu, Zexi Chen, Yuxiang Cui, Yue Wang, and Rong Xiong

***Abstract*—Motion retargeting from a human demonstration to a robot is an effective way to reduce the professional requirements and workload of robot programming, but faces the challenges resulting from the differences between humans and robots. Traditional optimization-based methods are time-consuming and rely heavily on good initialization, while recent studies using feedforward neural networks suffer from poor generalization to unseen motions. Moreover, they neglect the topological information in human skeletons and robot structures. In this paper, we propose a novel neural latent optimization approach to address these problems. Latent optimization utilizes a decoder to establish a mapping between the latent space and the robot motion space. Afterward, the retargeting results that satisfy robot constraints can be obtained by searching for the optimal latent vector. Alongside with latent optimization, neural initialization exploits an encoder to provide a better initialization for faster and better convergence of optimization. Both the human skeleton and the robot structure are modeled as graphs to make better use of topological information. We perform experiments on retargeting Chinese sign language, which involves two arms and two hands, with additional requirements on the relative relationships among joints. Experiments include retargeting various human demonstrations to YuMi, NAO, and Pepper in the simulation environment and to YuMi in the real-world environment. Both efficiency and accuracy of the proposed method are verified.**

***Index Terms*—Learning from demonstration, imitation learning.**

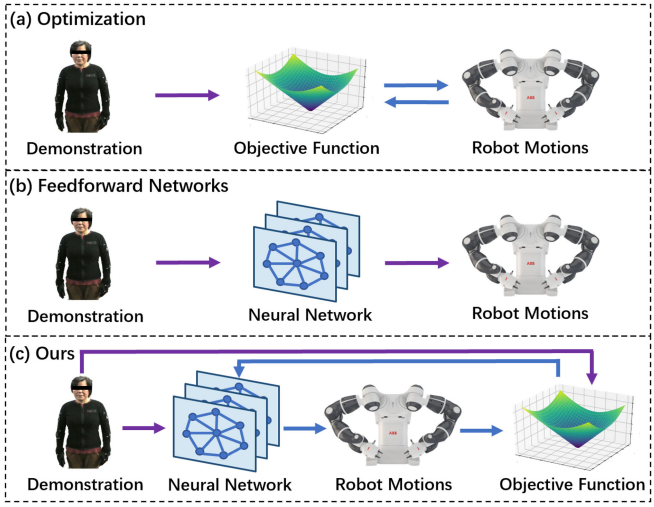

Fig. 1. Illustration of different motion retargeting methods. (a) Optimization methods. (b) Feedforward network methods. (c) Our method. Purple arrows denote feedforward pass, and blue arrows denote iterative pass.

I. Introduction

MOTION retargeting simplifies robot programming by learning human demonstrations, which can effectively reduce the requirement of programming expertise and enable rapid learning of complex robot movements. In this paper, we focus on generating kinematically feasible robot motions, which can help robots express specific information or emotion with body language. Nowadays, it has been applied to humanoid robots in entertainment parks [1] and sign language robots for communication with the hearing impaired [2]. Moreover, it could be used for service robots in museums or restaurants to interact with people using body movements. Specifically, we perform motion retargeting on the task of unseen sign language, which includes complex dual-arm movements and finger movements. However, it still remains an ongoing challenge due to the differences between humans and robots. Even for humanoid manipulators with structures similar to human beings, there exist differences in degrees of freedom, kinematic parameters and physical constraints. Together with various requirements of similarity, safety and rapidity, the problem becomes difficult to solve.

Previous work has been developed to address this problem. Direct mapping [3] transforms human motions through a human-defined mapping relationship, but is hard to define manually. Methods based on inverse kinematics [4] have been used to keep the end effector positions of the robot consistent with those of the human, whereas it does not consider the robot constraints and the similarity of other joints. To overcome these problems, optimization-based methods have been proposed to find the optimal solution that maximizes the motion similarity and satisfies the robot execution capability [2], [5], [6]. It is usually achieved by defining and optimizing an objective function with constraints. Although these methods are able to produce promising retargeting results, they have to take extensive time to optimize each motion and poor initialization may lead to a bad local minimum.

Manuscript received September 9, 2021; accepted January 27, 2022. Date of publication February 14, 2022; date of current version February 28, 2022. This letter was recommended for publication by Associate Editor N. Figueroa and Editor J. Kober upon evaluation of the reviewers' comments. This work was supported by the National Nature Science Foundation of China under Grant 62173293. *(Corresponding authors: Yue Wang; Rong Xiong.)*

The authors are with the State Key Laboratory of Industrial Control and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: aqz@zju.edu.cn; liweijie@zju.edu.cn; jiangpin@zju.edu.cn; chenzexi@zju.edu.cn; yuxiangcui@zju.edu.cn; wangyue@iipc.zju.edu.cn; rxiong@zju.edu.cn).

Digital Object Identifier 10.1109/LRA.2022.3151433

Latent motion is one of the research hotspots in recent years, and there is some work applying latent space to different tasks [7]–[9]. Ichter *et al.* [7] introduced an RRT-based algorithm to plan motions directly in the latent space for visual planning and humanoid robot planning. Watter *et al.* [8] proposed a locally linear latent dynamics model for control from raw pixel images. However, these methods cannot be directly applied in the task of motion retargeting, since they do not take into account motion similarity. A reinforcement learning based method [10] tends to address the problem by designing rewards for retargeted motions. Choi *et al.* [11] proposed a data-driven motion retargeting method that utilizes the results of an optimization-based method as ground truth for training. However, due to limited training data, these methods may perform poorly on unseen motions, which results in inaccurate or infeasible robot movements.

