A Trajectory Optimisation-Based Incremental Learning Strategy for Learning from Demonstration

Yuqi Wang, Weidong Li and Yuchen Li

1 School of Transportation and Logistics Engineering, Wuhan University of Technology, Wuhan 430063, China; yqwang@whut.edu.cn or y.wang.27@bham.ac.uk
2 School of Mechanical Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China
3 School of Mechanical Engineering, Jiangsu University, Zhenjiang 212013, China; liangyuchen@ujs.edu.cn
4 School of Engineering, University of Birmingham, Birmingham B15 2TT, UK

Abstract: The insufficient generalisation capability of the conventional learning from demonstration (LfD) model necessitates redemonstrations. In addition, retraining the model can overwrite existing knowledge, making it impossible to perform previously acquired skills in new application scenarios. These are not economical and efficient. To address the issues, in this study, a broad learning system (BLS) and probabilistic roadmap (PRM) are integrated with dynamic movement primitive (DMP)-based LfD. Three key innovations are proposed in this paper: (1) segmentation and extended demonstration: a 1D-based topology trajectory segmentation algorithm (1D-SEG) is designed to divide the original demonstration into several segments. Following the segmentation, a Gaussian probabilistic roadmap (G-PRM) is proposed to generate an extended demonstration that retains the geometric features of the original demonstration. (2) DMP modelling and incremental learning updating: BLS-based incremental learning for DMP (Bi-DMP) is performed based on the constructed DMP and extended demonstration. With this incremental learning approach, the DMP is capable of self-updating in response to task demands, preserving previously acquired skills and updating them without training from scratch. (3) Electric vehicle (EV) battery disassembly case study: this study developed a solution suitable for EV battery disassembly and established a decommissioned battery disassembly experimental platform. Unscrewing nuts and battery cell removal are selected to verify the effectiveness of the proposed algorithms based on the battery disassembly experimental platform. In this study, the effectiveness of the algorithms designed in this paper is measured by the success rate and error of the task execution. In the task of unscrewing nuts, the success rate of the classical DMP is 57.14% and the maximum error is 2.760 mm. After the optimisation of 1D-SEG, G-PRM, and Bi-DMP, the success rate of the task is increased to 100% and the maximum error is reduced to 1.477 mm.

Keywords: learning from demonstration (LfD); dynamic movement primitives (DMP); incremental learning; broad learning system (BLS); probabilistic roadmap (PRM)

1. Introduction

Industrial robots have been designed to perform repetitive tasks, so they have been extensively used in the mass production of automotive, aerospace, and marine industries. As the demand for customised production has dramatically increased in recent years, collaborative robots (cobots) have been actively developed to carry out customised tasks within unstructured environments [1]. Learning from demonstration (LfD) is an effective approach to facilitating cobots to fulfil the above purpose. LfD can enable cobots to efficiently realise customised operations based on human demonstrations without the need for detailed motion programming or reprogramming [2,3]. The process of LfD is
illustrated in Figure 1, where a cobot performs a pick-and-place task. To create a demonstration, a human operator drags the end-effector of the cobot to pick up an object from a start point (the yellow box in Figure 1) and place it at a target point (the marker in Figure 1). The demonstration (trajectory) is shown as the red curve in Figure 1. The demonstration is recorded and used to train an LfD-based learning model. To accomplish a similar task where the marker is moved to a new position, the trained LfD-based learning model can adaptively generate a new trajectory (generated trajectory). This is illustrated as the blue dashed line in Figure 1.

Figure 1. LfD-based learning for a pick-and-place application.

Dynamic movement primitive (DMP) is an effective learning model for implementing LfD [4]. Unlike Gaussian mixture modelling (GMM), the DMP model only needs to learn from one demonstration, thereby minimising the difficulty of creating multiple demonstrations [5–7]. The mathematical representation of the DMP model includes a second-order spring damping system and a force item [8]. The former ensures that the generated trajectory created using the DMP model can converge to the target without divergence. The latter, which is constructed based on a series of radial basis function (RBF) kernels, is used to control the shape of the generated trajectory and manage the convergence process. The advantage of the DMP model is its robustness to perturbations [9]. However, the model exhibits some limitations as well. For instance, when the position of the marker illustrated in Figure 1 is moved far away from the previous demonstration area, the generated trajectory via DMP cannot work properly [10]. That is, the difference between the positions of the generated trajectory and the marker should not exceed $\delta$, which is a threshold representing the maximum error allowable for the successful execution of the pick-and-place task. The situation is illustrated by the green dashed curve in Figure 1. To address the issue, the following considerations are given:

- A straightforward idea is to create a new demonstration for the new situation. However, redemonstration is time consuming as the experimental environment needs to be set up again [3]. To mitigate the problem, it will be beneficial to reuse part of the originally created demonstration to adaptively generate a new trajectory (namely, an extended demonstration) for the new target;
- DMP is a one-shot learning model, which means that when the extended demonstration is learned, the knowledge gained from the previous demonstration is forgotten. Therefore, it is imperative to develop an incremental learning mechanism for the DMP model to improve its generalisation capability for various situations.

Various approaches for trajectory generation were developed to complement demonstration data, such as task-parameterised Gaussian mixture models (TP-GMMs) [3], adversarial generative models [11], B-spline-based models [12], and PRM-based models [13]. These approaches can generate new trajectories flexibly. However, the lack of imitation and utilisation of prior knowledge (i.e., existing demonstrations) poses safety risks in LfD applications. In addition, some researchers attempted to improve the performance of
DMP [14]. For instance, neural networks and reinforcement learning [15–17] were used to enhance the learning performance of the force term in the DMP model. The development of the incremental learning mechanisms for the DMP model received attention as well [18–20]. However, the previous research has problems of high complexity and computational difficulty. A more detailed analysis is provided in the following Section 2.

Based on the above considerations, in this study, an improved DMP model with extended demonstration and incremental learning capabilities is designed. The capabilities are implemented based on a 1D-based topology trajectory segmentation algorithm (1D-SEG), a Gaussian probabilistic roadmap (G-PRM), and a broad learning system (BLS). By constructing an appropriate sampling strategy and loss function, G-PRM can generate an extended demonstration based on the features extracted from the segmented original demonstration via 1D-SEG. This extended demonstration and the force item of the DMP model are fed into the BLS to update the DMP model to store previously learned skills (i.e., incremental learning). Three innovative characteristics of this study are represented below.

- Segmentation and extended demonstration: 1D-SEG is combined with G-PRM to generate an extended demonstration by incorporating the features of the original demonstration, so that fewer demonstrations are required to minimise the cost of data collection;
- DMP modelling and incremental learning update: The BLS learns the difference between the extended demonstration and the original demonstration by incrementally increasing the number of network nodes (hereafter referred to as additional enhancement nodes). The force item of the previously constructed DMP model is updated with the results generated by the BLS.
- Electric vehicle (EV) battery disassembly cases and an experimental platform were used to verify the effectiveness of the developed approach. Based on the approach, the successful disassembly of nuts and battery cells were achieved.

The remainder of this paper is structured as follows. Section 2 reviews the related work from the perspective of the trajectory generation and DMP-based optimisation. The key research gaps of the related work are summarised in Section 2. According to these gaps, segmentation and extended demonstration-based methods (namely 1D-SEG and G-PRM) and BLS-based incremental learning for DMP (namely Bi-DMP) are developed in Section 3. In Section 4, a disassembly experimentation platform based on EV batteries and collaborative robots is presented. The effectiveness of the proposed algorithms was verified by three cases, i.e., pick-and-place, unscrewing nuts, and battery cell removal. A comprehensive discussion is given in Section 5. Finally, Section 6 concludes the research.

