applied sciences

MDPI

Article

Robust Learning from Demonstration Based on GANs and Affine Transformation

Kang An †, Zhiyang Wu †, Qianqian Shangguan, Yaqing Song * and Xiaonong Xu *

The College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai 201418, China; ankang526@foxmail.com (K.A.); 1000512056@smail.shnu.edu.cn (Z.W.); shangguan@shnu.edu.cn (Q.S.)

* Correspondence: yqsong@shnu.edu.cn (Y.S.); seuxxn@shnu.edu.cn (X.X.)

† These authors contributed equally to this work.

Abstract: Collaborative robots face barriers to widespread adoption due to the complexity of programming them to achieve human-like movement. Learning from demonstration (LfD) has emerged as a crucial solution, allowing robots to learn tasks directly from expert demonstrations, offering versatility and an intuitive programming approach. However, many existing LfD methods encounter issues such as convergence failure and lack of generalization ability. In this paper, we propose: (1) a generative adversarial network (GAN)-based model with multilayer perceptron (MLP) architecture, coupled with a novel loss function designed to mitigate convergence issues; (2) an affine transformation-based generalization method aimed at enhancing LfD tasks by improving their generalization performance; (3) a data preprocessing method tailored to facilitate deployment on robotics platforms. We conduct experiments on a UR5 robotic platform tasked with handwritten digit recognition. Our results demonstrate that our proposed method significantly accelerates generation speed, achieving a remarkable processing time of 23 ms, which is five times faster than movement primitives (MPs), while preserving key features from demonstrations. This leads to outstanding convergence and generalization performance.

Keywords: collaborative robots; learning from demonstration; imitation learning; generative adversarial networks

Citation: An, K.; Wu, Z.; Shangguan, Q.; Song, Y.; Xu, X. Robust Learning from Demonstration Based on GANs and Affine Transformation. *Appl. Sci.* **2024**, *14*, 2902. https://doi.org/10.3390/app14072902

Academic Editors: Giovanni Boschetti, Matteo Bottin and Riccardo Minto

Received: 9 March 2024
Revised: 23 March 2024
Accepted: 26 March 2024
Published: 29 March 2024

1. Introduction

Contemporary manufacturing endeavors are heavily focused on enhancing efficiency, with industrial robots garnering significant attention for their potential to revolutionize various industry sectors [1,2]. These robots possess the capability to replace humans in numerous tasks [3], thereby alleviating workers from repetitive, hazardous, or physically taxing responsibilities [4,5]. In collaborative environments, it is imperative for humans to have an intuitive means of programming robots to assist with tasks [6]. Learning from demonstration (LfD) or imitation learning (IL) has emerged as a pivotal technique, allowing robots to replicate tasks based on human demonstrations, thereby facilitating direct manual instruction for industrial robots [7,8]. A fundamental objective of LfD is to enable learned movements to adapt seamlessly to diverse task environments.

LfD is currently categorized into three mainstream algorithms. The first category is behavior cloning (BC), which trains policy networks and other machine learning algorithms to mimic expert behavior. However, BC faces challenges due to the discrepancy between the distribution of states generated by the trained policy and the distribution of states in the training data. This inconsistency can lead to suboptimal actions, especially in unfamiliar or novel scenarios, ultimately deviating from the correct trajectory. In sequential decision-making problems, this discrepancy causes errors to accumulate, resulting in trajectories deviating from the correct path.

Therefore, while BC offers a simple learning process without the need for environment interaction, it suffers from limitations such as single-step decision-making, inability to decide on unfamiliar states, and error accumulation. Expert datasets are also limited, requiring continuous expansion to cover all possible states, which increases costs. Common BC algorithms used in collaborative robotics include dynamic movement primitives (DMP) [9,10], Gaussian mixture regression (GMR) [11,12], stable estimation of dynamical systems (SEDS) [13], and kernelized movement primitives (KMP) [14].

Inverse reinforcement learning (IRL) [15,16] assumes that expert behavior is (approximately) optimal and aims to interpret it by reverse-engineering a reward function. This approach avoids the manual setting of a reward function by recovering it from expert trajectory data, allowing reinforcement learning to extract policies from this reward function. However, IRL demands substantial data to accurately deduce the reward function, making it computationally intensive. Additionally, its iterative nature in reinforcement learning processes can lead to slow training speeds and high time costs. Moreover, IRL may struggle to generalize to unseen states or environments, restricting its practical utility.

Generative adversarial imitation learning (GAIL) [17,18] aims to directly learn the policy from expert behavior without significant computational overheads associated with learning cost functions. By introducing generative adversarial networks (GANs) into IRL, GAIL leverages adversarial training to generate expert data distributions, improving the ability to imitate complex behaviors in large-scale, high-dimensional environments [19,20]. However, GAIL may encounter issues like mode collapse [21] and distributional mismatches between generated and expert data, particularly in complex environments.

As previously mentioned, approaches combining LfD and semantic information demand substantial computing resources that are often impractical for industrial environments. Traditional methods relying on clustering, model fitting, and regression necessitate meticulous tuning of hyperparameters [22], hindering the efficient generation of the target trajectory. Moreover, probability-based methods like the Gaussian mixture model (GMM) struggle with generalization when faced with new task requirements.

To enhance the performance of LfD and simplify operation, particularly for individuals lacking programming expertise, this paper proposes a robust LfD method with the following key attributes:

- Enhanced feature extraction performance with GANs: We harness the power of GANs to efficiently capture the distribution of expert demonstration trajectories. This approach facilitates the integration of information from multiple demonstration trajectories, leading to a more nuanced and comprehensive feature representation.
- Enhanced convergence performance with additional loss functions: We introduce novel loss functions, including DILATE [23] loss and Jerk [24] loss, to augment the learning process of GAN networks. These additional loss functions serve to further drive the convergence of the model and mitigate the occurrence of model collapse, thereby enhancing the stability and robustness of the learning process.
- Superior generalization ability: Our proposed method incorporates a geometric-based LfD generalization algorithm. Through the utilization of affine transformations, this approach adeptly addresses the challenge of trajectory generalization. By dynamically adjusting the trajectory through affine transformations, our method facilitates seamless adaptation to diverse and complex environments, showcasing superior generalization capabilities.

Our article is structured as follows: Section 2 provides background information on our work, focusing primarily on the MP method and GANS. Section 3 elaborates on the data preprocessing method, GAN-based trajectory learning method, and affine transformation-based generalization method. Section 4 presents experiments, where we apply our method to the Lasa handwriting dataset and the UR5 robotic arm under simulation. Finally, our conclusions and directions for future work are outlined.

2. Background

2.1. Movement Primitives

Methods rooted in machine learning, such as reinforcement learning (RL), and trajectory planning techniques like GMR and movement primitives (MPs), play a pivotal role in implementing such tasks. While RL boasts a plethora of applications, its utilization in industrial settings is often hindered by the substantial computational resources it demands. Traditional trajectory planning methods such as GMR, SEDS, and MPs remain predominant in LfD algorithms, particularly given the computational limitations of mobile platform-based industrial robots. MPs, in particular, show promise as an LfD framework for adapting learned movements to diverse task scenarios, owing to their capacity to capture demonstration variability. This variability elucidates how movements can be adjusted to accommodate different task environments. MPs can be succinctly denoted by the following equation:

$$\ddot{y} = \alpha_y(\beta_y(g - y) - \dot{y}) + f, \tag{1}$$

where y is our system state, g is the goal, α and β are gain terms, and f is an additional nonlinear system that represent the force over time to obtain the desired behaviors.

