IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

GAN-Based Editable Movement Primitive from High-Variance Demonstrations

Xuanhui Xu, Mingyu You*, Hongjun Zhou, Zhifeng Qian, Weisheng Xu, and Bin He

***Abstract*—Movement Primitive (MP) is a promising Learning from Demonstration (LfD) framework, which is commonly used to learn movements from human demonstrations and adapt the learned movements to new task scenes. A major goal of MP research is to improve the adaptability of MP to various target positions and obstacles. MPs enable their adaptability by capturing the variability of demonstrations. However, current MPs can only learn from low-variance demonstrations. The low-variance demonstrations include varied target positions but leave various obstacles alone. These MPs can not adapt the learned movements to the task scenes with different obstacles, which limits their adaptability since obstacles are everywhere in daily life. In this paper, we propose a novel transformer and GAN-based Editable Movement Primitive (EditMP), which can learn movements from high-variance demonstrations. These demonstrations include the movements in the task scenes with various target positions and obstacles. After movement learning, EditMP can controllably and interpretably edit the learned movements for new task scenes. Notably, EditMP enables all robot joints rather than the robot end-effector to avoid hitting complex obstacles. The proposed method is evaluated on three tasks and deployed to a real-world robot. We compare EditMP with probabilistic-based MPs and empirically demonstrate the state-of-the-art adaptability of EditMP. Code https://github.com/Xu-Xuanhui/EditMP.**

***Index Terms*—Learning from Demonstration, Movement Primitive, Generative Adversarial Network**

I. Introduction

Learning from Demonstration (LfD) is a commonly used technique in robotics, which can learn movements from human demonstrations [1], [2]. A key goal of LfD is to make the learned movements adaptable to various task scenes [3]. Movement Primitive (MP) is a promising LfD framework for adapting the learned movements to various task scenes since it can capture the demonstration variability [4], [5]. The variability of the demonstrations shows how to adjust the movement to adapt to different task scenes. Multiple MPs have been proposed, such as probabilistic movement primitives (ProMPs) [4], adaptation probabilistic movement primitives (AdaptProMPs) [5], generative adversarial movement primitives (GAMP) [6]. These MPs can efficiently learn movements from dozens of demonstrations and adapt the learned movements to task scenes with different target positions.

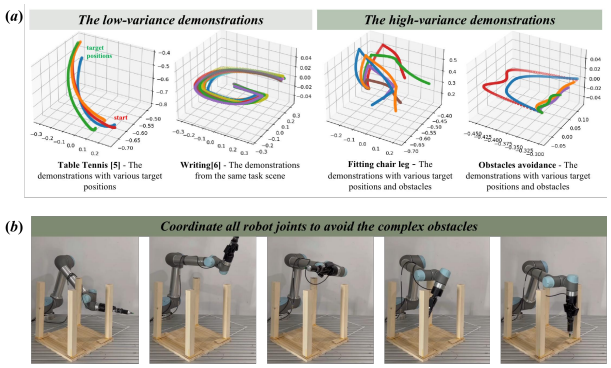

Fig. 1. (a) The low-variance demonstrations collected from the existing works [5] and [6]. The low-variance demonstrations represent the demonstrations with various target positions or the demonstrations from the same task scene. The high-variance demonstrations, which refer to the demonstrations with various target positions and obstacles that have different shapes and positions, belong to our work. (b) It is a fitting chair leg task. The robot needs to attach chair legs to the chair with a screw and electric screwdriver. The chair is a large and complex obstacle. MPs need to coordinate all robot joints to avoid the obstacle.

However, current MPs can only represent the variability of the low-variance demonstrations. As shown in Fig. 1 (a), we show the robot end-effector trajectories of the low-variance and high-variance demonstrations. The low-variance demonstrations collected from the existing works [5] and [6]. The low-variance demonstrations represent the demonstrations with various target positions or the demonstrations from the same task scene. For example, in the table tennis task [5], the expert hit the table tennis ball in different positions (in a specific area), but the hitting trajectories are similar. In the writing task [6], the expert wrote the letter "G" in the same place several times. Each "G" has a slight difference. The high-variance demonstrations are collected from the task scenes with various target positions and obstacles that have different shapes and positions. In the fitting chair leg task, the robot needed to avoid hitting chair legs and fit the leg (any one of four legs) in different positions. Notably, various sizes of chairs lead to varying obstacles and positions for fitting chair legs.