2. Review of Related Work

2.1. Trajectory Generation

Trajectory generation is an important function to support robotic applications. Zhu et al. [3] developed an LfD-enabled trajectory generation algorithm that combines Gaussian noise with a task-parameterised Gaussian mixture model (TP-GMM). This algorithm facilitates the creation of the generation trajectory by augmenting the original demonstration. Hu et al. [21] proposed a method based on multi-region cost learning. By dividing a demonstration (trajectory) into various sub-regions and establishing cost functions for each sub-region, this method can fine-tune and preserve the geometric features of the demonstration more accurately to optimise the generated trajectory. Peng et al. [11] designed a generative adversarial imitation learning (GAIL)-based algorithm to optimise generated trajectories. In this algorithm, reinforcement learning and generative adversarial networks were combined to allow for more diversity in trajectory generation.

In order to achieve the local adjustment capability and meet the precision requirement of trajectory generation, Li et al. [12] proposed a trajectory generation method that combines a B-spline algorithm with a pruning algorithm. The pruning algorithm was used
to create a trajectory, and B-spline was applied to smoothen the trajectory. However, the control points of the B-spline algorithm cannot accurately mimic the features of the created trajectory. To enhance the similarity of the generated trajectory to the original demonstration and enable local adjustment, sampling-based trajectory generation algorithms are highly promising choices, attributed to their simplicity, flexibility, and computational efficiency [13]. Hüppi et al. [13] and Zhou et al. [22] proposed a trajectory generation method based on PRM. PRM can generate trajectories through the Dijkstra algorithm and sampling points. The distribution of sampling points can affect both the algorithm’s efficiency and the quality of the generated trajectory. It is important to design a suitable sampling strategy.

However, several limitations persist when applying these approaches in manufacturing processes: (1) due to the lack of efficient trajectory segmentation methods, the generation of trajectories can only rely on the entire original demonstration, which may not be economical enough when trajectory adjustments are needed for just one sub-region. (2) Mimicking the characteristics of the demonstration proves challenging, and the shape of generated trajectories is hardly controlled because there is a lack of research on efficient sampling strategy and similarity measurement between the original demonstration and extended demonstration.

2.2. DMP-Based Optimisation

DMP has been widely used in LfD. Park et al. [23] and Ijspeert et al. [24] refined the classic DMP to accommodate demonstrations with the same start and target, as well as with obstacle avoidance, which provides a basis for further optimisation. The force item in DMP is constructed based on RBF kernels and weights which play a crucial role in determining the learning effect on demonstration. Consequently, the optimisation of the force item has received substantial attention. Teng et al. [25] integrated the Takagi–Sugeno fuzzy system with DMP, indicating that the fuzzy inference system can estimate the force item. To better learn the non-linear demonstrations, various researchers used neural networks to model the forcing item. For instance, Si et al. [15] adopted radial basis function neural networks (RBFNNs) to model the forcing item in DMP, enabling the learning of the position and orientation of demonstrations. To further improve the quality of DMP-generated trajectories, both Noohian et al. [16] and Kim et al. [17] not only replaced the force item with neural networks but also applied reinforcement learning for model training. The searching mechanisms in RL can improve the quality of trajectories generated by DMP. Moreover, Davchev et al. [6] introduced a residual correction policy aimed at improving the generalisation of DMP in peg-in-hole applications, thereby refining the quality of trajectories generated by DMP. Notably, the aforementioned methods are mainly focused on the improvement of force item and the DMP training process. Due to a lack of incremental learning capacity, when confronted with a new demonstration, the DMP has to be retrained.

To enable incremental learning within DMP, several scholars have made preliminary explorations. Lu et al. [18,19] attempted to combine a BLS with the force item of DMP, aiming to equip DMP with incremental learning capability. The BLS has a flat network structure and rapid update capability [20]. This allows the DMP to be iteratively updated efficiently. However, this approach is complex in force item construction. Simplifying the approach for combining BLS and DMP and reducing the computational complexity are necessary. In conclusion, optimising and enhancing the force item to endow DMP with incremental learning capabilities are crucial in real applications.

Therefore, the key research gaps of the related works can be summarised as follows:

- Mimicking the geometric features of the original demonstration proves challenging. There is a lack of research in designing an efficient sampling strategy and similarity measurement to avoid sharp turns in generated trajectories and enhance their resemblance to the demonstrations.
These approaches neglect integrating the incremental learning function within DMP. When presented with a new demonstration, the DMP model has to be retrained. Meanwhile, there is a shortage of industrial applications leveraging incremental learning.

3. Research Methodology

The methodology in this research consists of the following steps: (i) segmentation and extended demonstration; (ii) DMP modelling and incremental learning update. The framework of this section is shown in Figure 2. Details are given below.

![Figure 2. The framework of the research methodology.](image)

Before introducing the methodology of this paper, it is necessary to state the superiority of the designed method. Traditional trajectory segmentation methods mainly include subjective decision making based on trajectory geometric features [26,27] and adaptive segmentation based on neural networks [28]. In the optimised 1D-SEG developed in this paper, both the geometric features of the trajectory and a rigorous mathematical process are considered. This can make the segmentation results more objective. Compared to other trajectory generation algorithms like rapidly exploring random tree (RRT) [29], the advantage of PRM lies in the ease of optimising sampling points, as it constructs a roadmap to facilitate finding better trajectories in complex environments. In addition, the flat structure and rapid update strategy of the BLS do not require extensive training time, making it suitable for quickly building models.

3.1. Segmentation of Demonstration

Curvature serves as a quantitative measure of a trajectory’s features and, inspired by [30], a curvature-based trajectory segmentation algorithm has been designed. The details are as follows:

As illustrated in Figure 3, the blue dashed curve represents the curvatures of a trajectory. According to the changing trend of the curvatures, the trajectory can be divided into sub-regions represented as $P$. $M_a_i$ and $M_i$ denote the local maximum and local minimum curvatures in the $i$th region. $M_{a_0}$ and $M_{i_3}$ are the global maximum and global minimum curvature for the entire trajectory. The Euclidean distance ($E_{u\delta}$) between $M_{a_i}$ and $M_i$ in the $i$th region is defined as the curvature distance of the $i$th region $\|P\|$. To optimise the segmentation of the trajectory, a threshold $\epsilon$ is defined. If $\|P\| < \epsilon$, the $i$th region and its $M_{a_i}$ and $M_i$ will be eliminated in the segmentation process. The rest of the local maximum and local minimum curvatures are selected as the segmentation points.
Figure 3. A trajectory with curvature regions.

3.2. Extended Demonstration via G-PRM

An extended demonstration will be created for the new target. The extended demonstration not only retains the features of the original demonstration but also reduces the computational complexity compared to traditional neural network algorithms. The process is detailed below.

After segmentation, the original demonstration $y_d$ is divided into several sub-demonstrations (specifically, sub-1 and sub-2 as illustrated in Figure 4b). Subsequently, the $\text{Eudis}$ from the segmentation point and the target point to the new target point are calculated as $||d_1||$ and $||d_2||$, respectively, as shown in Figure 4b. Since $||d_2|| < ||d_1||$, sub-2 is closer to the new target point. Consequently, sub-2 is selected as the trajectory to be imitated ($\text{Tra}_{\text{ini}}$) for generating the extended demonstration ($\text{Tra}_{\text{gen}}$), as shown in Figure 4b,c.