In this paper, we propose a neural latent optimization method that leverages the advantages of both neural networks and optimization, as shown in Fig. 1. Specifically, we first formulate motion retargeting as a constrained optimization problem and transform it to an unconstrained one with the help of a deep decoder. The decoder learns the mapping from the latent space to the robot motion space. For any unseen motion, we further search for the optimal latent code that minimizes the objective function. To accelerate the optimization process and help converge to better results, we exploit a deep encoder to generate better initial values for the latent code. The human skeleton and robot structure are modeled as graphs, which can make better use of topological information and have better generalization. The graph encoder and graph decoder are trained end-to-end. To the best of our knowledge, this is the first work utilizing neural latent optimization for motion retargeting from a human to a robot.

Our contributions can be summarized as follows:

- propose a novel kinematic motion retargeting framework by integrating latent optimization and neural initialization, and build a Chinese sign language dataset for human-robot motion retargeting.
- introduce latent optimization to improve the performance on unseen motions, and design neural initialization to provide better initial values for the latent vector, which helps converge faster and better.
- model the human skeleton and robot structure as graphs, which can capture the inherent information of the topological structure and have better generalization.

II. Related Work

A. Learning From Demonstration

Learning from demonstration has been a fundamental problem in robotics research and holds great promise for industrial applications, since it enables robots to mimic human skills without programming [12]. Recent studies have shown progress in learning from demonstration. Hamaya *et al.* [13] proposed a method for learning soft robotic assembly strategies by designing reward functions from human demonstrations. Lee *et al.* [14] proposed a mixed generative adversarial imitation learning approach, which incorporates both expert demonstrations and negative demonstrations. Cai *et al.* [15] introduced a framework to learn skills from human demonstrations in the form of geometric nullspaces, which infers a parameterized constraint-based skill model independent of the robot kinematics or environment. Different from these approaches, we aim to address the human-robot motion retargeting problem, where the robot mimics human motions while maintaining motion similarity and satisfying the robot's kinematic constraints.

B. Optimization-Based Motion Retargeting

Previous work has attempted to solve the kinematic motion retargeting problem with optimization. By optimizing a predefined objective function of kinematics similarity in an iterative manner, these approaches minimize the gap between robot motions and human demonstrations. Liang *et al.* [2] proposed a motion retargeting method that leverages graph optimization and Dynamic Movement Primitives. Wang *et al.* [6] presented a generative framework for motion retargeting using a single depth sensor. Choi *et al.* [5] designed a motion retargeting pipeline composing of motion retarget descriptions, optimization, inverse kinematics and trajectories post-processing. However, these methods are computationally expensive and require a long iterative process for optimization. Moreover, due to the complex objective function, they are also prone to fall into a bad local minimum without a good initial value. As a comparison, we propose to optimize on the latent space rather than the robot motion space by utilizing a graph decoder. We also introduce a graph encoder to provide a better initial value for optimization, which can accelerate the iterative process.

C. Learning-Based Motion Retargeting

Most learning-based methods focus on motion retargeting of animation characters in computer graphics. Aberman *et al.* [16] introduced differentiable skeleton-aware convolution, pooling and unpooling operators for unpaired motion retargeting. They also presented a method for retargeting video-captured motion [17]. Villegas *et al.* [18] proposed a recurrent neural network with a forward kinematic layer and cycle consistency based adversarial training objective. Although these methods have achieved impressive results in animation characters, they cannot be directly applied to human-robot motion retargeting because they do not take into account the robot's kinematic constraints and would produce infeasible robot movements. For human-robot kinematic motion retargeting, Kim *et al.* [10] developed a cyclic three-phase optimization method based on deep reinforcement learning. Choi *et al.* [11] proposed a data-driven motion retargeting method combining the nonparametric regression and deep latent variable modeling. However, since the quantity and diversity of training data are limited, these methods may not perform well in unseen motions, resulting in inaccurate or infeasible robot motions. As a comparison, our method further optimizes the latent code for any unseen motion, which can improve the performance and combine the advantages of neural networks and optimization. Moreover, our method takes into account not only arm movements, but also complex finger movements.

III. Problem Statement

In this section, we focus on the problem of motion retargeting which requires to generate kinematically feasible robot motions, and formulate the problem as minimizing an objective function, which considers motion similarity and robot kinematic constraints. Suppose we have a frame of the human demonstration, denoted as D, which represents the poses of all human joints at this frame. Then the goal of motion retargeting is to minimize the difference between the robot motion and the human demonstration. While there exist several different ways to represent the robot motion, we propose to use joint angles instead of joint positions, and then calculate joint positions with a differentiable forward kinematics module. Such a representation can avoid changes of joint length and multiple solutions of inverse kinematics. Hence the robot motion can be represented by $K(\boldsymbol{r})$, where $\boldsymbol{r}$ is the robot joint angles and K is the forward kinematics module. Finally, the goal of motion retargeting can be formulated as follows:

$$\begin{aligned} &\underset{\boldsymbol{r}}{\text{minimize}} \;\; L(D, K(\boldsymbol{r})) \\ &\text{subject to } \boldsymbol{r}_{lower} \leq \boldsymbol{r} \leq \boldsymbol{r}_{upper} \end{aligned} \tag{1}$$

where L is the objective function that measures the difference between robot motions and human demonstrations, $\boldsymbol{r}_{upper}$ and $\boldsymbol{r}_{lower}$ are the upper and lower limits of the joint angle.