Several variants of MPs have been proposed. Sebastian et al. proposed probabilistic movement primitives (ProMPs) [25] to learn from multiple demonstrations using probabilistic method, Li et al. proposed the ProDMP [26] method using neural network for the learning of DMP, and Xu et al. proposed the EditMP [27] method using GANs for the learning of high covariance demonstrations. These MPs efficiently learn movements from numerous demonstrations and adeptly adapt learned movements to task scenarios featuring varying target positions. Additionally, Huang [14] proposed an algorithm that amalgamates GMM and MPs, enhancing the adaptability of MPs to multiple demonstrations. However, this technique is constrained to a limited range of conditions, as MPs can only address specific skill types and encounter challenges in generating circular trajectories. Consequently, the DMP method necessitates not only demonstrations but also a corresponding skill segmentation model [28].

2.2. Generative Adversarial Networks

GANs [21,29] are a novel way of training neural networks such as convolutional neural networks. In contrast to discriminative models, generative models are unsupervised learning tasks in machine learning that focus on capturing the joint probability distribution from data. Once the probability has been estimated, more data can be sampled from the distribution represented by generative models. GANs, on the other hand, are an abstract alternative to mean square loss that makes the models more general.

As shown in Figure 1, GANs consists of two opposing networks whose main role is to construct a derivable procedure to optimize the following functions [29]:

$$\underset{G}{min}\underset{D}{max}V(D, G), \tag{2}$$

where $V(D, G)$ has the following form:

$$V(D, G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))], \tag{3}$$

where x is a real sample and z is a random noise vector sampled from the latent space. The first term of $V(D, G)$ is the entropy by which data from real distribution p_{data} passes through the discriminator. The discriminator tries to maximize this to 1. The second term is entropy by which data sampled from input p_z passes through the generator, which we call a fake example. The discriminator tries to identify the fakes, which are represented by the $-log()$ term. Thus, overall, the discriminator is trying to maximize the function $V(D, G)$, but the generator is exactly the opposite; i.e., it tries to minimize the function.

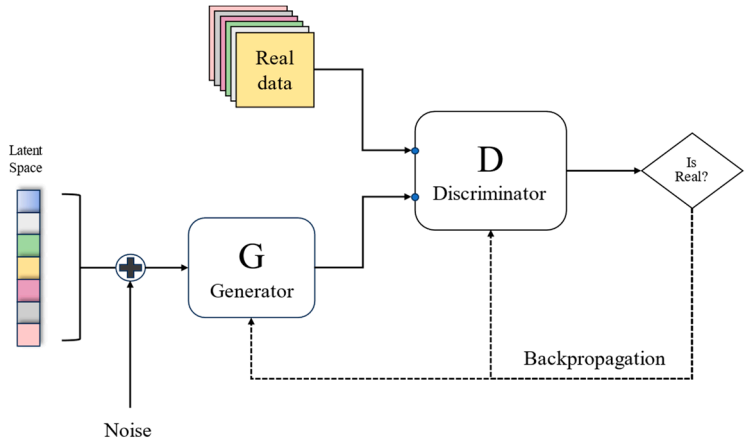

Figure 1. Overview of GANs. GANs take a latent vector as input and generate desired data. These generated data are then compared with real data by the discriminator, guiding their subsequent updates.

3. Proposed Method

3.1. Overview

The organization of the proposed method is illustrated in Figure 2. Initially, raw data is collected from the UR5 robotic arm, followed by the application of the proposed preprocessing method to process the data. Subsequently, we train the proposed GANs using the previously collected data and fine-tune the hyperparameters until the model achieves stability. Finally, the generator component of the GANs is employed for trajectory generation, coupled with affine transformation to represent the task parameters.

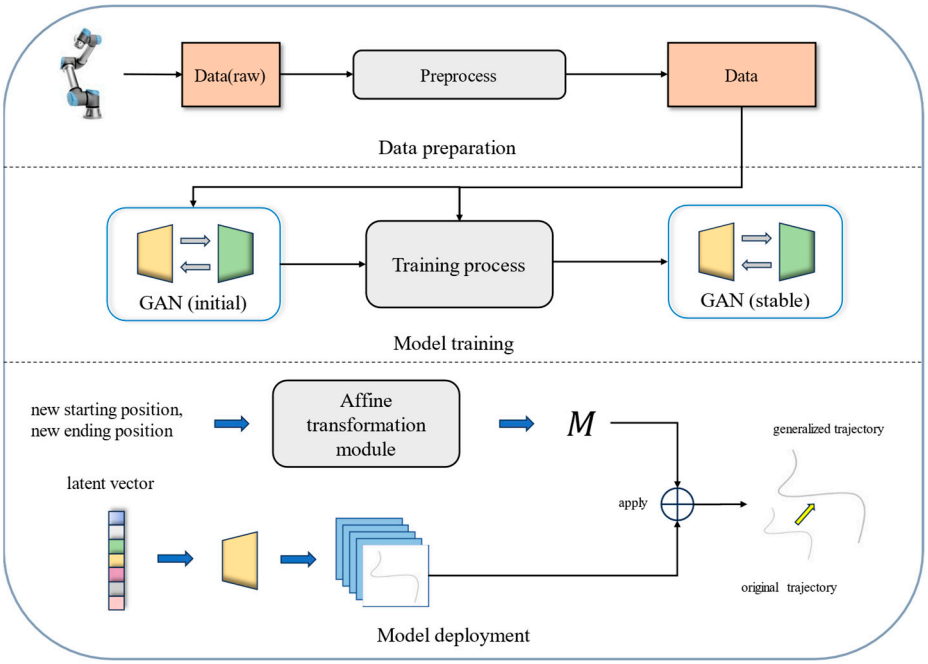

Figure 2. Overview of the proposed method. We outline a data acquisition approach for collecting data, setting the stage for training a GAN-based trajectory learning module. Subsequently, an affine transformation module is applied to ensure optimal generalization performance.

3.2. Preprocess Method

3.2.1. Trajectory Mapping

In this paper, we represent the trajectory using three-dimensional points in the Cartesian coordinate system. However, noise in the sensor's calculation of the end positions may result in outliers in the collected three-dimensional point sequences. To mitigate the

impact of outliers, we employ the MAD (median absolute deviation) method. MAD utilizes two fundamental metrics to filter the data:

$$\begin{aligned} P_{me} &= median(P), \\ P_{mad} &= median(|P - P_{me}|), \end{aligned} \tag{4}$$

where function $median(\ )$ means to find the median of the point set. Through (4) we can obtain P_{me} and P_{mad}, which can help define the upper and lower bound of the algorithm:

$$\begin{aligned} P_{up} &= P_{me} + a \times b \times P_{mad}, \\ P_{down} &= P_{me} - a \times b \times P_{mad}, \end{aligned} \tag{5}$$

where a and b are two parameters that fit the outlier tolerance. We then apply the following filter to the collected data:

$$P = \begin{cases} P & P_{down} < P < P_{up} \\ P_{up} & P > P_{up} \\ P_{down} & P < P_{down} \end{cases}, \tag{6}$$

where P represents a set of 3D points $P = \{p_1, p_2 \ldots, p_n\}$, and each $p_i = (x_i, y_i, z_i)$ is a point in three-dimensional space. After processing, we obtain a set of 3D trajectory points.