Figure 4. Demonstration segmentation and extended demonstration. (a) Original demonstration. (b) Demonstration segmentation. (c) Trajectory generation.

For generating the $\text{Tra}_{\text{gen}}$, the PRM-based method is a preferred choice [31]. Figure 5 illustrates the fundamental processes of PRM. Initially, PRM involves generating sampling points (represented by blue dots) while avoiding obstacles, as depicted in Figure 5a. Subsequently, each sampling point is connected to its adjacent sampling points to form a roadmap, as illustrated in Figure 5b. Finally, to determine the shortest trajectory from position 1 to position 2, the Dijkstra algorithm is employed to search results in the roadmap. Figure 6 displays the flowchart of the G-PRM, with two red blocks highlighting the improvements. To more accurately express the meaning of the following equations, specific letters that represent sets and matrices will be presented in bold.
Figure 5. Schematic diagram of PRM. (a) Sampling. (b) Roadmap construction. (c) Trajectory generation.

3.2.1. Generation of Sampling Points

According to Figure 5, the sampling points in PRM are generated randomly, which makes it difficult to control the shape of the resulting trajectory. By using Gaussian distribution-based sampling points (Gaussian sampling points, $\text{GSP}$), the adjustment of the generated trajectory $\text{Tra}_{\text{gen}}$ can be effectively facilitated.

The initialisation of $\text{GSP}$ relies on Gaussian noise and coordinate differences. Here are the details: a Gaussian distribution is defined as $\mathcal{N}(\mu, \sigma^2)$, where $\mu$ is the mean, and $\sigma^2$ represents the variance. Gaussian noise points directly generated from the $\mathcal{N}(\mu, \sigma^2)$ are defined as $\text{G}_N = \{G_N_1, G_N_2, ..., G_N_{\text{sample}}\}$. The ‘$\sim$’ symbol indicates that each dimension of the element in the $\text{GN}$ follows $\mathcal{N}(\mu, \sigma^2)$, which is expressed as Equation (1).

$$G_N_i \sim \mathcal{N}(\mu, \sigma^2), G_N_{iy} \sim \mathcal{N}(\mu, \sigma^2), G_N_{iz} \sim \mathcal{N}(\mu, \sigma^2)$$

$\text{GN}$ needs further processing to yield $\text{GSP}$, which is distributed between the segmentation point and the new target point. The number of elements in $\text{GSP}$ matches that in $\text{GN}$, hence $\text{GSP} = \{GSP_1, GSP_2, ..., GSP_{\text{sample}}\}$, and $GSP_i = (GSP_{ix}, GSP_{iy}, GSP_{iz})$, with the same $i \in [1, \text{sample}]$. A time series $t = \{t_1, t_2, ..., t_{\text{sample}}\}$ serves as a control variable to calculate the $GSP_i$ from $G_N_i$ based on the coordinate differences between the segmentation point and the new target point. The segmentation point and new target point for $\text{Tra}_{\text{gen}}$ are $\text{seg} = (x_s, y_s, z_s)$ and $\text{n.tar} = (x_e, y_e, z_e)$, respectively. The calculation for $t_i$ in $t$ and $GSP_i$ is described in Equation (2):

$$\begin{align*}
GSP_{ix} &= x_s + t_i \cdot (x_e - x_s) + G_N_{ix} \\
GSP_{iy} &= y_s + t_i \cdot (y_e - y_s) + G_N_{iy}, t_i = \frac{i}{\text{sample}}, i = 0, 1, ..., \text{sample} \\
GSP_{iz} &= z_s + t_i \cdot (z_e - z_s) + G_N_{iz}
\end{align*}$$

Figure 6. Overall flowchart of the G-PRM.
3.2.2. Bias Optimisation

G-PRM can generate an initial \( \text{Tra}_{\text{gen}} \) based on \( \text{GSP} \). To derive bias \( \text{bv} \) for updating \( \text{GSP} \) and \( \text{Tra}_{\text{gen}} \), two key processes are required: (1) it is essential to design a Fréchet distance-based [32] similarity criterion to measure the similarity between \( \text{Tra}_{\text{gen}} \) and \( \text{Tra}_{\text{imi}} \) (\( \text{Fresim}(\text{Tra}_{\text{gen}},\text{Tra}_{\text{imi}}) \)). \( \text{Fresim}(\text{Tra}_{\text{gen}},\text{Tra}_{\text{imi}}) \) is derived from Fréchet distance. (2) Scaling \( \text{Tra}_{\text{gen}} \) and \( \text{Tra}_{\text{imi}} \) to the same scale facilitates the calculation of coordinate differences.

(1) To calculate the Fréchet distance between \( \text{Tra}_{\text{imi}} \) and \( \text{Tra}_{\text{gen}} \), three steps are involved:

Step 1—Initialisation definitions: This paper defines \( \text{Fr}(a,b) \) as the Fréchet distance between the first \( a \) points of \( \text{Tra}_{\text{imi}} \) and the first \( b \) points of \( \text{Tra}_{\text{gen}} \). Both \( \text{Tra}_{\text{imi}} \) and \( \text{Tra}_{\text{gen}} \) contain an equal number of points, with \( a \) and \( b \in (1,k] \). \( k \) represents the number of points on the two trajectories, respectively. Since \( \text{Tra}_{\text{imi}} \) and \( \text{Tra}_{\text{gen}} \) are discrete, the computation of the Fréchet distance of the two trajectories is based on \( \text{Eudis} \).

Step 2—Boundary conditions for Fréchet distance:

\[
\text{Fr}(1,1) = \text{Eudis}(\text{seg}, \text{seg}) \\
\text{Fr}(a,1) = \max \left( \text{Fr}(a-1,1), \text{Eudis}(\text{Tra}_{\text{imi},a}, \text{seg}) \right), a > 1 \\
\text{Fr}(1, b) = \max \left( \text{Fr}(1, b-1), \text{Eudis}(\text{seg}, \text{Tra}_{\text{gen},b}) \right), b > 1
\]

where \( \text{Fr}(1,1) \) in Equation (3) represents the Fréchet distance between the first points (seg) of \( \text{Tra}_{\text{imi}} \) and \( \text{Tra}_{\text{gen}} \), the \( \text{Eudis}(\cdot) \) represents the distance between two elements in the bracket. \( \text{Tra}_{\text{imi},a} \) and \( \text{Tra}_{\text{gen},b} \) represent the \( a \)th point and the \( b \)th point in \( \text{Tra}_{\text{imi}} \) and \( \text{Tra}_{\text{gen}} \), respectively. The \( \max (\cdot) \) the largest element in the brackets.

Step 3—Recursive calculation for Fréchet distance: Based on Equation (3), the Fréchet distance between \( \text{Tra}_{\text{imi}} \) and \( \text{Tra}_{\text{gen}} \) can be calculated as shown in Equation (4):

\[
\text{Fr}(a, b) = \max \left( \min(\text{Fr}(a-1, b), \text{Fr}(a, b-1), \text{Fr}(a-1, b-1)), \text{Eudis}(\text{Tra}_{\text{imi},a}, \text{Tra}_{\text{gen},b}) \right), a > 1 \text{ and } b > 1
\]

where \( \min (\cdot) \) opts for the smallest value in brackets. When \( a = k \) and \( b = k \), the Fréchet distance \( \text{Fr}(\text{Tra}_{\text{imi}}, \text{Tra}_{\text{gen}}) \) between \( \text{Tra}_{\text{imi}} \) and \( \text{Tra}_{\text{gen}} \) is obtained.