In order to encourage robot motions to be as similar as possible to human demonstrations and kinematically feasible, the overall objective function is composed of five terms: end effector loss L_{ee}, orientation loss L_{ori}, elbow loss L_{elb}, finger loss L_{fin} and collision loss L_{col}, where λ_{ee}, λ_{ori}, λ_{elb}, λ_{fin} and λ_{col} are their weights with sizes of 1000, 100, 100, 100 and 1000, respectively. The overall objective function needs to be modified to adapt to other tasks. For example, for the task of object grasping and manipulation, the end effector loss and the finger loss should be redesigned and an additional pose estimation module is needed.

$$\begin{aligned} L = \lambda_{ee} L_{ee} + \lambda_{ori} L_{ori} + \lambda_{elb} L_{elb} \\ + \lambda_{fin} L_{fin} + \lambda_{col} L_{col} \end{aligned} \tag{2}$$

1) End Effector Loss: The end effector loss L_{ee} encourages the robot to match end effector positions of human demonstrations, and compares the difference of the normalized end effector positions, using mean square error. The normalization coefficient is the actual length from the shoulder to the wrist. Let $\boldsymbol{p}_j$ and l_j be the position and the normalization coefficient of the end effector j, and $\hat{\boldsymbol{p}}_j$ and $\hat{l}_j$ be the corresponding variables of the demonstration. Then L_{ee} is defined as:

$$L_{ee} = \sum_j || \frac{\boldsymbol{p}_j}{l_j} - \frac{\hat{\boldsymbol{p}}_j}{\hat{l}_j} ||_2^2 \tag{3}$$

2) Orientation Loss: The orientation loss L_{ori} is calculated by comparing the difference of end effector orientations, using mean square loss as well. Let $\boldsymbol{R}_j$ and $\hat{\boldsymbol{R}}_j$ be the end effector rotation matrix of the robot and the human respectively. Then L_{ori} is given as:

$$L_{ori} = \sum_j ||\boldsymbol{R}_j - \hat{\boldsymbol{R}}_j||_2^2 \tag{4}$$

3) Elbow Loss: The elbow loss L_{elb} encourages the movements of other joints to be similar to the demonstration. It is calculated by comparing the difference of the normalized vectors from the elbow to the wrist. The normalization coefficient is the actual length from the elbow to the wrist. Let $\boldsymbol{p}_j^{wrist}$, $\boldsymbol{p}_j^{elbow}$, and l_{we} be the wrist position, the elbow position, and the normalization coefficient of the robot's arm j, and $\hat{\boldsymbol{p}}_j^{wrist}$, $\hat{\boldsymbol{p}}_j^{elbow}$, and $\hat{l}_{we}$ be the corresponding variables of the demonstration. Then L_{elb} is calculated as:

$$L_{elb} = \sum_j || \frac{\boldsymbol{p}_j^{wrist} - \boldsymbol{p}_j^{elbow}}{l_{we}} - \frac{\hat{\boldsymbol{p}}_j^{wrist} - \hat{\boldsymbol{p}}_j^{elbow}}{\hat{l}_{we}} ||_2^2 \tag{5}$$

3) Finger Loss: The finger loss L_{fin} encourages the robot to match the finger movements of the human, which are an important part of sign language. It compares the vectors from the metacarpophalangeal joint to the fingertip, normalized by the finger length. Let $\boldsymbol{p}_j^{tip}$, $\boldsymbol{p}_j^{meta}$, and l_{tm} be the fingertip position, the metacarpophalangeal joint position, and the normalization coefficient of the robot's finger j, and $\hat{\boldsymbol{p}}_j^{tip}$, $\hat{\boldsymbol{p}}_j^{meta}$, and $\hat{l}_{tm}$ be the corresponding variables of the demonstration. Then L_{fin} is defined as:

$$L_{fin} = \sum_j || \frac{\boldsymbol{p}_j^{tip} - \boldsymbol{p}_j^{meta}}{l_{tm}} - \frac{\hat{\boldsymbol{p}}_j^{tip} - \hat{\boldsymbol{p}}_j^{meta}}{\hat{l}_{tm}} ||_2^2 \tag{6}$$

4) Collision Loss: The collision loss L_{col} is designed to penalize robot motions that result in collisions. We model the links of the robotic arms as capsules and calculate the distance between pairs of capsules. If this distance is less than the minimum distance without collision, the loss is calculated. Let $d_{i,j}$ be the distance between the capsule i and the capsule j, and d_{min} be the threshold of no collision. Then L_{col} is given as:

$$L_{col} = \sum_{d_{i,j} < d_{min}} e^{-d_{i,j}^2} \tag{7}$$

IV. Neural Latent Optimization

A class of traditional methods attempt to solve the constrained optimization problem in (1) by adding the joint limit loss L_{lim} as a soft constraint. The new objective function L_{new} is formulated as (8). However, the objective function is highly nonconvex, making these methods difficult to solve well. The main reasons are as follows: 1) poor initialization may lead to a bad local minimum as well as a time-consuming optimization process; 2) the optimization results may exceed the joint limits because L_{lim} may not be minimized to zero.

$$L_{new} = L + \lambda_{lim} L_{lim} \tag{8}$$

In contrast, our method aims to overcome these drawbacks with the help of latent optimization and neural initialization. As shown in Fig. 2, we propose to optimize in the latent space instead of the robot motion space, where the latent vectors are mapped to hard constrained robot motions by a graph decoder.