In order to decrease data dimensionality, we make the assumption that the three-dimensional trajectory points are situated within a common plane $a^T p = b$, where a is the normal vector of the plane, and b is the distance of the plane from the origin. We can employ singular value decomposition (SVD) to model this plane and project all points onto it. Initially, we construct the data matrix A, where each row represents a data point:

$$A = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \cdots & \cdots & \cdots \\ x_n & y_n & z_n \end{bmatrix}. \tag{7}$$

To simplify computations, the dataset is centered by subtracting the mean from each column:

$$\begin{aligned} \mu &= \begin{bmatrix} \overline{x} & \overline{y} & \overline{z} \end{bmatrix}, \\ \overline{A} &= A - \mu. \end{aligned} \tag{8}$$

SVD is then performed on the centered data matrix $\overline{A}$:

$$\overline{A} = U \Sigma V^T, \tag{9}$$

where U is an $n \times n$ orthogonal matrix, Σ is an $n \times 3$ diagonal matrix with singular values sorted in descending order, V is a 3×3 orthogonal matrix. The normal vector a of the plane is extracted from the right singular vector corresponding to the least singular value:

$$\mathrm{a} = V[:, -1]. \tag{10}$$

Consequently, we can map the trajectory into 2D points, significantly improving the convergence of the model, as we will discuss later.

To project a point p_i onto the fitted plane, the formula for vector projection is utilized:

$$p_i = p_i - (\mathrm{a} \cdot p_i)a. \tag{11}$$

This projection process is repeated for all data points, resulting in their projections onto the fitted plane.

3.2.2. Bézier Curve Representation

After recording the demonstrations, we require a method that not only represents these trajectories but also generates additional trajectories. The Bézier curve is employed to fulfill these requirements. A second-order Bézier curve is defined by three control points P_0, P_1, P_2, where P_0 and P_2 are the start and end points of the curve, and P_1 is the middle control point that influences the shape of the curve. The mathematical expression of the second-order Bézier curve is:

$$B(t) = (1-t)[(1-t)P_0 + tP_1] + t[(1-t)P_1 + tP_2], 0 \leq t \leq 1. \tag{12}$$

By varying the parameter t between 0 and 1, we can generate any points desired between the control points. Consequently, demonstration data can be augmented by generating points with different sampling rates for training the neural network.

Another advantage of representing trajectories using Bézier curves lies in their ability to abstract temporal intricacies. When sampling a new trajectory from the curve, the sequential order of samples accurately mirrors the original trajectory's sequence. This adherence to the correct order serves to circumvent abrupt transitions when the trajectory is executed by a robotic arm.

3.3. Trajectory Learning Based on GANs

GANs can learn trajectory representations from multiple demonstrations, enabling the generator to generate trajectories similar to the demonstrations, but they are highly sensitive to noise present in the trajectories. GANs demonstrate strong fitting capabilities for noise and non-smooth parts of the trajectories, resulting in trajectories with high jerk. Additionally, the discriminator cannot quantitatively describe the magnitude of trajectory errors. When the discriminator loss reaches a certain value, it fails to provide sufficient gradients for updating the generator, potentially leading to model collapse.

To address these issues, this paper introduces a new loss function that combines DILATE [23] loss and jerk measurement, integrating traditional trajectory planning and sequence similarity metrics, building upon the loss function of GANs. Denote the trajectory generated by the generator as x and the ground truth trajectory as y, and the equations for these new loss functions are as follows:

$$\begin{gathered} L_{DILATE}(x,y) = \beta \cdot L_{shape}(x,y) + (1-\beta) \cdot L_{temporal}(x,y), \\ L_{Jerk} = \int_0^t || \dddot{x} ||^2 dt, \end{gathered} \tag{13}$$

where $\dddot{x}$ is the third derivative of trajectory x, β is a hyperparameter used to balance the weights between shape and temporal, and t represents the duration of the motion, under discrete conditions (i.e., the number of trajectory points).

L_{DILATE} consists of both shape loss and temporal loss. The shape loss adopts Soft-DTW [30] to ensure differentiability which is defined as follows:

$$L_{shape} = DTW_\gamma(x,y) = -\gamma log\left(\sum_{A \in A_{n,\mathrm{m}}} exp\left(-\frac{\langle A \,|\Delta(x,y)\rangle}{\gamma}\right)\right), \tag{14}$$

where $A_{n,m}$ is the alignment matrices of x and y, $\langle x|y\rangle$ denotes the inner product operation, $\Delta(x,y)$ is the cost matrix of x and γ where cost function can be replaced with Euclidian distance. Soft-DTW apply soft-minimum algorithm with a hyperparameter γ to replace the minimum function which make the shape loss differentiable.

On the other hand, DILATE models the temporal loss, calculating the temporal error of the trajectory points using the optimal transition matrix A_γ^* obtained from Soft-DTW. The algorithm is defined as follows:

$$L_{temporal}(x,y) = \langle A_\gamma^* | \Omega \rangle, \tag{15}$$

where Ω is the temporal penalty matrix, defined as the second norm of the temporal sequence, and the function is fully differentiable. Based on this, the revised loss function for the generator aspect in GAN is defined as follows:

$$L_{Generator} = \alpha_1 V(D, G) + \alpha_2 L_{DILATE} + \alpha_3 L_{Jerk}, \tag{16}$$

where α_1, α_2, α_3 are the loss coefficients.

The discriminator loss remains unchanged. This model can function as a template library for conditional generation when guided by $L_{Generator}$, and as a demonstration trajectory generation module when guided by $L_{Discriminator}$ to generate additional demonstration trajectories.

The GAN framework described above demonstrates strong learning capabilities, addressing the issue of current LfD algorithms lacking learning capacity. The proposed method enables rapid learning and guaranteed convergence to demonstration trajectories. Compared to current optimal teaching learning methods, it requires no hyperparameter tuning, converges more easily, and can be applied to multidimensional inputs. With the incorporation of the novel loss function, GANs can extract common features from multiple demonstration trajectories, mitigating the impact of noise when trajectories contain noise.

After the training process is complete, we can utilize the generator to produce target tasks using Formula (17):

$$T = G(z), \tag{17}$$

where z denotes the latent vector sampled from the latent space $p_z(z)$, $G(\ )$ represents the generator of GANs, and T signifies the task trajectory learned from demonstrations.

In this paper, we proposed a GAN whose component is an MLP network, as shown in Figure 3. The two perceptrons have a symmetric structure. Taking the generator as an example, it takes one cell as input and expands to 512 cells in the hidden space, with the number of output cells being the same as the trajectory length.