Based on the calculation results from Equation (4), \( \text{Fresim}(\text{Tra}_{\text{gen}}, \text{Tra}_{\text{imi}}) \) is calculated as shown in Equation (5):

\[
\text{Fresim}(\text{Tra}_{\text{gen}}, \text{Tra}_{\text{imi}}) = \max \left( 1 - \frac{\text{Fr}(\text{Tra}_{\text{gen}}, \text{Tra}_{\text{imi}})}{\sqrt{\text{len}(\text{Tra}_{\text{gen}}) \times \text{len}(\text{Tra}_{\text{imi}})}}, 0 \right)
\]

where \( \sqrt{\text{len}(\text{Tra}_{\text{gen}}) \times \text{len}(\text{Tra}_{\text{imi}})} \) is the geometric mean of the lengths of the \( \text{Tra}_{\text{imi}} \) and \( \text{Tra}_{\text{gen}} \). \( \text{len}(\cdot) \) represents the length of the trajectory calculated via \( \text{Eudis} \). \( \text{Fresim}(\cdot) \)\( \in [0,1] \). A score of 0 indicates complete dissimilarity between the two trajectories, whereas 1 signifies that the trajectories are identical.

(2) To facilitate bias optimisation, \( \text{Tra}_{\text{gen}} \) needs to be scaled to the same scale as \( \text{Tra}_{\text{imi}} \). The target point of \( \text{Tra}_{\text{imi}} \) is represented by \( \text{tar} = (x_o, y_o, z_o) \), as shown by the red dot in Figure 4. The calculated scaling vector is denoted as \( \text{scalefactor} \), as shown in Equation (6):

\[
\text{scalefactor} = \left( \frac{x_e - x_i}{x_o - x_i}, \frac{y_e - y_i}{y_o - y_i}, \frac{z_e - z_i}{z_o - z_i} \right)
\]

\( \text{scalefactor} \) includes scaling factors for \( x \), \( y \), and \( z \) axes, denoted as \( \text{scalefactor}_x \), \( \text{scalefactor}_y \), and \( \text{scalefactor}_z \), respectively. The scaled trajectory of \( \text{Tra}_{\text{gen}} \) is defined as \( \text{Tra}_s \). Each coordinate in \( \text{Tra}_s \) can be calculated by Equation (7):
\[
\begin{align*}
Tra_{s,ix} &= x_s + \text{scale}_{factor,x} \cdot (Tra_{gen,ix} - x_s) \\
Tra_{s,iy} &= y_s + \text{scale}_{factor,y} \cdot (Tra_{gen,iy} - y_s), \quad i = (1,2,...,k) \\
Tra_{s,iz} &= z_s + \text{scale}_{factor,z} \cdot (Tra_{gen,iz} - z_s)
\end{align*}
\]

where \( Tra_{s,ix} \), \( Tra_{s,iy} \), and \( Tra_{s,iz} \) represent the \( x \), \( y \), and \( z \) coordinates of the \( i \)th point in \( Tra_s \). \( Tra_{gen,ix} \), \( Tra_{gen,iy} \), and \( Tra_{gen,iz} \) refer to the \( x \), \( y \), and \( z \) coordinates of the \( i \)th point in \( Tra_{gen} \).

Finally, the bias \( bv = \{bv_1, bv_2, ..., bv_k\} \) for updating \( GSP \) is calculated based on the results of (1) and (2). The initial \( bv \) in each dimension is zero. \( \rho \) is the learning factor, and \( \text{iter} \) represents the current number of iterations. Equation (8) shows the calculation process of \( bv \):

\[
\begin{align*}
bv_{ix} &\leftarrow bv_{ix} + (1 - \text{Fresim}) \cdot \rho^{\text{iter}} \cdot (Tra_{imi,ix} - Tra_{s,ix}) \\
bv_{iy} &\leftarrow bv_{iy} + (1 - \text{Fresim}) \cdot \rho^{\text{iter}} \cdot (Tra_{imi,iy} - Tra_{s,iy}), \quad i = (1,2,...,k) \\
bv_{iz} &\leftarrow bv_{iz} + (1 - \text{Fresim}) \cdot \rho^{\text{iter}} \cdot (Tra_{imi,iz} - Tra_{s,iz})
\end{align*}
\]

In Equation (8), \( bv_{ix} \), \( bv_{iy} \), and \( bv_{iz} \) represent coordinate differences between \(Tra_s\) and \( Tra_{imi}\) in the \( x\), \( y\), and \( z\) dimensions, respectively. \(Tra_{imi,ix}\), \(Tra_{imi,iy}\), and \(Tra_{imi,iz}\) are the \( x\), \( y\), and \( z\) coordinates of the \( i\)th point in \(Tra_{imi}\). To update \( GSP \) based on the \(bv\), interpolation is a necessary method to align the number of elements in \(bv\) with \(GSP\). After interpolation, the \(bv\) is renamed and reformulated as \(ibv = \{ibv_1, ibv_2, ..., ibv_{\text{sample}}\}\). Subsequently, updated \(GSP\) (\(GSP_{\text{new}}\)) is generated through Equation (9), where each coordinate is calculated accordingly:

\[
\begin{align*}
GSP_{\text{new},ix} &= GSP_{ix} + ibv_{ix} \\
GSP_{\text{new},iy} &= GSP_{iy} + ibv_{iy}, \quad i = 0,1,\ldots,\text{sample} \\
GSP_{\text{new},iz} &= GSP_{iz} + ibv_{iz}
\end{align*}
\]

The \(GSP_{\text{new}}\) is used to generate updated \(Tra_{gen}\) via G-PRM. \(GSP\) is regenerated based on Equations (1) and (2) during each iteration. Subsequently, \(\text{Fresim}(\cdot)\) needs to be recalculated based on updated \(Tra_{gen}\) and \(Tra_{imi}\). Additionally, Equations (6)–(9) are applied iteratively to continue updating \(Tra_{gen}\) and \(GSP_{\text{new}}\). Through this process, the similarity between \(Tra_{gen}\) and \(Tra_{imi}\) gradually increases. After reaching the pre-set number of iterations, bias optimisation is completed.

3.3. DMP Modelling and Incremental Learning Updating

3.3.1. Modelling of DMP

In this session, the \(y_d\) is used as an example to model the DMP. This study focuses on discrete DMP represented in the Cartesian space. The Cartesian space representation exhibits advantages over the joint space representation as it allows for trajectory planning without considering the joints and their relative positions [33]. The DMP model is represented below [4,23,24]:

\[
\tau^2 \ddot{y}_d = \alpha_{yd}(\beta_{yd}(g - y_d) - \tau \dot{y}_d) + K f(\theta) - K \dot{\theta}(g - y_{d0})
\]

where \(y_d\) represents the original demonstration; \(\dot{y}_d\) and \(\ddot{y}_d\) denote the first- and second-order derivatives of the demonstration, respectively; \(\alpha_{yd}\) and \(\beta_{yd}\) (\(\alpha_{yd}, \beta_{yd} > 0\)) are damping and stiffness factors (normally \(\alpha_{yd} = 4\beta_{yd}\)); \(\tau\) is the timing parameter used to adjust the timestep of the demonstration; \(y_{d0}\) and \(g\) mark the start and target of the demonstration; \(K\) is a constant that is equal to \(\alpha_{yd}\); \(\dot{\theta}\) is the phase variable computed by the canonical system as expressed using Equation (11). It is used to calculate the value of the force item for each timestep; \(f(\theta)\) is the force item represented in Equation (12); \(\omega_i\) in \(f(\theta)\) is calculated by locally weighted regression (LWR) [24].

\[
\tau \dot{\theta} = -\alpha_\theta \dot{\theta}
\]

(11)
\[
\begin{align*}
    f(\theta) &= \sum_{i=1}^{N} \omega_i \psi_i(\theta) - \theta_i, \\
    \psi_i(\theta) &= \exp(-h_i(\theta - c_i)^2)
\end{align*}
\]  

In Equations (11) and (12), \( \alpha_\theta \) is a constant, and it is always set to one; \( \tau \) is the same as that defined in Equation (10); \( N \) denotes the number of RBF kernels; \( \psi_i(\theta) \) represents the \( i \)th RBF kernel; \( c_i > 0 \) and \( h_i > 0 \) are the centre and width of the \( i \)th RBF kernel.