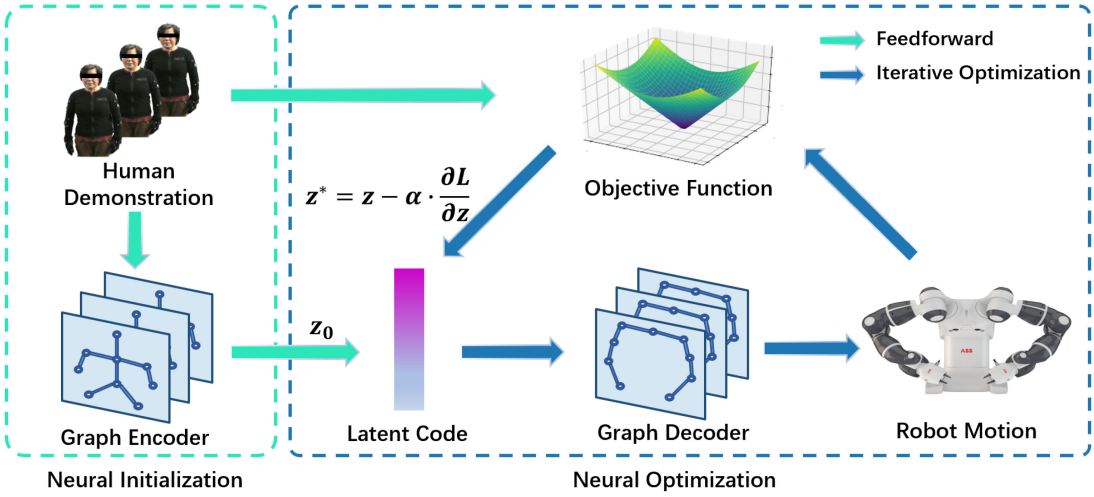

Fig. 2. Overall framework of our method. We establish a mapping relationship between the latent space and the robot motion space with a graph decoder. Utilizing the difference between human demonstrations and robot motions, we search for the optimal latent code $\boldsymbol{z}^*$ that minimizes the gap by gradient descent. The initial value of the latent code $\boldsymbol{z}_0$ is provided by neural initialization instead of random initialization, which improves the convergence speed.

Furthermore, we utilize a deep graph encoder to provide better initialization, which improves the convergence speed and results.

A. Latent Optimization

We first describe how to convert the constrained optimization problem into an unconstrained one. In order to constrain the output joint angles, a practical way is to use a nonlinear activation function. Here we use $tanh$ to bound the output to a finite range. Then we linearly remap the output to the upper and lower limits of the robot's joint angles, which ensures that the joint limit constraints are satisfied.

1) Auto-decoder: To improve the performance on unseen motions, we propose to employ an auto-decoder [19] architecture, where we choose to optimize the objective function in the latent space rather than the robot motion space. To achieve this, we introduce a latent vector $\boldsymbol{z}$ as a latent embedding of the robot motion. Different latent codes correspond to different robot motions. A deep decoder f_ψ is utilized to establish the connection between the latent space and the robot motion space, where ψ is the learnable parameters of the decoder. The goal of the decoder is to map the latent vector to the corresponding robot motion, which adheres the joint limit constraints with the help of the nonlinear activation function. A simple approach to train the decoder is to jointly optimize the parameters of the decoder f_ψ and the latent vector $\boldsymbol{z}$ by gradient descent as (9). The model simultaneously learns the mapping function and finds the optimal latent code for each motion. The regularization term is designed to encourage the latent space to be close to the a multivariate Gaussian distribution with a mean of 0 and a variance of σ. The variance σ is chosen as 1, similar to the Variational Auto-encoders [20]. More details of the motivation of using latent space can be found in the Appendix [21].

$$\min_{\boldsymbol{z},\psi} L(D, K(f_\psi(\boldsymbol{z}))) + \frac{1}{\sigma^2}||\boldsymbol{z}||_2^2 \tag{9}$$

2) Optimization-based Inference: At inference time, the initial value of $\boldsymbol{z}$, denoted as $\boldsymbol{z}_0$, is sampled from the Gaussian distribution as (10). We keep the parameters of the decoder fixed and search for the optimal latent code $\boldsymbol{z}^*$ by gradient descent as (11). It is an iterative process as follows: 1) sampling an initial value $\boldsymbol{z}_0$; 2) decoding the corresponding robot motion; 3) calculating the loss of the objective function; 4) optimizing the latent code by gradient descent. In this way, we generate the best possible results for any unseen motion.

$$\boldsymbol{z}_0 \sim \mathcal{N}(0,\, \sigma^2) \tag{10}$$

$$\boldsymbol{z}^* = \text{argmin}_{\boldsymbol{z}} L(D, K(f_\psi(\boldsymbol{z}))) + \frac{1}{\sigma^2}||\boldsymbol{z}||_2^2 \tag{11}$$

After searching for the optimal latent code for each motion, our method can provide promising results for unseen motions. However, it still has some shortcomings due to random initialization. It is time-consuming to start the optimization process from random values. Moreover, different ways of random initialization may lead to different convergence results, and it may easily fall into a bad local minimum without a good initial value.