Current MPs, such as ProMPs, can not learn movements from high-variance demonstrations. ProMPs is a probabilistic-based MP approach that employs the Gaussian mixture models (GMM) [7]. GMM enables the ProMPs to capture the variability of human demonstrations with a probability distribution. However, GMM cannot model the distribution of high-variance demonstrations well [8]. Since GMM needs a

Manuscript received: January 5, 2023; Revised: April 1, 2023; Accepted: June 8, 2023. This paper was recommended for publication by Editor Pietro Valdastri upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62073244 and 61825303, NSFC 62088101 Autonomous Intelligent Unmanned Systems, the Science and Technology Commission of Shanghai Municipality (No. 2021SHZDZX0100). X. Xu, M. You, H. Zhou, Z. Qian, W. Xu and B. He are with the College of Electronic and Information Engineering, Frontiers Science Center for Intelligent Autonomous Systems, Tongji University, Shanghai 201800, China. (M. You is the corresponding author with email: myyou@tongji.edu.cn.)

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

large number of components (single Gaussian distribution) to model a complex high-dimensional distribution. Solving the parameters and covariance of these components is difficult and underperforming. AdaptProMPs and GAMP have the same problem since they also employ GMM. As shown in Fig 1 (a), demonstration movements are similar in the table tennis or writing task. Although AdaptProMPs successfully avoided hitting the coffee maker in a coffee preparation task [5], the coffee maker has only changed position, not shape. Therefore, the demonstrations [5] have low-variance variability. It is significant to give robots the ability to avoid hitting obstacles, such as chairs, cups, cabinets, since obstacles are everywhere in real life. To enhance the adaptability of MP, a new framework is needed to represent the high-variance variability of human demonstrations efficiently.

Furthermore, when the obstacle becomes large and complex (The obstacle impedes the movement of the whole robot), MPs need to coordinate all robot joints to avoid hitting obstacles. As shown in Fig. 1 (b), all robot joints need to cooperate to prevent hitting chair legs. GMM represents the cooperation between robot joints during movement by covariance, but it can not represent complex cooperation well.

To address these problems, we present a novel MP approach which is based on transformer [9] and GAN [10], and name it Editable Movement Primitive (EditMP). We design a transformer network that is suitable for sequential robot movement representation and can efficiently learn movements from high-variance demonstrations, which enhances the adaptability of EditMP. Moreover, the transformer enables the EditMP to represent the complex cooperation between robot joints better. After movement learning, EditMP can controllably and interpretably edit movements for various task scenes based on GAN [10] since GAN learns various semantics in some projection directions of the latent space [11]. EditMP generates suitable movements for different task scenes by editing the latent code in projection directions. We test the EditMP with three tasks (The three tasks are drawing rectangle, obstacles avoidance, and fitting chair leg.) in a real-world robot (UR5) to evaluate the adaptability of the EditMP.

The main contributions of this work are as follows:

1) This work presents a novel MP approach that can learn movements from complex and high-variance demonstrations. The proposed MP approach is based on the transformer and GAN and is named Editable Movement Primitive (EditMP).
2) **Controllably:** EditMP can quantitatively adapt the movement to new task scenes.
   Interpretably: EditMP disentangles the latent space and find some projection directions that correspond to the movement variables.
3) We evaluate our EditMP on three tasks and empirically demonstrate the state-of-the-art performance.

II. Related Work

A. Movement primitive

Movement primitive (MP) [4] is one of the LfD technologies typically used to represent the movements. Multiple MP frameworks have been proposed, such as DMP [12], ProMPs [4], [5], GAMP [6]. DMP [13] employs nonlinear systems to learn the mean behavior from demonstrations that have been used successfully for a variety robotic tasks such as grasping [12], locomotion [14], and table tennis [15]. However, DMP only represents the mean movement of the human demonstrations rather than the variability of the demonstrations. It is a big challenge for DMP to generate movements that can adapt to new task scenes without relearning.