For computational convenience, Equation (11) is transformed into Equation (13) to calculate \( \theta \):

\[
d\theta = -\tau \alpha_\theta \theta dt \\
\theta = \theta + d\theta
\]  

where \( dt \) is the inverse of the number of the demonstration’s timestep points. When designating a new target \( g_{new} \) to replace \( g \) in Equation (10), each \( y_\dot{} \) in the generated trajectory, denoted as \( y_{gen} \), is calculated by using Equation (10). It is updated with the change in \( \theta \); \( l \) is a timestep within the generated trajectory.

The next coordinate \( y_{gen+1} \) is calculated by using Equation (14):

\[
y_{gen+1} = y_{gen} + \tau y_{gen} dt \\
y_{gen+1} = y_{gen} + \tau y_{gen} dt
\]  

3.3.2. Bi-DMP for Incremental Learning Updating

In this section, the structure and mathematical formulation of Bi-DMP are constructed. The core task of Bi-DMP is to construct the incremental force item for DMP. Inspired by Lu et al. [18,19] and Chen et al. [20], the main structure of the BLS designed for DMP is shown in Figure 7.

In Figure 7, the grey dots symbolise the output of the BLS. The hidden layer of the BLS is represented by blue, red, and orange dots. The blue dots denote mapping features, while the red and orange dots correspond to enhancement nodes and additional enhancement nodes, respectively. Therefore, the incremental force item is the hidden layer combined with the weight corresponding to \( \Delta F_{In} \). Further explanations of the remaining labels in Figure 7 will be provided in the following sections.

This paper postulates that the force item of the \( y_d \) is \( F_{ori} \). \( F_{ori} \) has been computed and parameterised by LWR [24] and is represented in Equation (12). The \( Tra_{gen} \) should combined with sub-1 to form a complete trajectory (namely \( Tra_{gen} \)-based complete demonstration) as shown in Figure 4c, namely \( y_{dn} \). It is used to calculate the original target force item \( F_{target} \). Both \( F_{ori} \) and \( F_{target} \) are sets containing the force item value for each point in demonstrations. Replacing \( y_d \) with \( y_{dn} \) in Equation (10), the initial \( \Delta F_{In} \) can be written as Equation (15):
\[
\Delta F_{\text{In}} = F_{\text{target}} - F_{\text{ori}} \\
= \frac{\tau^2 \gamma_{dn} - \alpha \gamma_{dn} \left( \beta_{\gamma_{dn}} (\gamma_{dn} - y_{dn}) - \gamma_{dn} \right) + \theta (\gamma_{dn} - y_{dn}) - f(\theta)}{K} 
\]

The key components of the hidden layer are \( Z \), \( H \), and \( E \), respectively. \( Z \) refers to the mapping features, \( H \) represents the enhancement nodes, and \( E \) indicates the additional enhancement nodes. This paper assumes that \( u \) is the number of mapping features, \( l \) denotes the number of enhancement nodes, and \( s \) indicates the number of additional enhancement nodes, as depicted from Equation (16) to Equation (18):

\[
Z_i = \delta(\varphi(\theta)WZ_i + \beta_{Z_i}), \quad i = 1, 2, \ldots, u 
\]

\[
H = \xi(\Xi(\varphi)WH_i + \beta_{H_i}) 
\]

\[
E = \xi(\Xi(\varphi)WE_s + \beta_{E_s}) 
\]

\( W_{Z_i} \) is a randomly initialised weight matrix and subsequently fine-tuned via a sparse autoencoder [20]. Both \( W_{H_i} \) and \( W_{E_s} \) are initialised as orthogonal matrixes. The terms \( \beta_{Z_i} \), \( \beta_{H_i} \), and \( \beta_{E_s} \) represent biases, while \( \delta \) and \( \xi \) are activation functions. As suggested in [20], \( \delta \) can be a linear activation function, while \( \xi \) can be the Tansig function, as detailed in Equation (19):

\[
\xi(\cdot) = \frac{2}{1 + e^{-2(\cdot)}} - 1 
\]

To determine the weight \( W^\Delta F_{\text{In}} \) corresponding to \( \Delta F_{\text{In}} \), it is necessary to calculate the basis weight matrix \( W^b \) based on mapping features and enhancement nodes. Equations (20)–(22) represent the computation process:

\[
\Delta F_{\text{In}} = [Z, \ldots, Z_u][\xi(\Xi(\varphi)WH_i + \beta_{H_i})]W^b 
\]

\[
= [Z|H]W^b 
\]

\[
W^b = [Z|H]^{-1}\Delta F_{\text{In}} = A^I\Delta F_{\text{In}} 
\]

\[
A^I = \lim(\lambda I + A\lambda^T)^{-1}A^T 
\]

In Equation (21), \( A \) denotes matrix \([Z|H]\), where \( A^I \) represents the Moore–Penrose pseudo-inverse matrix of \( A \). \( A^T \) is a transpose matrix of \( A \), and \( \lambda \) is the regularisation parameter. \( I \) is an identity matrix. Once the \( W^b \) is obtained, the \( E \) is incorporated into Equation (20) to improve \( W^b \) for higher accuracy in approximating \( \Delta F_{\text{In}} \). Then, Equation (20) is rewritten as Equation (23):

\[
\Delta F_{\text{In}} = [Z|H][\xi(\Xi(\varphi)WE_s + \beta_{E_s})]W^\Delta F_{\text{In}} 
\]

\[
= [A|E]W^\Delta F_{\text{In}} 
\]

\[
= \text{Anew}W^\Delta F_{\text{In}} 
\]

where \( A_{\text{new}} = [A|E] \) (hidden layer matrix). The advantage of incremental learning is that when \( E \) is added, there is no need to recalculate \([A|E]^{-1}\), thereby accelerating the solution process and facilitating network expansion [20]. The \([A|E]^{-1}\) is defined as \( A_{\text{new}}^I \). \( A_{\text{new}}^I \) is calculated by Equation (24):

\[
A_{\text{new}}^I = [A^I - DB^T] 
\]

The temporary matrix \( B \) and \( D \) can be calculated according to Equation (25):

\[
B = \begin{cases} 
C^I, & \text{if } C \neq 0 \\
(I + D^TD)^{-1}D^TA^I, & \text{if } C = 0 
\end{cases}, \quad D = A^IE \]
where $C = E - AD, C^T$ represents the Moore–Penrose pseudo-inverse matrix of $C$. Therefore, the $W^{\Delta F_{\text{in}}}$ can be calculated according to Equation (23) as Equation (25):

$$W^{\Delta F_{\text{in}}} = \mathcal{A}_{\text{new}}^\dagger \Delta F_{\text{in}} = \begin{bmatrix} W^b - DB^T \Delta F_{\text{in}} \\ B^T \Delta F_{\text{in}} \end{bmatrix}$$  \hspace{1cm} (26)