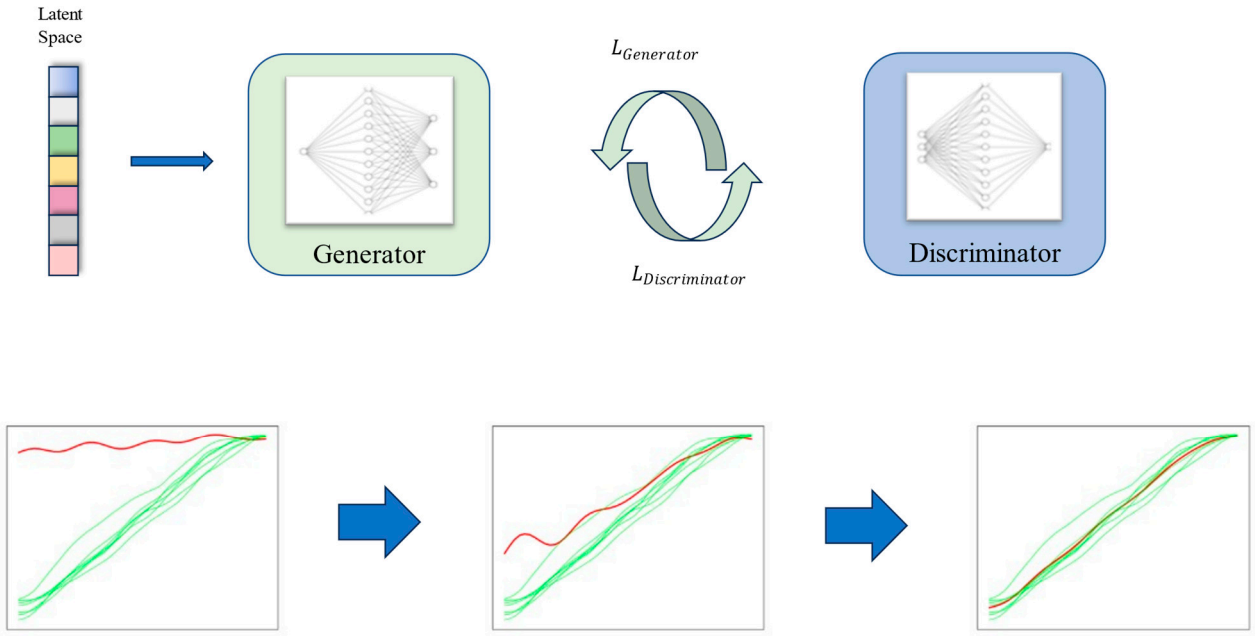

Figure 3. The framework of proposed GANs with novel loss function.

3.4. Generalization Based on Affine Transformation

The generalization performance of demonstration learning primarily pertains to the learning system's ability to adapt to diverse scenario demands, even when encountering new environments characterized by novel starting points, endpoints, intermediate points, and other factors. This adaptability is crucial for effective demonstration learning.

However, traditional trajectory demonstration learning algorithms often struggle with poor generalization performance: ProMP [25] lacks extrapolation ability, and promoted DMP [9] cannot specify intermediate points. To overcome these limitations, this paper introduces a trajectory generalization method based on affine transformation. This method automatically selects trajectory points using perpendicular bisectors, constructing affine transformation solving equations tailored to the generalization requirements of starting and target points, thereby directly obtaining the target trajectory.

An affine transformation involves translation and linear mapping and serves as a deterministic approach that avoids convergence issues. Unlike DMP, in which learns trajectories based on external force terms, the proposed affine transformation method preserves the original attributes of the trajectory. While DMP algorithms struggle to control the trajectory's shape when modifying the endpoint, our method achieves the desired target trajectory by adjusting the scale factor. Additionally, our affine transformation-based approach allows for specifying intermediate point poses without encountering convergence issues.

The subsequent section delineates the algorithm for two-dimensional trajectory imitation learning. We denote the trajectory generated by the generator as $T_{origin} = \{p_1, p_2, \ldots, p_N\}$. For clarity, we denote original starting position of T_{origin} as a and the original ending position as b. In a generalization task, the new starting position and ending position can be denoted as a' and b' (new starting position and ending position can be assigned by user directly), and the generalized trajectory can be denoted as T_{new}. Our proposed method aims to find an affine transformation matrix M to satisfy the equation:

$$T_{new} = M \times T_{origin}, \tag{18}$$

where M is a 3×3 homogeneous matrix which has 6 degrees of freedom. To determine M, 6 constraints are required, corresponding to 3 pairs of trajectory points. In order to find the third corresponding pair, we utilize the following method.

The equation of the line passing through points a and b, as well as the equation of the perpendicular bisector of the line segment joining a and b, denoted as L_{origin} and PB_{origin}, respectively, can be obtained from the following expressions:

$$\begin{aligned} L_{origin} &\triangleq y = kx + \tau, \\ PB_{origin} &\triangleq y = -\tfrac{1}{k} \times \left(x - \tfrac{\mathrm{a}_x + b_x}{2}\right) + \tfrac{\mathrm{a}_y + \mathrm{b}_y}{2}. \end{aligned} \tag{19}$$

where $\triangleq$ represents 'is defined as'; k is the slope of L_{origin}, given by $k = \frac{a_y - b_y}{a_x - b_x}$; τ is the y-intercept, which can be determined by substituting the coordinates of either point a or b into the equation; and $(\,)_x$ and $(\,)_y$ represent the x-coordinate and y-coordinate of the point, respectively. For discrete trajectory points, the approximate intersection point p_{mid} of the trajectory point p_i and the line is obtained by minimizing the distance between the trajectory point p_i and the line:

$$\mathrm{c} = \underset{p_i, p_i \in T_{origin}}{\operatorname{argmin}} \; dist\left(p_i, \mathrm{PB}_{origin}\right), \tag{20}$$

where $dist(\,)$ is used to calculate the distance between a point and a line. After changing the starting position and ending position of the trajectory, the two new lines L_{new} and PB_{new} can be obtained using Formula (19). Point c' corresponding to point c can be determined by Formula (21):

$$\begin{aligned} dist(c', \mathrm{L}_{new}) &= dist\left(c, L_{origin}\right), \\ dist(c', \mathrm{PBn}ew) &= dist\left(c, PB_{origin}\right), \\ \begin{vmatrix} 1 & a_x & a_y \\ 1 & b_x & b_y \\ 1 & c_x & c_y \end{vmatrix} &\times \begin{vmatrix} 1 & a'_x & a'_y \\ 1 & b'_x & b'_y \\ 1 & c'_x & c'_y \end{vmatrix} > 0. \end{aligned} \tag{21}$$

After the unique c' has been determined, the affine matrix M can be solved by the following linear equations:

$$\begin{bmatrix} a_x & a_y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_x & a_y & 1 \\ b_x & b_y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & b_x & b_y & 1 \\ c_x & c_y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_x & c_y & 1 \end{bmatrix} \times \begin{bmatrix} M_{00} \\ M_{01} \\ M_{02} \\ M_{10} \\ M_{11} \\ M_{12} \end{bmatrix} = \begin{bmatrix} a'_x \\ a'_y \\ b'_x \\ b'_y \\ c'_x \\ c'_y \end{bmatrix}, \tag{22}$$

where $(\,)_{ij}$ represents the element in the i-th row and j-th column of the affine matrix M. Finally, the generalized trajectory can be obtained from Formula (18). The process of the algorithm is illustrated in Figure 4. The overall complexity of the algorithm is $O(n)$, making it conducive to rapid trajectory generalization and parameterized trajectory shape transformation. In instances where specific intermediate points are designated, the affine transformation point pair relationship can be recalculated as needed. In comparison to KMP, which has a complexity of $O(n^3)$, our method boasts faster processing speed.