Alexandros et al. [4] proposed a probabilistic-based method (ProMPs) that models the variability of demonstrations by the gaussian mixture model [6]. ProMPs is a well-known method since it can generate movements for different task scenes by conditioning operation after learning the movement primitive from demonstrations. However, ProMPs can only learn from low-variance demonstrations, such as demonstrations with different target positions. Many researchers followed ProMPs and presented their method based on ProMPs. Rudolf et al. [16] proposed the movement primitive library. Their approach could segment unlabeled demonstrations into a set of movement primitives and allow for the combination of learned movement primitives to generate new movements. However, their approach can only change the combination order, not movement primitives. Yanlong et al. [8] presented a nonparametric MP method named KMP which has better adaptability than ProMPs. However, KMP controls the end-effector of the robot rather than joints, which makes it easy to hit obstacles in complex scenes. Yu et al. [17] map demonstration trajectories to simple trajectories in Euclidean space, such as straight lines, which can adapt the robot movement to new task scenes by modifying the straight line. However, their approach can only adapt the trajectory of the end-effector instead of robot joints to new task scenes. Sebastian et al. [5] presented the AdaptProMPs, which achieved tasks with multiple target positions. As with ProMPs, their methods can only learn from the low-variance demonstrations, as shown in Fig 1 (a). The performance of these methods is limited by the GMM. To have better adaptability, researchers have proposed a number of new methods, such as the neural probabilistic movement primitive [18], GAMP [6]. Although GAMP introduced GAN into its framework, it was still based on GMM. The demonstrations of GAMP are collected from the same task scene or the task scenes with varied target positions, which are low-variance. These methods cannot represent the high-variance distribution, which significantly limits their adaptability.

B. GAN inversion

To generate specific properties images by editing latent codes [19]–[21], researchers need to find corresponding latent codes of given images, and analyze the relationship between changes in image properties and changes in latent codes. GAN inversion [21]–[23] is commonly used to find the corresponding latent code of the given image.

There are three common categories of GAN inversion: *Optimization-Based Inversion*, *Encoder-Based Inversion*, and *Hybrid Inversion* [21]. *Optimization-Based Inversion* methods [22], [24] find the corresponding latent code by optimiz-

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

ing the $z' = \arg\min_z \|x, G(z)\|$. z' is the desired latent code. x is the given image. G represents the pre-trained generator. *Optimization-Based Inversion* methods can find the near-perfect corresponding latent code for the given image. However, these methods commonly take several minutes for each inversion, so these methods are not suitable for large-scale GAN inversion.

Encoder-Based Inversion methods [23], [25] train a encoder along with the generator. A trained encoder can map any image from the training data distribution to its latent code without fine-tuning. However, *Encoder-Based Inversion* methods are not as effective as *Optimization-Based Inversion* methods. *Hybrid Inversion* methods [26] combine both *Optimization-Based Inversion* and *Encoder-Based Inversion* methods. *Hybrid Inversion* methods attempt to find a better initialization z' to optimize by a trained encoder. These methods can save optimization time while getting the near-perfect corresponding latent code. We employed the *Optimization-Based Inversion* since the time required to generate the trajectory is much less than the time required to generate the image. *Optimization-Based Inversion*'s shortcoming has less impact on our mission.

III. Approach

In this work, we propose a novel Movement Primitive approach that can learn movements from high-variance human demonstrations. The presented MP approach is based on the transformer and GAN and is named Editable Movement Primitive (EditMP).