B. Neural Initialization

1) Encoder As Initialization: To overcome the shortcomings of random initialization, we introduce a parametrized encoder f_ϕ to provide a better initialization for faster and better convergence, where ϕ is the parameters of the encoder. The encoder takes the human demonstration as input and generates a better initial value $\boldsymbol{z}_0$, which should be close to the optimal latent code $\boldsymbol{z}^*$ so that latent optimization will take fewer iterations to converge and reduce the probability of falling into a bad local minimum. At training time, we optimize the parameters of the encoder f_ϕ and the decoder f_ψ simultaneously:

$$\min_{\psi,\phi} L(D, K(f_\psi(f_\phi(D)))) + \frac{1}{\sigma^2}||f_\phi(D)||_2^2 \tag{12}$$

At inference time, unlike feedforward networks that output the results directly, the encoder finds a good initial value $\boldsymbol{z}_0$ for each human demonstration as (13). Then latent optimization starts from this initial value $\boldsymbol{z}_0$ and searches for the best latent code by gradient descent as (14). The maximum number of iterations is set to 100, and the stopping criterion is that the maximum number of iterations is reached or the loss does not decrease in five consecutive iterations.

$$\boldsymbol{z}_0 = f_\phi(D) \tag{13}$$

$$\boldsymbol{z}^* = \text{argmin}_{\boldsymbol{z}} L(D, K(f_\psi(\boldsymbol{z}))) + \frac{1}{\sigma^2}||\boldsymbol{z}||_2^2 \tag{14}$$

C. Graph-Based Encoder & Decoder

A common choice for designing the encoder and decoder is to concatenate the features of all joints and pass them through a fully connected network. However, the human and robot structures are rich in topological information, which may be lost in the fully connected layer. It is a more natural way to represent the human skeleton and robot structure as graphs, where each node corresponds to a joint of the human or robot, and each edge represents the connection relationship between the joints. The graph representation is invariant to node and edge permutations and has proved its effectiveness of extracting relevant information of the human body skeleton in many tasks [22]–[24]. Its advantage is further discussed in the Appendix [21]. In the task of motion retargeting, we choose to represent the data as graphs rather than vectors to better capture topological information.

Consider a graph $G = (V, E)$ that contains a set of N nodes with feature dimension C_1, denoted by $V \in \mathbb{R}^{N \times C_1}$, and a set of M edges with feature dimension C_2, denoted by $E \in \mathbb{R}^{M \times C_2}$. The graph may be a human skeleton graph or a robot structure graph. Let $\boldsymbol{x}_i$ be the feature of node i, $N(i)$ be the set of neighbor nodes of node i, and $\boldsymbol{e}_{j,i}$ be the edge feature from node j to node i. For each neighbor node j of node i, we first concatenate their node features and edge features, denoted as $\boldsymbol{z}_{i,j} = [\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{e}_{j,i}]$. Then we obtain the message from each neighbor node by nonlinear mapping and aggregate all the information by summation. Finally, we update the feature of node i in a residual way by adding the aggregation result to the original feature. The update of node features is formulated as follows:

$$\boldsymbol{x}_i{}' = \boldsymbol{x}_i + \sum_{j \in N(i)} g(\boldsymbol{W}_f \boldsymbol{z}_{i,j} + \boldsymbol{b}_f) \tag{15}$$

where g is the LeakyReLU activation function, $\boldsymbol{W}_f$ and $\boldsymbol{b}_f$ are learnable parameters.

The constructed human skeleton graph and robot structure graphs of three different robots are shown in Fig. 3. Specifically, in the human skeleton graph, the input features of each node are the human joint positions and rotations, while the edge features are the offsets between human joints. On the other hand, in the robot structure graph, the output features of each node represent the rotation angle of the corresponding robot joint, while the edge features are the initial offsets and rotations between robot links. Both the human skeleton graph and the robot structure graph are directed graphs. We assume that there are two types of nodes, depending on whether the joint is located in the arm or in the hand. Nodes of the same type share the same graph convolution weights. By stacking multiple layers of graph convolution, the model can utilize the inherent information of the topological structure and learn deep-level features more effectively.

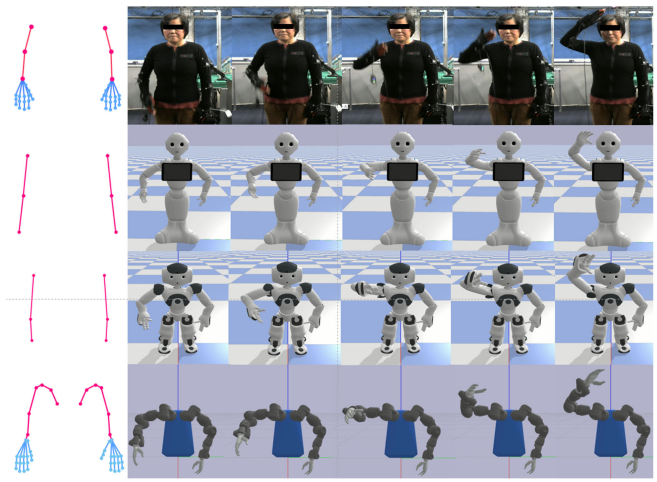

Fig. 3. Snapshots of "Can" sequence transferred by our proposed method on three different robots in the simulation environment. From the first row to the last row are the human demonstrator, Pepper, NAO, and YuMi, respectively. The first column is the corresponding human skeleton graph or robot structure graph.