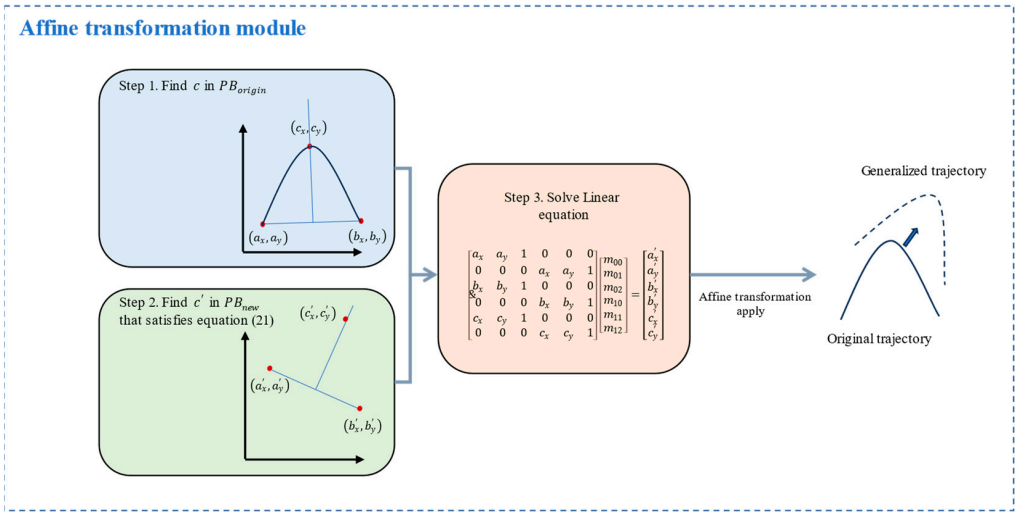

Figure 4. The framework of proposed affine transformation module.

4. Experiment

4.1. Experiment Setting

As shown in Figure 4, in the experiment, a UR5 robotic arm was employed to collect 3D trajectory data. For this study, all data were obtained through simulation. Following preprocessing of the collected data, several schematic trajectories were derived and stored using Bézier curves. Specifically, the digit '7' was selected as an example in this paper. The experimental setup and 2D trajectories are illustrated in Figure 5. Importantly, these trajectories exhibit spikes and other factors that may hinder convergence when using traditional methods. To expedite the training process, trajectories were normalized before being fed into the model.

The Lasa handwriting dataset [13] comprises a collection of 2D handwriting motions recorded from a tablet personal computer. For each motion, users were instructed to perform 7 demonstrations of a desired pattern, beginning from various initial positions (albeit relatively close to each other) and concluding at the same final point. The patterns were illustrated below, and we will utilize a portion of the recorded trajectory for method estimation purposes.

We partitioned the dataset into training and testing subsets, utilizing a portion for hyperparameter tuning. Once the hyperparameters were finalized, we train the model on the entire training set and evaluate its performance on the test set. GANs were implemented

and trained using PyTorch [31]. For updating network parameters, we sampled training batches of fixed size 64, with the input being character digits. We employed the Adam optimizer with default settings and a learning rate of 0.0001. Training halted when the discriminator's accuracy converged to 0.5, at which point we deployed the generator to generate the desired trajectory.

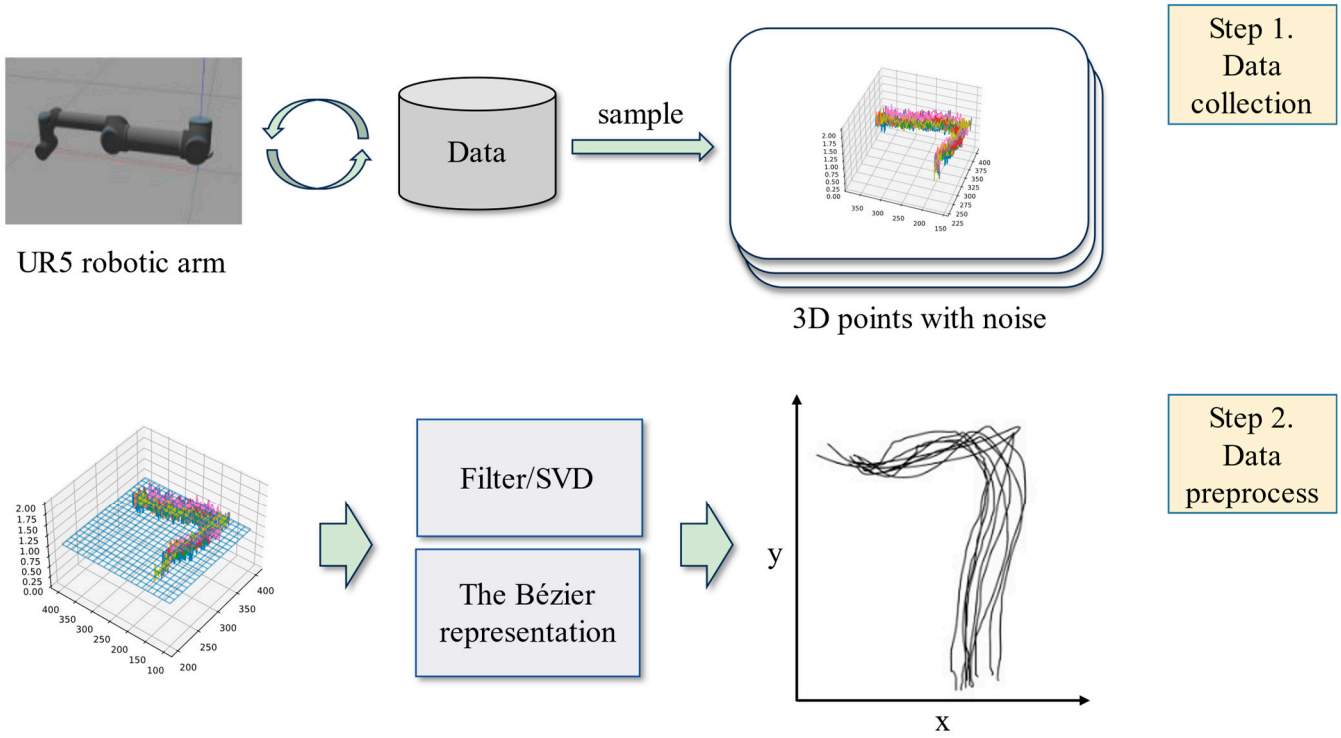

Figure 5. The framework of the proposed preprocessing method. After recording data from the UR5 robot, depicted in different colors in the figure, we employ a MAD filter and SVD method to project the trajectory into two dimensions. Additionally, we introduce a Bézier representation to illustrate the collected data for further utilization.

4.2. Method Deployment

4.2.1. Comparison with Other LfD Methods

After 300 epochs of training, our model achieved numerical stability and can generate the target trajectory with fairly high accuracy with a single numeric character, still taking the digit '7' as an example. Figure 6 shows the generated result of the model on the *x*-axis and *y*-axis. Respectively; we can see that the error is quite small and retains a large number of features in the demonstrations, without excessive smoothing.

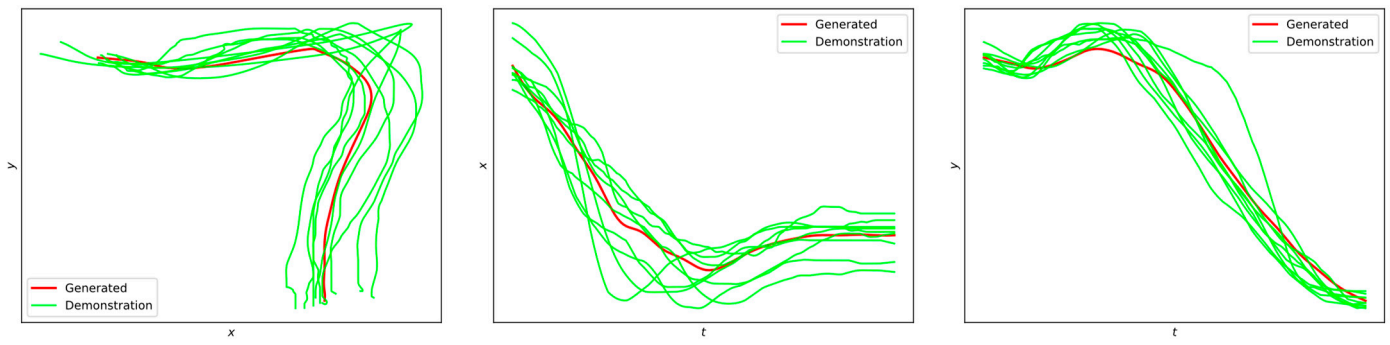

Figure 6. Fitting performance of the proposed network on each dimension.