EditMP consists of movement learning and editing, illustrated in Fig. 2. For movement learning, EditMP learns the movement from human demonstrations based on GAN, which maps latent codes to movements. We illustrate the pipeline in Sec. III-A. In the movement editing, we edit the latent code based on the environment variables of the test scene first. Then, the generator takes the edited latent code as input to generate the movement which can adapt to the test scene. The movement editing is controllable and can generate suitable movements for test scenes with different target positions and obstacles. Notably, the environment variable $VE = \{ve_0, ve_1, ..., ve_n\}$ [1]represents characteristics of the test scene, which can affect the execution of the movement, such as the size of the obstacle, the target position. The details of movement editing are illustrated in Sec. III-B. In addition, movements are represented by robot joint trajectories. For example, a movement $\tau = \{y_t\}^T$ represents joint angles in a T time sequence (T takes a value of 8.). y_t is a 6-dimensional vector representing joints of the 6-DoF UR5.

A. Movement Learning in EditMP

To achieve remarkable adaptability, EditMP must be able to learn movements from high-variance demonstrations and generate various movements. We introduce GAN to EditMP since GAN has the potential to generate various data [10].

[1]For the drawing rectangle task, VEs represent the height, width, and starting position. For the obstacle avoidance task, VEs indicate the size of the obstacles and the target position. For the fitting chair leg task, VEs denote the size and position of the chair leg, as well as the target position.

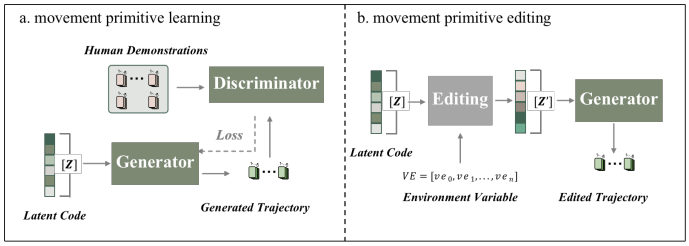

Fig. 2. The framework of the EditMP which learns the movement from the high-variance human demonstrations and edits the learned movement to adapt to test scenes with different target positions and obstacles. Movement learning is based on GAN. EditMP edits the latent codes to generate the movement which can adapt the test scene.

To learn from sequential and high-variance demonstrations, EditMP employs the transformer [9] as the base structure. Moreover, the transformer can represent the complex cooperation between robot joints through self-attention.

Fig. 3 shows the pipeline of the movement learning. EditMP learns the movement from demonstrations based on GAN. The generator samples an 8-dimensional noise vector from the Gaussian noise (latent space) as the input. Then, the noise vector is transformed into a noise matrix of $[8 * 6]$ after the MLP (Multilayer Perceptron) layer. In this work, the time step is fixed ($T = 8$). Hence, we use the one-hot vector as the positional encoding [9]. We concatenate the noise matrix with the position code to get the transformer embedding, that is $[8 * 14]$. Then, the transformer embedding passes through several *Block* layers. In the first step, the normalized transformer embedding passed through the *Self Attention* layer and added to itself. In the second step, the normalized transformer embedding passed through the *mlp* layer and added to itself. Here, the normalize layer is the *BatchNorm* [27]. The *Self Attention* layer which is the most important component of the *Block* layer, consists of matrix multiplication (matmul) and *mlp*. Its main calculation process is $softmax\left(q \times k^T\right) \times v^T$. Finally, a *mlp* layer transforms the output of the several *Block* layers into the movement trajectory τ_G.

As for the discriminator, the transformer embedding consists of the generated trajectory and the positional encoding. The transformer embedding is concatenated with the classify token ([1*14]) after passing through several *Block* layers. Finally, a *mlp* layer transforms the concatenated transformer embedding into a probability that the generated trajectory belongs to the demonstration distribution. The probability is used to train the discriminator and generator.