3.3.3. Iterative Update of Force Item under Multi-Demonstrations

Finally, the force item $f(\vartheta)$ can be rewritten as incremental force item $f_{\text{in}}(\vartheta)$. According to Equations (16)–(18), $\vartheta$ in DMP can modify each element within the BLS. Consequently, to maintain consistency in $f_{\text{in}}(\vartheta)$, $\mathcal{A}_{\text{new}}$ is reformulated as $\mathcal{A}_{\text{new}}(\vartheta) = [Z_1, ..., Z_n] \xi \{ \xi(\vartheta) W_{H_1} + \beta H_1 \} \xi \{ \xi(\vartheta) W_{E_4} + \beta E_4 \}$. Equation (27) shows the $f_{\text{in}}(\vartheta)$.

$$f_{\text{in}}(\vartheta) = \frac{\sum_{i=1}^{N} \omega_i \psi_i(\vartheta)}{\sum_{i=1}^{N} \psi_i(\vartheta)} \vartheta_1 + \mathcal{A}_{\text{new}}(\vartheta) W^{\Delta F_{\text{in}}_{\text{b}}}_{\text{in}}$$  \hspace{1cm} (27)

When there are additional demonstrations, $f_{\text{in}}(\vartheta)$ should be revised to $f_{\text{in,p}}(\vartheta)$, where $P$ is the number of demonstrations. Additionally, the $\Delta F_{\text{in}}$ associated with the new demonstration equals the difference between the current $f_{\text{in}}(\vartheta)$ and the new demonstration’s $F_{\text{target}}$. The new difference can be calculated based on Equation (15). Then, the new $W^{\Delta F_{\text{in}}_{\text{b}}}_{\text{in}}$ (namely $W^{\Delta F_{\text{in}}_{\text{b}}}_{\text{in,p}}$, $P = 1, 2, ..., P$) and new $\mathcal{A}_{\text{new}}(\vartheta)$ (namely $\mathcal{A}_{\text{new}}(\vartheta)_{\text{p}}, P = 1, 2, ..., P$) are determined according to Equations (20)–(26) based on the new $\Delta F_{\text{in}}$. Consequently, the $f_{\text{in,p}}(\vartheta)$ can be expressed as Equation (28). Moreover, when $P = 1$, Equation (28) simplifies to Equation (27).

$$f_{\text{in,p}}(\vartheta) = \vartheta(\vartheta) W + \sum_{p=1}^{P} \mathcal{A}_{\text{new}}(\vartheta)_{\text{p}} W^{\Delta F_{\text{in}}}_{\text{p}}_{\text{b}} \hspace{1cm} (p = 1, 2, ..., P)$$  \hspace{1cm} (28)

4. Experiments and Case Studies

In simulations, trajectories from the LASA dataset [34] and authors’ handwriting trajectories were adopted to validate the algorithm.

4.1. Extended Demonstration Based on 1D-SEG and G-PRM

Handwriting A-shaped trajectories and M/S-shaped trajectories from LASA are selected to validate the 1D-SEG and G-PRM. Initially, 1D-SEG is applied to segment the original demonstrations based on their curvatures, as illustrated from Figures 8a, b and 9a. Subsequently, $Tr_{a_{\text{gen}}}$ is calculated according to 3.1, depicted by the purple dashed curves in Figure 9b. Following the algorithm in 3.2, G-PRM generates $Tr_{a_{\text{gen}}}$ (yellow curves in Figure 9b). More specifically, the red and green dots in Figure 9b represent the segmentation and new target points of $Tr_{a_{\text{gen}}}$, respectively. The blue dots are the target points of the original demonstrations.
Figure 8. The results of 1D-SEG on handwriting demonstration trajectories. (a) Curvatures. (b) Curvature filter via 1D-SEG.

Figure 9. The results of G-PRM on handwriting demonstration trajectories. (a) Segmentation points on demonstrations. (b) G-PRM-based extended demonstrations.

4.2. Modelling of DMP

In this subsection, three trajectories are selected to construct DMP. As illustrated in Figure 10, the blue curve trajectories represent demonstrations, whereas the red dashed curve trajectories are the outcomes of DMP. The force item of the DMP will serve as a benchmark for the following incremental learning processes.

Figure 10. Handwriting demonstration trajectories. (a) DMP-based generated A-shaped trajectory. (b) DMP-based generated M-shaped trajectory. (c) DMP-based generated S-shaped trajectory.
4.3. Bi-DMP for Extended Demonstration

Figure 11 intuitively expresses the geometric differences between original and extended demonstrations and the learning performance of Bi-DMP. The red dashed curves in Figure 11 represent the trajectories generated via Bi-DMP.

![Figure 11](image)

Figure 11. The results of Bi-DMP. (a) Bi-DMP for A-shaped trajectory. (b) Bi-DMP for M-shaped trajectory. (c) Bi-DMP for S-shaped trajectory.

4.4. Case Study 1 — Pick-and-Place-Based Bi-DMP

As the most common industrial step, pick-and-place is always used as a benchmark trajectory to validate algorithms. In this part of the study, operators of different ages and genders were selected to perform pick-and-place experiments to collect their demonstration trajectories. Specifically, the operators were asked to use a gripper to catch an object, drag the robot to move, and place the object into an orange box. Figure 12 shows the details:

![Figure 12](image)

Figure 12. Experimental platform for pick-and-place. (a) Main hardware equipment. (b) Demonstration trajectories’ collection.

Figure 12a shows the experimental platform, while Figure 12b presents the demonstrations from operators. These demonstrations were collected via the TM collaborative robot, and the trajectories are shown in Figure 13a. It can be observed that there are discrepancies in the demonstration trajectories when performing the same task, due to factors such as gender, height, and arm span [35,36]. These factors should not be ignored, as operators prefer to work with a robot that matches their habits and physical characteristics in HRC. For example, the trajectory demonstrated by a person of short stature may collide
with the body of a person of tall stature. Therefore, using only one DMP to generalise this pick-and-place task is not appropriate. Bi-DMP, on the other hand, can model and retain the differences in all demonstrations.

In the experiment, the red curve trajectory is chosen as a benchmark demonstration for the construction of DMP, as shown in Figure 13b. The purple dashed curve in Figure 13c is the path generated by the DMP. In order to preserve the discrepancies between different operators, Bi-DMP is used to learn the two new trajectories based on the constructed DMP as shown in Figure 13d. The orange and blue dashed curves are generated by Bi-DMP.

![Figure 13](image.png)

**Figure 13.** Personalised demonstrations for pick-and-place. (a) Demonstration trajectories. (b) Selected trajectory. (c) Construction of a DMP. (d) Bi-DMP results.

According to the above results, Bi-DMP can effectively extend the force item of DMP. Starting from the constructed DMP, the force item is continuously updated so that the differences in the demonstrations can be fully learned.

4.5. Case Study 2 — Unscrewing Nuts and Battery Cell Disassembly

4.5.1. Experiment Platform and Problem Descriptions

Unscrewing nuts and removing battery cells are critical steps in EV battery disassembly applications. The experimental platform is shown in Figure 14a. A multi-purpose adaptor and a vacuum chuck are designed and employed to perform the mentioned tasks, as shown in Figure 14b and Figure 14c, respectively.