For a comprehensive comparison, we conducted controlled experiments on the Lasa handwriting dataset by introducing noise and assessing the outcomes against the original data. In these experiments, we employed the parameter *sigma* to regulate the variance of the Gaussian noise added to the trajectory. We utilized mean squared error (MSE) and generation time as quantitative metrics for comparison purposes. The final outcomes are illustrated in Figure 7 and summarized in Table 1. Our proposed method exhibited learning and generation capabilities comparable to existing algorithms while requiring relatively less generation time. Furthermore, the algorithm's learning efficacy under noise was akin to probability-based methods and notably surpassed the performance of the DMP algorithm.

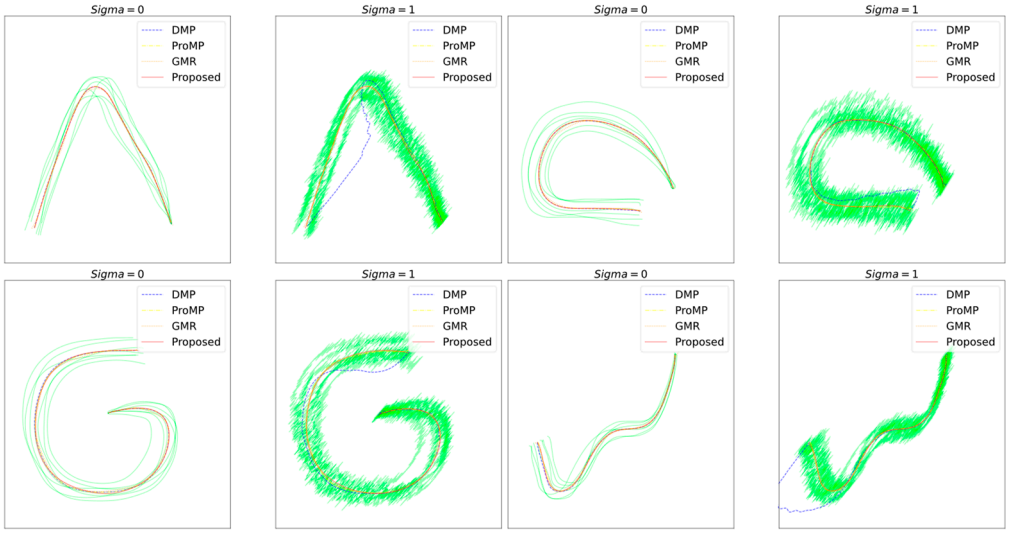

Figure 7. Performance and Learning Reproduction Capability. The green lines in the figure represent the demonstrations.

Table 1. Comparison with other LfD methods in terms of MSE and generation time.

Method	MSE		Time of Generation (ms)
	Common	Noise	
DMP	25.26	53.00	130.38
GMR	27.06	26.75	218.10
ProMP	25.01	**25.16**	116.98
Proposed	**24.71**	25.78	**23.12**

In order to conduct a more comprehensive comparison, we conducted controlled experiments on the Lasa handwriting dataset by introducing noise and comparing the results with the original data. Both mean squared error (MSE) and generation time were used as quantitative metrics for comparison. The final results are presented in the figures and tables. Our proposed method demonstrated learning and generation capabilities comparable to those of existing algorithms while taking relatively less generation time. Moreover, the algorithm's learning ability under noise is similar to that of probability-based methods and significantly outperforms the DMP algorithm.

Probabilistic-based methods such as ProMP, GMR, and KMP are noted for their limited generalization performance [22]. Hence, this section of the experiment primarily aims to compare the performance of DMP with the proposed method. In contrast to Section 4.2.1, the starting and ending points of the generalized trajectory in this section deviate significantly from the demonstration area. To assess the similarity between the trajectories before and after generalization, we introduced the distance correlation analysis (DCA) index [32]. A higher DCA index value indicates a greater similarity in shape. As illustrated in Figure 8 and Table 2, the proposed method exhibited generalization capability superior to DMP, all while incurring a relatively low time cost. Furthermore, due to DMP's reliance on dynamic

systems modeling, the coupling between force terms and endpoint error makes it challenging to control system outputs as the trajectory approaches the endpoint. This often results in significant deviations from the demonstrated trajectory, as depicted in Figure 8b–d, and the average DCA value is relatively low, indicating an inability to maintain the shape after changing the start and end points of the trajectory. In contrast, the proposed method ensured consistent shape preservation, as evidenced by an average DCA value of 1.00.

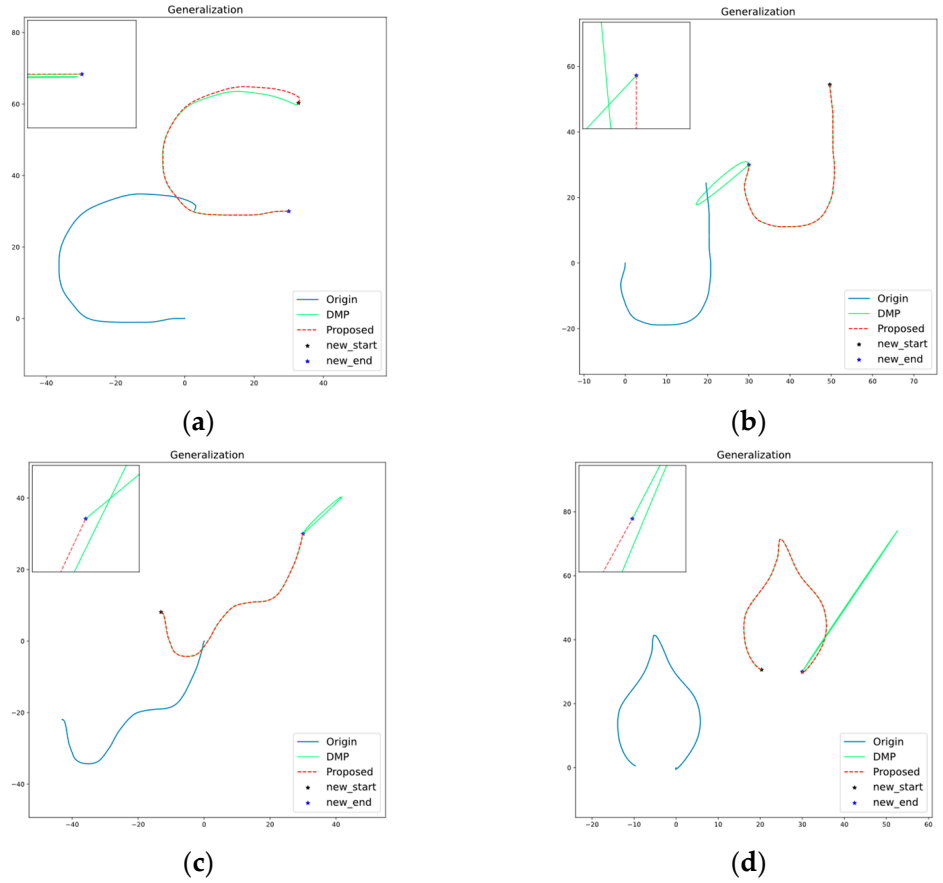

Figure 8. Performance of generalization ability. Demonstrations (**a**–**d**) represent four instances from the Lasa handwriting dataset. The DMP method fails to converge in tasks (**b**–**d**), whereas the proposed method ensures convergence in all tasks.