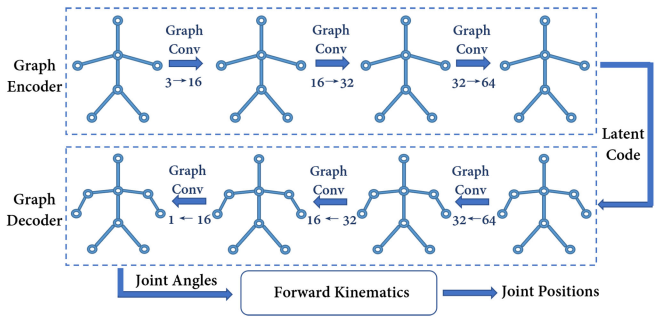

Fig. 4. Network architecture.

D. Network Architecture

The network architecture is shown in Fig. 4. The model consists of a graph encoder, a graph decoder, and a forward kinematics module. The graph encoder is composed of three layers of graph convolution, which extend the channels of the input features from 3 to 16, 32, and 64. Since the number of nodes in the human skeleton graph may differ from the robot structure graph, we use a linear layer to map the output features of the graph encoder to the latent code. The latent code is fed into the graph decoder after concatenation with the upper and lower limits of robot joints.

The graph decoder is also composed of three layers of graph convolutions, which refine the channels of the output features to 32, 16, and 1. With the $tanh$ activation function, the output features of the decoder are constrained to the range -1 to 1. Then they are linearly remapped to the upper and lower limits of each robot joint. Thus each joint angle is ensured to be within

the joint limits of the robot. Finally, the output joint angles are converted to joint positions using a forward kinematics layer. The forward kinematics is utilized as a differentiable but learning free network module, which allows us to optimize on the objective function.

V. Experiments

A. Experimental Setup

We conduct experiments on motion retargeting from Chinese sign language to three different robots, including ABB's YuMi dual-arm robot, NAO, and Pepper. ABB's YuMi dual-arm robot contains 14 degrees of freedom and is equipped with Insipire-Robotics' dexterous hands, each with 6 degrees of freedom and 12 joints. For NAO and Pepper, we use only their robotic arms, which contain 10 degrees of freedom. The robot parameters are parsed from the unified robot description format, which includes the initial offsets, initial rotations, and upper and lower limits of the joint angles. The origin of the robot coordinate system is set at the center of its two shoulder joints, with the Z-axis in the vertical direction, the X-axis in the forward direction, and the Y-axis in the left-hand direction.

The Chinese sign language is performed by a professional sign language teacher whose shoulder, elbow, and wrist poses are captured by a high-precision optical system called OptiTrack. And the finger trajectories are obtained by data gloves called WiseGlove. The position data are normalized with a fixed coefficient and the rotation data are expressed in terms of Euler angles. The human coordinate system is set up in the same way as the robot. The total data contains sign language sequences of 5 daily scenarios with a total of 86 sequences and 13,847 frames. The statistics of the collected dataset can be found in the Appendix [21]. We split the data into a training set and a test set, where the training set includes sequences from 3 scenarios, "Business", "Railway Station" and "West Lake", and the test set includes sequences from 2 scenarios, "Hospital" and "Introduction". Therefore, the total training data contains 61 sign language sequences and 9,691 frames and the testing data contains 25 sign language sequences and 4,156 frames. The collected dataset will be open source soon.

During training, we use a batch size of 16 and an Adam optimizer with a constant learning rate of 1e-4. The training process takes 1,301 s until the model converges with an NVIDIA TITAN X GPU and an Intel(R) Xeon(R) E5-2696 CPU.

B. Ablation Study

1) Initialization Selection: To evaluate the benefit of using a deep graph encoder to provide initial values for latent vectors, we compare the performance of neural initialization and random initialization in the task of retargeting unseen motions in the test set to YuMi. We use three Gaussian distributions for random initialization as comparisons, two of which have a mean of 0 and a variance of 0.1, 0.2, and the other uses the mean and variance of the training set. From Fig. 5, we could see that: 1) different initialization methods will lead to different convergence results; 2) random initialization may fall into a local minimum; 3) the deep graph encoder is able to provide a better initial value that helps to converge faster and better.

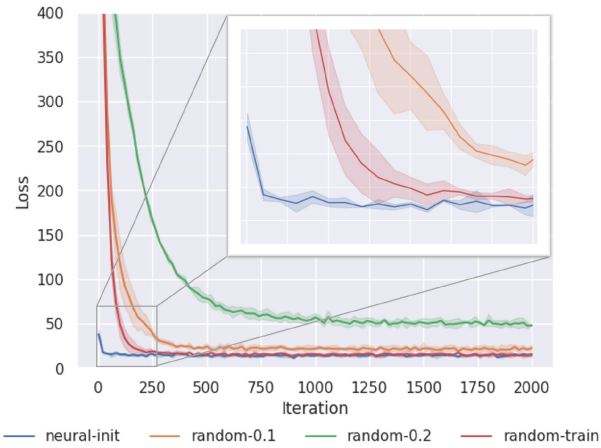

Fig. 5. Convergence results of neural initialization and random initialization of Gaussian distributions with a mean of 0 and a variance of 0.1, 0.2, and the training set's mean and variance.