However, the discriminator only calculates the probability that the generated trajectory belongs to the demonstration. The generator cannot be explicitly told how to increase the likelihood. Moreover, with such an ambiguous loss function, the difference between generated and demonstration trajectories' details are significant. To address this problem, we propose the *Teacher Net* $T(x)$ that can correct the generated trajectory and provide more explicit guidance for the optimization of the generator. The *Teacher Net* needs to be trained. The optimization of the *Teacher Net* is shown in Equ. 1.

$$loss = \|T(\alpha * \tau_H + (1 - \alpha) * noise) - \tau_H\| \quad (1)$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

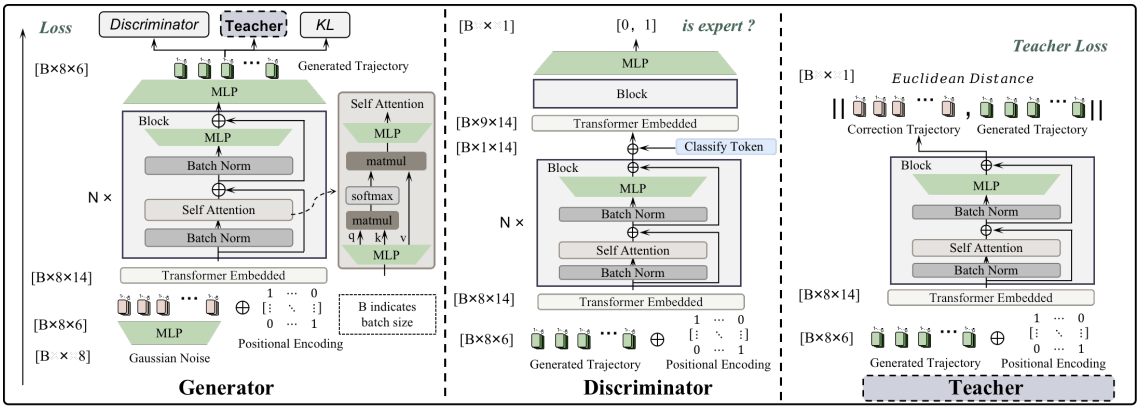

Fig. 3. The framework of movement primitive learning. EditMP is a GAN-based MP framework, which uses the transformer as the base structure. The total loss of the generator consists of three components: Discriminator, Teacher, and KL Divergence.

τ_H are the movement trajectories sampled from the human demonstrations. α is a constant usually taken as 0.95. The noises are the generated trajectories minus their mean. We approximate the generated trajectory with the noise-added demonstration trajectory. When the teacher network can remove the added noise, we consider that it can correct the generated trajectory. After training, the *Teacher Net* can calculate the $loss_{Teacher} = \|T(\tau_G) - \tau_G\|$ for the generator. The *Teacher Net* takes the generated trajectory as input and outputs the correction trajectory τ_C. The Euclidean distances of generated and corrected trajectories are used to optimize the generator.

To improve GAN's robustness and convergence speed, we employ the *KL Divergence* (Kullback-Leibler Divergence) [28] as another loss function. *KL Divergence* is commonly used to measure the distance between two distributions.

$$loss_{KL} = \|D(\tau_G) - D(\tau_H)\|_{KL} \tag{2}$$

Calculating *KL Divergence* using the discriminator's output allows optimizing the generator while preventing the discriminator's output from being polarized. This can stabilize the training process of GAN to prevent failure and accelerate the generator's training. In summary, the total loss of the generator consists of three components: $loss_{Discriminator}$, $loss_{Teacher}$, and $loss_{KL}$.

B. Movement Editing in EditMP

The aim of MPs is to adapt learned movements to new task scenes, which have different target positions, obstacle positions, and obstacle shapes. The movement editing should be controllable and interpretable since the robot has to reach the target position and avoid hitting obstacles (e.g., passing between two obstacles).

The movement editing of EditMP is based on the propositions of GAN. As shown in **Proposition 1**, the semantics of neighboring latent codes tend to be similar [29]. In other words, movements generated by two neighboring latent codes are similar. As for **Proposition 2**, GAN learns various semantics in some linear subspaces of the latent space [11]. Combining these two properties, we can conclude that when the latent code moves along a projection direction in the latent space, the change in its corresponding trajectory is approximately linear. (The linearity here is an approximate linearity.) Fig. 4 is the visualization of **Proposition 1** and **2**. When the latent code varies linearly in a projection direction, the height of its corresponding rectangle undergoes an approximately linear change. Z is the latent space, and τ_i is the robot end-effector trajectory of drawing rectangle task ($\tau_i = G(z_i)$, G is the generator.).