Table 2. Comparison with DMP in terms of DCA and generation time.

Method	DCA	Time of Generalization (ms)
DMP	0.39	139.28
Proposed	**1.00**	**24.48**

4.2.2. Performance of UR5 Robotic Arm under Simulations

To further validate the reliability of the algorithm proposed in this paper, we employed the previously outlined data collection method to gather trajectory data for the UR5 robot. In the simulated environment, we utilized the UR5 simulation provided by the ROS platform, employing position control mode for precise trajectory control and assessment. This data served as the foundation for training our algorithm. Subsequently, the trajectories generated by our algorithm were implemented within a UR5 simulation environment to evaluate their efficacy. The outcomes of these experiments are presented in Table 3 and Figure 9. Notably, Figure 9 illustrates three distinct handwriting tasks: 'A' in the first row, 'G' in the second row, and 'X' in the third row. As demonstrated in Table 3, our

method exhibited a higher DCA index score, indicating its ability to preserve the shape of demonstrations when generalized to new starting and ending positions.

Table 3. Comparison with DMP in terms of DCA.

Handwriting Task	DCA	
	DMP	Proposed
'A'	0.495	**0.973**
'G'	0.607	**0.986**
'X'	0.326	**0.986**

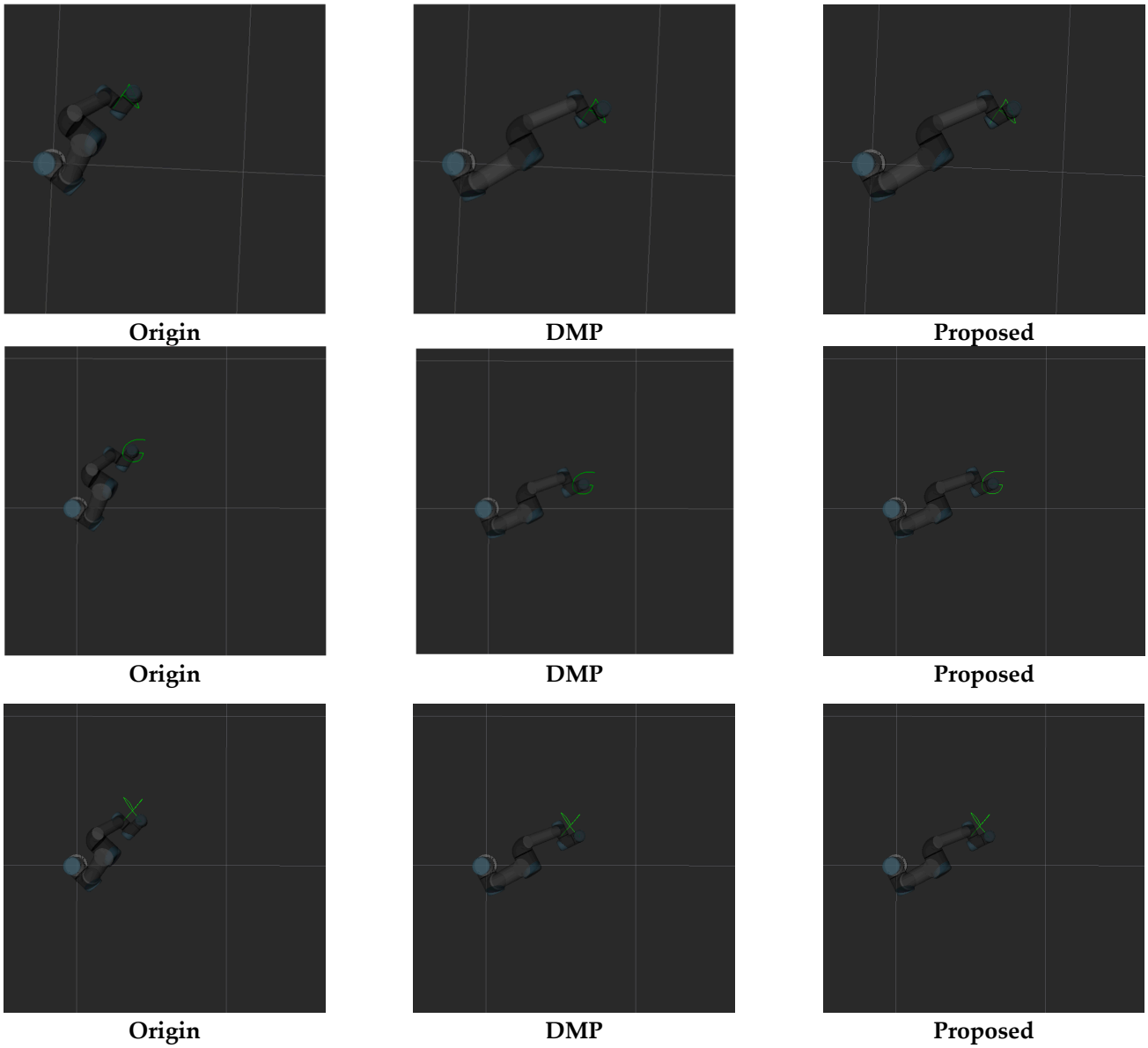

Figure 9. Performance of UR5 robotic arm under simulation.

5. Conclusions

In this paper, we introduce a novel algorithm for trajectory learning and reproduction utilizing GANs coupled with affine transformation. Our method aims to address the challenges in LfD tasks. To facilitate practical implementation, we present a tailored data acquisition and preprocessing method.

By leveraging the capabilities of GANs, our proposed algorithm adeptly learns from demonstration trajectories and generates new trajectories that closely mimic the learned skills. The integration of affine transformation further enhances the algorithm's generalization capabilities and ensures convergence, even in noisy environments. Our data preprocessing method is meticulously designed to enhance the quality of the training data. We employ the MAD filter for noise reduction, followed by the SVD method for dimensionality reduction of the 3D data.

To comprehensively evaluate the performance of our algorithm, we conducted experiments to assess its generation performance, generalization capability, and performance under simulation. Our generation experiment demonstrates the effectiveness of our method in learning the features of demonstrations, even in noisy environments, while achieving a notable fivefold increase in generation speed. Additionally, our generalization experiment confirms the algorithm's ability to maintain trajectory shapes during generalization, with a relatively fast generalization speed. Furthermore, experimental results strongly validate the efficacy of our algorithm in reproducing learned skills. Successful simulations conducted on the UR5 robotic arm further demonstrate its ability to generate accurate trajectories in practical scenarios.