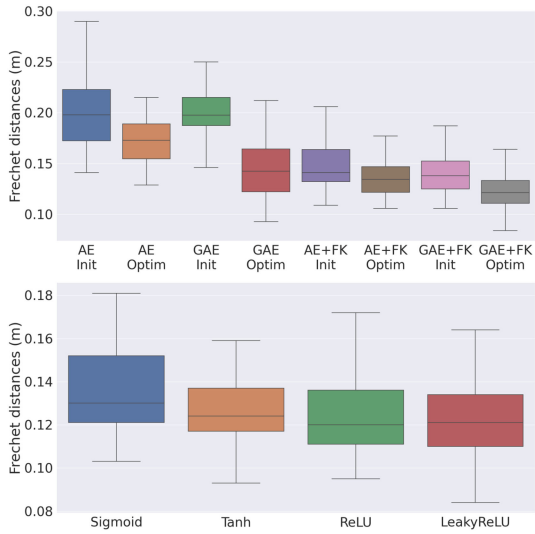

Fig. 6. Frechet distances of different network architectures for unseen motions in the test set. More quantitative results can be found in the Appendix [21].

2) Network Architecture: To validate the effectiveness of the network architecture, we conduct a comparative experiment on the use of graph representation, forward kinematics and latent optimization. We refer to the versions with and without graph representation as 'AE' and 'GAE'. 'FK' denotes the use of forward kinematics. 'Optim' and 'Init' indicates with and without latent optimization. Details can be found in the Appendix [21]. We also compare the retargeting results of LeakyReLU with several common activation functions. The experiment is performed on the task of retargeting unseen motions in the test set to YuMi. From Fig. 6, we could see that: 1) the use of graph representation helps generalize better on unseen motions; 2) the use of forward kinematics as a differentiable but learning free network module helps optimize on the objective function; 3) optimization in the latent space further improves the performance; 4) Tanh,

Fig. 7. Snapshots of motions transferred by our proposed method on the physical YuMi robot. The first two rows are "I can speak sign language" and the last two rows are "Give me this ability".

ReLU and LeakyReLU achieve similar results and LeakyReLU is better than the others.

C. Case Study

In Fig. 3, we first demonstrate the results of motion retargeting from a human demonstrator to three different types of robots in the simulation environment built with PyBullet [25]. Based on their structural proximity to the human demonstrator, the complexity of motion retargeting to these robots is ranked in ascending order as Pepper, Nao, and YuMi. Specifically, Pepper and Nao have fewer degrees of freedom and the proportion of their robot links is closer to the human, while YuMi is equipped with dexterous hands and has a more complex structure. The results show that our method can be applied to robots with different structures, which include different degrees of freedom, joint lengths, and joint limits. Furthermore, our method performs well not only on robots with relatively simple structures, but also on the robot with a significantly complex structure.

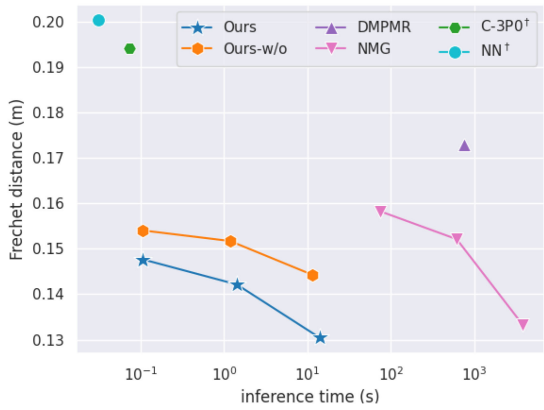

Fig. 8. Comparison between our neural latent optimization method and the baselines. We refer to our method as 'Ours' and the version without using the graph-based network design as 'Ours-w/o'. The closer the data points are to the lower left corner, the better the retargeting results, which means higher similarity is achieved with less inference time. Methods with the superscript † have produced infeasible robot motions in unseen motions of the test set. More quantitative results can be found in the Appendix [21].

D. Comparative Study

To thoroughly evaluate the performance of our approach, we compare against four representative baselines on unseen motions in the test set:

- **DMPMR** [2] is an optimization-based method that combines graph optimization with Dynamic Movement Primitives (DMPs). It adopts a three-step optimization procedure to generate joint angles of robotic arms and calculates finger joint angles using linear mapping.
- **NMG** [5] is an optimization-based pipeline that utilizes link length modifications and task space fine-tuning. It consists of four steps: descriptions preparation, motion optimization, inverse kinematics and post-processing.
- **C-3PO** [10] is a cyclic three-phase deep reinforcement learning method. It adopts the proximal policy optimization (PPO) [26] algorithm to learn the motion retargeting skill in quantitative and qualitative manners.
- **NN** is a feedforward neural network that takes all joint poses of the human body as input and directly outputs the joint angles of the robot. It is trained in a supervised learning manner with the optimization results of DMPMR as ground truth, similar to [11].