Due to the hazardous nature of the EV battery and the constraints of the working space, it is imperative for operators to minimise the duration of the demonstration. This results in nuts closer to the original demonstration being easily disassembled, as illustrated by the red curves in Figure 14d. However, nuts positioned far from the demonstration bring challenges for disassembly due to generalisation issues, as illustrated by the red dashed trajectory in Figure 14d. In the case of battery cell removal, trajectories generated from the Bi-DMP previously used for unscrewing nuts exhibit a deviation due to the lack of specific demonstrations which may cause unforeseen collisions with the working environment. The algorithms proposed in this paper can address these problems.
4.5.2. Design and Use of End-Effectors

Figure 15a,b show the structural design and application of end-effectors for unscrewing nuts and removing battery cells, respectively. The end-effector designed for unscrewing nuts is a multi-purpose adaptor, comprising a series of steel cylinders and a spring system, as depicted in Figure 15a. During the unscrewing process, the steel cylinders adaptively enclose the nuts, allowing for their removal through rotation and friction induced by the multi-purpose adaptor. The end-effector for battery cell removal is a vacuum chuck. It consists of four chucks and pipelines, as shown in Figure 15b. To remove a battery cell, two chucks secure it, as highlighted in the red rectangular box in Figure 15b. Due to safety and payload considerations, only one battery cell should be removed at a time.

4.5.3. Demonstration and Implementation

To ensure the safety of the operator, only the nut at position No. 3 was unscrewed during the demonstration, as shown in Figure 16a. The demonstration adhered strictly to the disassembly rules: moving the multi-purpose adaptor from the start point to the target position (nut), unscrewing it, and then returning to the start place.
The diameter of the multi-purpose adaptor is 25 mm, and the length of the groove is 28.5 mm, as shown in Figure 16c. If the distance between the central area of the nut and the endpoint of the DMP-generated trajectory exceeds the threshold value of 1.75 mm, the unscrewing nuts task is likely to fail due to collisional interference, as highlighted in the red ellipse in Figure 16c. Figure 16b illustrates the generated trajectory via DMP. Positions No. 1 to No. 8 corresponding to the light blue trajectories generated by DMP were able to complete the unscrewing nuts task. In contrast, positions from No. 9 to No. 14 (purple trajectories) failed to complete the task due to excessive errors. Thus, the EV battery is categorised as having a successful area and a failed area. Table 1 shows the error associated with each position.

Table 1. 1. Error between the central area of nut and the target point generated by DMP. 2. Error between the central area of nut and the target point generated by DMP (Continued).

<table>
<thead>
<tr>
<th></th>
<th>Error (mm)</th>
<th>No. 1</th>
<th>No. 2</th>
<th>No. 3</th>
<th>No. 4</th>
<th>No. 5</th>
<th>No. 6</th>
<th>No. 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0.404</td>
<td>0.260</td>
<td>0</td>
<td>0.989</td>
<td>1.023</td>
<td>1.069</td>
<td>1.487</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1.431</td>
<td>1.751</td>
<td>1.760</td>
<td>1.787</td>
<td>2.760</td>
<td>2.743</td>
<td>2.730</td>
</tr>
</tbody>
</table>

In this scenario, 1D-SEG and G-RPM are used to segment the demonstration and generate $Tr_{\text{gen}}$. The outcomes of 1D-SEG are shown in Figure 17a. Typically, the new target point for $Tr_{\text{gen}}$ is selected to be at the centre of the failed nut positions, marked by the brown pentagram in Figure 17b. This position provides better coverage of the working area. The yellow trajectory in Figure 17b shows the $Tr_{\text{gen}}$-based complete demonstration. Subsequently, Bi-DMP is trained based on the force item of previously constructed DMP. The trajectories generated by Bi-DMP are used to re-execute the unscrewing nuts task in the failed area as shown in Figure 17c. According to the error analysis provided in Table 2, all errors are smaller than the threshold. Therefore, the remaining tasks can be executed successfully. Figure 18 shows the implementation of unscrewing nuts.
Figure 17. The results of 1D-SEG, G-PRM, and Bi-DMP on unscrewing nut task. (a) The 1D-SEG for segmenting the original demonstration. (b) G-PRM-generated $T_{\text{tra}}$\textsubscript{gen}. (c) Generated trajectory via Bi-DMP.

Figure 18. Unscrewing nut implementation.

Another significant application is the removal of battery cells. However, due to the height difference between the battery cells and nuts, the shape of the generated trajectories shows deviation, as shown in the purple trajectories in Figure 19a. To address the issue, the Bi-DMP previously used for unscrewing nuts needs to be updated. The 1D-SEG and G-PRM are adopted to generate new $T_{\text{tra}}$\textsubscript{gen} for battery cell removal based on the yellow trajectory as shown in Figure 17b. The red trajectory in Figure 19c presents the new $T_{\text{tra}}$\textsubscript{gen}-based complete demonstration for removing the battery cell, and the new target point for this demonstration is positioned at the geometric centre of B6, as illustrated in Figure 19b. The light blue trajectories are generated by Bi-DMP for battery cell removal.

Figure 20 shows the implementation of the battery cell removal process. In summary, the above case studies address two critical tasks in EV battery disassembly. It can be concluded that the algorithms proposed in this paper can meet the requirements of these applications.
Figure 19. G-PRM and Bi-DMP results for battery cell removal. (a) Generated trajectories via former Bi-DMP. (b) The new target point of \( TR_{gen} \) for battery cell removal. (c) Generated trajectories via new Bi-DMP.

Figure 20. Battery cell removal implementation.

5. Discussion

5.1. Time Complexity Analysis

Time complexity can be used to measure the complexity of an algorithm and serves as a reference when computational resources are limited or real-time performance is required. Compared to trajectory segmentation based on neural networks, the discrete curvature-based method offers a faster computation speed because it does not require the collection of large amounts of data for training. At the same time, the Gaussian distribution, as the most widespread form of data distribution in nature, is used in the roadmap construction process in PRM. The fundamental algorithms involved in this paper are DMP, PRM, and BLS. Gaussian mixture model (GMM), TP-GMM, and RRT have similar functionalities. Analysis of trajectory segmentation algorithms using a CNN as an example [29]. The \( o(\cdot) \) is used to denote the time complexity function. According to the studies on time complexity by Liang et al. [37], Curry et al. [38], and Bianchini et al. [39], the time complexity of the aforementioned algorithms is qualitatively expressed in Table 3.