Author Contributions: Conceptualization, Z.W. and K.A.; methodology, Z.W.; coding and realization, Z.W.; validation, Z.W. and K.A.; formal analysis, K.A.; investigation, Z.W.; resources, Z.W.; data curation, Z.W.; writing—original draft preparation, Z.W.; writing—review and editing, K.A.; visualization, K.A., Y.S., X.X. and Q.S.; supervision, K.A., Y.S., X.X. and Q.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Natural Science Foundation of China (Grant No. 62073245).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Robla-Gomez, S.; Becerra, V.M.; Llata, J.R.; Gonzalez-Sarabia, E.; Torre-Ferrero, C.; Perez-Oria, J. Working Together: A Review on Safe Human-Robot Collaboration in Industrial Environments. *IEEE Access* **2017**, *5*, 26754–26773. [CrossRef]
2. Pedersen, M.R.; Nalpantidis, L.; Andersen, R.S.; Schou, C.; Bøgh, S.; Krüger, V.; Madsen, O. Robot Skills for Manufacturing: From Concept to Industrial Deployment. *Robot. Comput. Integr. Manuf.* **2016**, *37*, 282–291. [CrossRef]
3. Gao, Z.; Wanyama, T.; Singh, I.; Gadhrri, A.; Schmidt, R. From Industry 4.0 to Robotics 4.0—A Conceptual Framework for Collaborative and Intelligent Robotic Systems. *Procedia Manuf.* **2020**, *46*, 591–599. [CrossRef]
4. Cherubini, A.; Passama, R.; Crosnier, A.; Lasnier, A.; Fraisse, P. Collaborative Manufacturing with Physical Human-Robot Interaction. *Robot. Comput. Integr. Manuf.* **2016**, *40*, 1–13. [CrossRef]
5. Mohammed, A.; Schmidt, B.; Wang, L. Active Collision Avoidance for Human–Robot Collaboration Driven by Vision Sensors. *Int. J. Comput. Integr. Manuf.* **2017**, *30*, 970–980. [CrossRef]
6. Bauer, A.; Wollherr, D.; Buss, M. Human-Robot Collaboration: A Survey. *Int. J. Humanoid Robot.* **2008**, *5*, 47–66. [CrossRef]
7. Ravichandar, H.; Polydoros, A.S.; Chernova, S.; Billard, A. Recent Advances in Robot Learning from Demonstration. *Annu. Rev. Control Robot. Auton. Syst.* **2020**, *3*, 297–330. [CrossRef]
8. Qu, J.; Zhang, F.; Wang, Y.; Fu, Y. Human-like Coordination Motion Learning for a Redundant Dual-Arm Robot. *Robot. Comput. Integr. Manuf.* **2019**, *57*, 379–390. [CrossRef]
9. Ginesi, M.; Sansonetto, N.; Fiorini, P. Overcoming Some Drawbacks of Dynamic Movement Primitives. *Robot. Auton. Syst.* **2021**, *144*, 103844. [CrossRef]
10. Kong, L.H.; He, W.; Chen, W.S.; Zhang, H.; Wang, Y.N. Dynamic Movement Primitives Based Robot Skills Learning. *Mach. Intell. Res.* **2023**, *20*, 396–407. [CrossRef]
11. Lin, H.I. Design of an Intelligent Robotic Precise Assembly System for Rapid Teaching and Admittance Control. *Robot. Comput. Integr. Manuf.* **2020**, *64*, 101946. [CrossRef]
12. Sung, H.G. Gaussian Mixture Regression and Classification. Doctoral Thesis, Rice University, Houston, TX, USA, 2004.
13. Khansari-Zadeh, S.M.; Billard, A. Learning Stable Nonlinear Dynamical Systems with Gaussian Mixture Models. *IEEE Trans. Robot.* **2011**, *27*, 943–957. [CrossRef]

14. Huang, Y.; Rozo, L.; Silvério, J.; Caldwell, D.G. Kernelized Movement Primitives. *Int. J. Robot. Res.* **2019**, *38*, 833–852. [CrossRef]
15. Ziebart, B.D.; Maas, A.; Bagnell, J.A.; Dey, A.K. Maximum Entropy Inverse Reinforcement Learning. In Proceedings of the 23rd AAAI Conference on Artificial Intelligence, AAAI 2008, Washington, DC, USA, 7–14 February 2008.
16. Peng, X.B.; Kanazawa, A.; Toyer, S.; Abbeel, P.; Levine, S. Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
17. Fei, C.; Wang, B.; Zhuang, Y.; Zhang, Z.; Hao, J.; Zhang, H.; Ji, X.; Liu, W. Triple-GAIL: A Multi-Modal Imitation Learning Framework with Generative Adversarial Nets. In Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, Yokohama, Japan, 11–17 July 2020; Volume 2021-January.
18. Zuo, G.; Chen, K.; Lu, J.; Huang, X. Deterministic Generative Adversarial Imitation Learning. *Neurocomputing* **2020**, *388*, 60–69. [CrossRef]
19. Zhang, T.; Ji, H.; Sil, A. Joint Entity and Event Extraction with Generative Adversarial Imitation Learning. *Data Intell.* **2019**, *1*, 99–120. [CrossRef]
20. Jiang, H.; Yamanoi, Y.; Kuroda, Y.; Chen, P.; Togo, S.; Jiang, Y.; Yokoi, H. Conditional Generative Adversarial Network-Based Finger Position Estimation for Controlling Multi-Degrees-of-Freedom Myoelectric Prosthetic Hands. In Proceedings of the 2022 IEEE International Conference on Cyborg and Bionic Systems, CBS 2022, Wuhan, China, 14–16 March 2023.
21. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; Courville, A. Improved Training of Wasserstein GANs. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 2017-December.
22. Huang, Y.L.; Xu, D.; Tan, M. On Imitation Learning of Robot Movement Trajectories: A Survey. *Zidonghua Xuebao/Acta Autom. Sin.* **2022**, *48*, 315–334.
23. Le Guen, V.; Thome, N. Shape and Time Distortion Loss for Training Deep Time Series Forecasting Models. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Volume 32.
24. Shadmehr, R.; Wise, S.P. A Minimum-Jerk Trajectory. In *Supplementary Documents for "Computational Neurobiology of Reaching and Pointing"*; The MIT Press: Cambridge, MA, USA, 2005; Volume 5.
25. Gomez-Gonzalez, S.; Neumann, G.; Scholkopf, B.; Peters, J. Adaptation and Robust Learning of Probabilistic Movement Primitives. *IEEE Trans. Robot.* **2020**, *36*, 366–379. [CrossRef]
26. Li, G.; Jin, Z.; Volpp, M.; Otto, F.; Lioutikov, R.; Neumann, G. ProDMP: A Unified Perspective on Dynamic and Probabilistic Movement Primitives. *IEEE Robot. Autom. Lett.* **2023**, *8*, 2325–2332. [CrossRef]
27. Xu, X.; You, M.; Zhou, H.; Qian, Z.; Xu, W.; He, B. GAN-Based Editable Movement Primitive from High-Variance Demonstrations. *IEEE Robot. Autom. Lett.* **2023**, *8*, 4593–4600. [CrossRef]
28. Yin, X.; Chen, Q. Trajectory Generation with Spatio-Temporal Templates Learned from Demonstrations. *IEEE Trans. Ind. Electron.* **2017**, *64*, 3442–3451. [CrossRef]
29. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
30. Cuturi, M.; Blondel, M. Soft-DTW: A Differentiable Loss Function for Time-Series. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, Australia, 6–11 August 2017; Volume 2.
31. Ketkar, N.; Moolayil, J. Introduction to PyTorch. In *Deep Learning with Python*; CreateSpace Independent Publishing Platform: North Charleston, SC, USA, 2021.
32. Cowley, B.R.; Semedo, J.D.; Zandvakili, A.; Smith, M.A.; Kohn, A.; Yu, B.M. Distance Covariance Analysis. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Lauderdale, FL, USA, 20–22 April 2017.