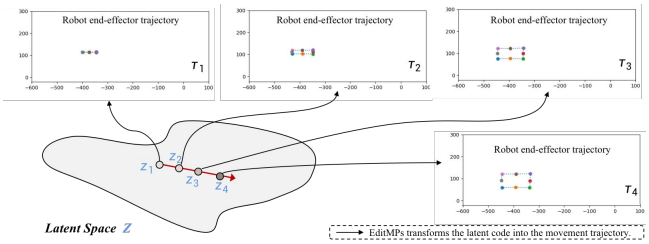

Fig. 4. Visualization of **Proposition 1** and **2**. τ_i is the robot end-effector trajectory of drawing rectangle task, $\tau_i = G(z_i)$. The latent space Z is a multidimensional Gaussian distribution. When the latent code varies linearly in a projection direction, the height of its corresponding rectangle undergoes an approximately linear change.

Proposition 1. *The semantics of neighboring latent codes tend to be similar [10]. C is a small constant.*

$$\begin{aligned} \forall \|z_2 - z_1\| \to 0, \\ \exists \|G(z_2) - G(z_1)\| \leq C \cdot \|z_2 - z_1\| \end{aligned} \tag{3}$$

Proposition 2. *GAN learns various semantics in some linear subspaces of the latent space [11].*

$$\|G(z) - G(z + \triangle z)\| \propto \|\triangle z\| \tag{4}$$

Fig. 5 shows the pipeline of the movement editing. First of all, we have to explore projection directions for movement variables in the latent space. The movement variable $VM = \{vm_0, vm_1, ..., vm_n\}$ describes the characteristics of the movement[2] which corresponds to the environment variable. We disentangle the latent space to find the projection

[2]For the drawing rectangle task, VMs represent the height, width, and starting position of the movement. For the obstacle avoidance task, VMs indicate the size of the avoided obstacles and the end position of the movement. For the fitting chair leg task, VMs denote the size and position of the avoided chair leg, as well as the end position of the movement.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

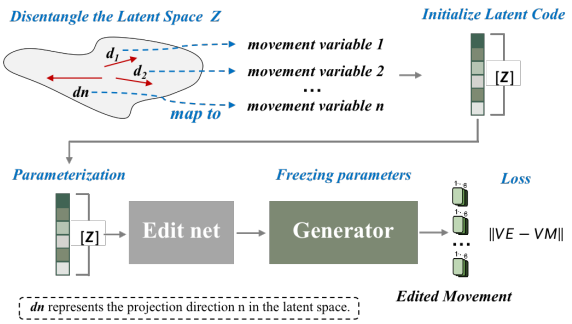

Fig. 5. The pipeline of the movement editing. We parameterize z so that z can be optimized. The parameters of the pre-trained GAN are frozen.

directions, which map to the movement variables. Specifically, we randomly generate 10000 movement trajectories and record their corresponding latent codes [3]. Then, we calculate movement variables for each trajectory. We select two movement trajectories (τ_n, τ_m) with only vm_i different in their movement variables. For example, two rectangular trajectories differ only in width. The direction in the latent space corresponding to the movement variable vm_i is z_d, $z_d = z_n - z_m$ (z_n and z_m are the hidden variables of τ_n and τ_m respectively). To improve the accuracy, we can calculate multiple z_d for the same changing movement variable and then take the average value. It is worth mentioning that when the z_d of two different movement variables are added to a latent code z, two corresponding variables of the generated trajectory ($\tau = G(z + z_d^1 + z_d^2)$) are changed.

After disentangling the latent space, we can start movement editing. We calculate movement variables based on environment variables of task scenes. Then we edit the latent code in the corresponding directions. This process is called initializing latent code. However, when the latent code moves along a direction in the latent space, the change in its corresponding trajectory is not linear. In other words, sampling z at uniform intervals on a certain z_d, the generated trajectories vary unevenly. Therefore, the movement generated by the initialized latent code is imprecise. We need to fine-tune the initialized latent code to achieve controllable movement editing.