Table 3. Time complexity analysis for different algorithms.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Time Complexity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-based</td>
<td>( o(N \cdot D \cdot F^2 \cdot C \cdot L) )</td>
<td>( N ): the number of samples, ( D ): the dimension of input data (feature number), ( F ): the size of convolution kernel, ( C ): the number of convolution kernels, ( L ): the number of layers, ( I ): the number of iterations of expectation-maximisation (EM) algorithm, ( K ): the number of Gaussian clusters, ( k ): the number of sampling knots, ( n ): the number of sampling points, ( E ): the number of BLS nodes</td>
</tr>
<tr>
<td>TP-GMM</td>
<td>( o(N \cdot K \cdot D \cdot I) )</td>
<td></td>
</tr>
<tr>
<td>GMM</td>
<td>( o(N \cdot K \cdot D \cdot I) )</td>
<td></td>
</tr>
<tr>
<td>BLS</td>
<td>( o(N \cdot D \cdot E) )</td>
<td></td>
</tr>
<tr>
<td>DMP</td>
<td>( o(N \cdot D) )</td>
<td></td>
</tr>
<tr>
<td>RRT</td>
<td>( o(k \cdot \log (k)) )</td>
<td></td>
</tr>
<tr>
<td>PRM</td>
<td>( o(n \cdot \log (n)) )</td>
<td></td>
</tr>
</tbody>
</table>

The sorting in Table 3 is based on the time complexity of the algorithms from high to low, with a detailed analysis as follows: the convolution operations and the large number of parameters make the time complexity of the CNN the highest, especially in large-scale datasets and deep learning structures. Table 3 lists the time complexity of only a one-layer CNN. In actual computations, the time complexity of all CNN layers in the network must be summed. The time complexity of GMM is influenced by the number of Gaussian
clusters [1], the total number and dimensionality of the samples, and the number of iterations of the EM algorithm. When TP is added, the data dimensionality $D$ will increase, making the complexity of TP-GMM slightly higher than that of GMM with the same parameters. The time complexity of the BLS mainly depends on the number of samples, the data dimensionality, and the number of nodes in the BLS. Although the computation of the BLS relies on large-scale matrix operations, its time complexity is lower than that of a CNN. The time complexity of DMP is mainly determined by the number of samples and data dimensionality. The time complexities of PRM and RRT are related to the number of sampling points/knots. Their time complexities are similar, while PRM has an advantage in trajectory generation due to the construction of the roadmap.

5.2. Trajectory Generation Analysis among Different Algorithms

The most prominent LfD methods are the GMM-based and DMP-based algorithms. Table 4 discusses the differences among the four algorithms: GMM, TP-GMM, DMP, and Bi-DMP, during the execution of the unscrewing nuts task depicted in Figure 16. In model construction, the GMM-based algorithm requires a larger number of demonstrations, whereas DMP requires only one demonstration. Therefore, in terms of cost, DMP is the more optimal choice.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Success Rate</th>
<th>Running Time</th>
<th>Maximum Error (mm)</th>
<th>Difference Range (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM</td>
<td>7.14%</td>
<td>0.97s</td>
<td>11.21</td>
<td>132.15–180.41</td>
</tr>
<tr>
<td>TP-GMM</td>
<td>14.29%</td>
<td>1.36s</td>
<td>105.34</td>
<td>201.96–970.53</td>
</tr>
<tr>
<td>DMP</td>
<td>57.14%</td>
<td>0.16s</td>
<td>2.76</td>
<td>128.65–147.88</td>
</tr>
<tr>
<td>Bi-DMP</td>
<td>100%</td>
<td>0.67s</td>
<td>1.48</td>
<td>133.74–165.79</td>
</tr>
</tbody>
</table>

In the task of unscrewing nuts, there are differences in the performance of GMM and TP-GMM compared with DMP and Bi-DMP. GMM can generalise the distribution of demonstrations and then subsequently generate trajectories via Gaussian mixture regression (GMR) to perform tasks. However, since GMM cannot adapt to dynamic environments, it can only perform the demonstrated tasks. It means that it can only handle unscrewing a nut from a fixed position. The TP-GMM gains generalisation capabilities by adding frames of reference (frames). However, due to the sensitivity of the frames’ orientation and the inherent uncertainty of the probabilistic model, the success rate of TP-GMM is sub-optimal in high-precision applications. In addition, due to generalisation issues, DMP cannot complete all the tasks of unscrewing nuts. In contrast, DMP enhanced with a BLS can accomplish all the tasks, achieving the highest success rate. Due to differences in programming logic and computer hardware configuration, algorithm running times vary. Overall, GMM-based methods have longer running times than DMP-based methods. This is because the K-means algorithm and the EM algorithm used in GMM computations require multiple iterative operations, especially the EM algorithm. As previously mentioned, GMM and TP-GMM are probabilistic models. Thus, their generated trajectories have uncertainties, with maximum errors and difference fluctuations greater than those of DMP. The difference is the sum of the coordinates of the Euclidean distance between the generated trajectory and the demonstration trajectory for the same target. Particularly for TP-GMM, significant deviations can occur (maximum error and difference will be huge) if the frames are incorrectly specified. Therefore, for the issues discussed in this paper, DMP-based methods are more suitable.

Table 5 shows the differences in trajectory generation between the G-PRM and the classic PRM. As shown in the figures within the table, the blue curve represents the imitated trajectory, the purple dashed line is the trajectory generated by the classic PRM, and the red curve is the trajectory generated by G-PRM. The cyan dots are Gaussian distribution sampling points used to construct the roadmap. The blue dot is the starting point,
while the red and purple dots represent the target point and the new target point, respectively. Since the classical PRM lacks imitation capabilities, the generated trajectory is merely a curve connecting the starting point and the new target point, with a similarity to the blue curve of less than 60%. Despite resampling in each iteration, the similarity to the blue curve does not improve. The G-PRM designed in this paper optimises the sampling points to continuously improve the similarity of the generated trajectory (red curve) to the blue trajectory. Its final similarity is 34.04% higher than the trajectory generated by the classical PRM.

Table 5. Quantitative comparison of PRM-based algorithms.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Changes in Similarity (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-PRM</td>
<td>79.61%</td>
</tr>
<tr>
<td>PRM</td>
<td>59.35%</td>
</tr>
</tbody>
</table>

The study has some limitations. The incremental learning process in Bi-DMP currently handles one task at a time, which may limit its efficiency in more complex applications requiring simultaneous task learning. Expanding the BLS structure to support multi-task incremental learning could significantly enhance its utility. In addition, the implementation of the method in more complex industrial applications, such as turbine and engine assembly, should be explored to fully understand its potential and limitations.

6. Conclusions

In summary, this paper proposed a systematic framework aimed at improving the application scope of DMP and BLSs. The research is organised into three main contents: (1) segmentation and extended demonstration: as a series of demonstration pre-processing methods, 1D-SEG and G-PRM are designed to segment the original demonstration and to generate $T_{\text{ran}}$. These methods effectively preserve the geometrical characteristics of the demonstration and simplify the calculation. (2) DMP modelling and incremental learning updating: Bi-DMP is trained based on $T_{\text{ran}}$ along with the force item of constructed DMP. The incremental learning mechanism of Bi-DMP enables the DMP to continually update without training from scratch. (3) EV battery disassembly case study: this study established a decommissioned battery disassembly experimental platform. Un-screwing nuts and battery cell removal have shown that the proposed algorithms effectively support industrial applications.

Author Contributions: Conceptualisation, Y.W. and W.L.; methodology, Y.W.; software, Y.W.; validation, Y.W. and Y.L.; formal analysis, Y.W.; data curation, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, W.L. and Y.L.; visualisation, Y.W.; supervision, W.L.; project administration, W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was sponsored by the National Natural Science Foundation of China (Project No. 51975444), the International Cooperative Project of the Ministry of Science and Technology of China (Project No. G2022013009), the Science and Technology Commission of Shanghai Municipality (Project No. 23010503700), the Engineering and Physical Sciences Research Council, UK (Project No. EP/N018524/1), the China Scholarship Council (CSC No. 202106950049), and China Postdoctoral Science Foundation (Project No. 2023M741426).

Institutional Review Board Statement: Not applicable.
Informed Consent Statement: Not applicable.

Data Availability Statement: Part of the research data from https://cs.stanford.edu/people/khansari/download.html#LearningLyapunovFunctions (accessed on 2 June 2024). If more demonstration data are needed, please email the first author.

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

References


Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.