We conduct an experiment to retarget unseen motions in the test set to the YuMi robot and compare the tracking error of the wrist and elbow as well as the optimization time. The tracking error is measured by the average Frechet distances [27] of the

wrist and elbow trajectories. The smaller the Frechet distance, the more similar the robot motion is to the demonstration. Details of it can be found in the Appendix [21]. To better visualize the results, we compare both the average Frechet distance and the optimization time in Fig. 8. We notice that our proposed method is able to provide promising results in unseen motions with less optimization time compared with optimization-based methods. Moreover, as the iterations of latent optimization increase, our method is able to achieve better motion retargeting results. We also observe that C-3PO and NN perform worse than optimization-based methods, since they may suffer from generalization to unseen motions, leading to some inaccurate or infeasible results. NN has the worst performance, which indicates that using the optimization output as the ground truth for training will lead to cumulative error consisting of optimization error and fitting error.

E. Real-World Qualitative Experiment

We perform the real-world experiment on the ABB's YuMi dual-arm collaborative robot equipped with Insipire-Robotics' dexterous hands. The movement of the robotic arms and the dexterous hands is controlled by an external computer. As shown in Fig. 7, the robot motion generated by our proposed method is smooth and very close to that of the demonstrator. The smoothness is further discussed in the Appendix [21].

VI. Conclusion

This paper proposes a human-robot motion retargeting method that generates kinematically feasible robot motions based on neural latent optimization. By optimizing in the latent space, our method can achieve better results compared to feedforward neural networks. With the better initialization provided by our deep graph encoder, our method can converge faster than previous optimization-based methods. Experimental results show that our method can generate smooth and similar motions that can be performed on physical robots. Future work will improve the limitation that it does not consider physical contact with the environment.

References

[1] S. Hoshyari, H. Xu, E. Knoop, S. Coros, and M. Bächer, "Vibration-minimizing motion retargeting for robotic characters," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–14, 2019.

[2] Y. Liang, W. Li, Y. Wang, and R. Xiong, Y. Mao, and J. Zhang, "Dynamic movement primitive based motion retargeting for dual-arm sign language motions," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 8195–8201,.

[3] L. Penco *et al.*, "Robust real-time whole-body motion retargeting from human to humanoid," in *Proc. IEEE-RAS 18th Int. Conf. Humanoid Robots* , 2018, pp. 425–432.

[4] T. Asfour and R. Dillmann, "Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (Cat. No 03CH37453)*, 2003, vol. 2, pp. 1407–1412.

[5] S. Choi and J. Kim, "Towards a natural motion generator: A pipeline to control a humanoid based on motion data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4373–4380.

[6] S. Wang, X. Zuo, R. Wang, F. Cheng, and R. Yang, "A generative human-robot motion retargeting approach using a single depth sensor," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5369–5376.

[7] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2407–2414, Jul. 2019.

[8] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, *arXiv:1506.07365*.

[9] J. Merel *et al.*, "Neural probabilistic motor primitives for humanoid control," 2018, *arXiv:1811.11711*.

[10] T. Kim and J.-H. Lee, "C-3PO: Cyclic-three-phase optimization for human-robot motion retargeting based on reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8425–8432.

[11] S. Choi, M. Pan, and J. Kim, "Nonparametric motion retargeting for humanoid robots on shared latent space," in *Proc. Robot.: Sci. Syst.*, Jul. 2020.

[12] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, p. 17, 2018.

[13] M. Hamaya *et al.*, "Learning soft robotic assembly strategies from successful and failed demonstrations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 8309–8315.

[14] G. Lee, D. Kim, W. Oh, K. Lee, and S. Oh, "MixGAIL: Autonomous driving using demonstrations with mixed qualities," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5425–5430.

[15] C. Cai, Y. S. Liang, N. Somani, and W. Yan, "Inferring the geometric nullspace of robot skills from human demonstrations," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 7668–7675.

[16] K. Aberman, P. U. Li, D. Lischinski, O. Sorkine-Hornung, D. Cohen-Or, and B. Chen, "Skeleton-aware networks for deep motion retargeting," *ACM Trans. Graph.*, vol. 39, no. 4, p. 62, 2020.

[17] K. Aberman, R. Wu, D. Lischinski, B. Chen, and D. Cohen-Or, "Learning character-agnostic motion for motion retargeting in 2D," 2019, *arXiv:1905.01680*.

[18] R. Villegas, J. Yang, D. Ceylan, and H. Lee, "Neural kinematic networks for unsupervised motion retargetting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8639–8648.

[19] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.

[21] H. Zhang *et al.*, "Kinematic motion retargeting via neural latent optimization for learning sign language," 2021, *arXiv:2103.08882*.

[22] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 7444–7452.

[23] M. Li, S. Chen, Y. Zhao, Y. Zhang, Y. Wang, and Q. Tian, "Dynamic multiscale graph neural networks for 3D skeleton based human motion prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 214–223.

[24] A. Zeng, X. Sun, L. Yang, N. Zhao, M. Liu, and Q. Xu, "Learning skeletal graph neural networks for hard 3D pose estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 11436–11445.

[25] E. Coumans and Y. Bai, "Pybullet, A python module for physics simulation for games, robotics and machine learning (2016–2021)," [Online]. Available: http://pybullet.org

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[27] T. Eiter and H. Mannila, "Computing discrete fréchet distance," Tech. Rep. CD-TR 94/64, 1994.