To fine-tune latent code, we parameterize the initialized z so that the z can be optimized. We add an Edit net between the z and the generator, which consists of several fully connected layers: $[8\times8]$, $[8\times8]$, $[8\times8]$ (The activation function is leaky_relu). Moreover, the parameters of the pre-trained generator are frozen (We described the training process in Sec III-A). During the fine-tuning, we observe the environment variable $VE = \{ve_0, ve_1, ..., ve_n\}$ of the test task scene, firstly. The VE is used as the label. Edit net transforms the initialized z into z'. Then, the pre-trained generator transforms the z' into the movement. Parameters of the pre-trained generator are frozen, so only the initialized z and the Edit net are optimized. The loss is $\frac{1}{N}\sum_{i=0}^{N}\|ve_i - vm_i\|$. vm_i is the movement variable of the generated movement. When $ve_i - vm_i <= \eta,\ i \subseteq [0, T]$ ($\eta = -5$) and fine-tuning epochs < 1000, we define that the editing is completed[4].

[3]We calculate the trajectory of the end-effector by forward kinematics. The movement variables depend on the task and are based on the trajectory of the end-effector.

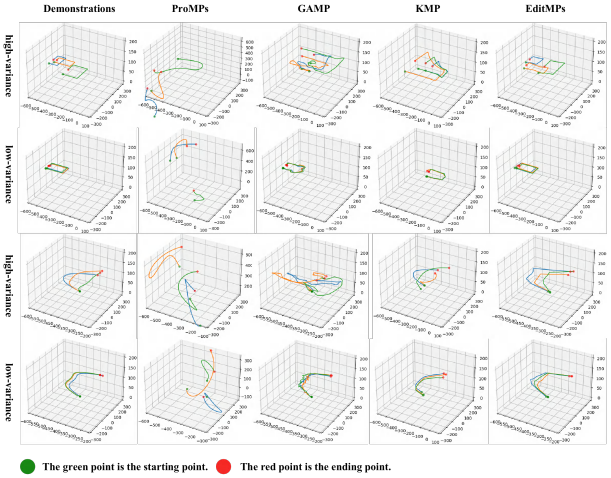

Fig. 6. The top two lines are for the drawing rectangle task. The following two lines are for the obstacle avoidance task. High-variance and low-variance represent high-variance demonstrations and low-variance demonstrations, respectively.

IV. Experiment

In this section, we evaluate the EditMP with various tasks and compare the performance of EditMP and current MPs on these tasks. We demonstrate that EditMP can learn movements from high-variance demonstrations and controllably and interpretably edit learned movements to adapt to various task scenes.

TABLE I
SUCCESS RATES OF DIFFERENT MPS WITH HIGH-VARIANCE AND LOW-VARIANCE DEMONSTRATIONS IN THE OBSTACLE AVOIDANCE TASK

	ProMPs [4]	GAMP [6]	KMP [8]	EditMP
high-variance	5.4% (27/500)	10.6% (53/500)	77.4% (387/500)	97.4% (487/500)
low-variance	5.8% (29/500)	90.6% (453/500)	100.0% (500/500)	100.0% (500/500)

A. Learning from high/low-variance demonstrations

Drawing rectangle and obstacle avoidance are used to evaluate the proposed methods. We compare the performance of EditMP, ProMPs, KMP, and GAMP on the low-variance and high-variance demonstrations. The drawing rectangle task can be considered as a special obstacle avoidance task. The passable trajectory is rectangular. [5] In these tasks, trajectories are in a plane.

Notably, for the EditMP and ProMPs, each step y_t is a 6-dimensional vector representing robot joints. However, the step y_t of the GAMP and KMP is a 2-dimensional vector representing end-effector coordinates (x, y). The time step (T) of the EditMP is 8. The time step of the ProMPs, KMP, and GAMP is 200 since these methods need dense time step.

Fig. 6 shows the results of experiments. As for the drawing rectangle task, when demonstrations have high variance, only

[4]We use an Nvidia 1080 GPU for our work. Training for movement learning takes 12 hours. Movement editing only requires 30 seconds.

[5]The task scene of high-variance demonstrations has different target positions and obstacles that have different shapes and positions. The task scene of low-variance demonstrations has the same obstacle.

EditMP succeeds. Trajectories generated by the ProMPs are curved and do not lie in a plane. Trajectories generated by the GAMP and KMP are more like a rectangle than that of ProMPs. However, GAMP and KMP still fail. EditMP successfully learns the intent of demonstrations to draw various rectangles. Although the GAMP almost succeeds when demonstrations have low variance, trajectories are not smooth enough. KMP succeeds when faced with low-variance demonstrations. Table. I shows the success rate of three methods in the obstacle avoidance task. KMP performs better than ProMPs and GAMP. The two success rates of KMP are 77.4% and 100.0%, respectively. As for the EditMP, the success rates of the two tests are 97.4% and 100.0%, respectively. ProMPs has the worst success rate.

To summarize, EditMP learns the intent of the demonstration movements through the generative adversarial process rather than memorizing the demonstration movements. Moreover, EditMP uses the transformer as its network framework, which is more efficient in modeling complex high-dimensional distributions [9], [10]. GAMP and KMP perform better than ProMPs since their demonstrations consist of the end-effector coordinates, which are low dimension. Therefore, it is easier for them to represent the characteristics of demonstrations. Moreover, current MPs use GMM directly or indirectly in learning movements. GMM is underperforming in facing complex distribution. A large number of components (single Gaussian distribution) are required to model a complex high-dimensional distribution. Solving the parameters and covariance of these components is difficult and underperforming. Moreover, ProMPs has the worst performance since it is challenging to select a large number of basis functions when the input dimension is high [8].

Fig. 7. This figure shows the task scene of the complex task and the process of robot performing the complex task. The robot is required to install the chair leg with a self-tapping screw. During the installation, the robot needs to avoid hitting the chair legs in space. The chair legs to be fitted can be any one of four. The size of the chair can also be varied. *Please see the video in the supporting materials for more experimental demonstrations.*

TABLE II
SUCCESS RATE OF THREE METHODS ON THE TASK OF FITTING CHAIR LEG

	AdaptProMPs [5]	GAMP [6]	KMP [8]	EditMP
success rate	9.4% (47/500)	4.6% (23/500)	4.2% (21/500)	91.6% (458/500)

B. Learning robot joints cooperation

In this section, we employ a complex task to prove that EditMP can represent the complex cooperation between robot joints. As shown in Fig. 7, the complex task is an assembly task. The robot needs to use self-tapping screws to fix the link plate of the chair leg to the seat plate. The electric screwdriver is grasped in the robot's hand. Notably, we use a strong magnetic ring to connect the screw to the screwdriver, as shown in the image named *experiment environment*. When the robot is in the target position, we manually turn on the electric screwdriver.

We have collected 512 demonstration trajectories in the task scenes. We dragged the robot arm to demonstrate the task and recorded the trajectory of all joints at a frequency of 1 Hz. Then we select eight points evenly from the recorded trajectory as the demonstration trajectory. In this task, the chair size of the demonstration and test scene is determined randomly within a certain range. For the chair legs, the size is $40mm \times 40mm \times height, height >= 250mm$. For the seat plate, the size is $200mm <= width <= 400mm, 200mm <= length <= 400mm$.

Table. II shows the success rate of AdaptProMPs, GAMP, KMP, and EditMP in the fitting chair leg task. AdaptProMPs performs better than GAMP and KMP since GAMP and KMP control the end-effector rather than joints, making them more likely to hit obstacles. However, AdaptProMPs' success rate is 9.4%, much lower than EditMP's success rate, is 91.6%. In the fitting chair leg task, not only does the robot's end-effector need to avoid hitting obstacles, but all joints of the robot need to avoid hitting obstacles. Current MPs fail since they cannot represent the cooperation between robot joints well. EditMP performs well but still fails in some cases. The distribution of demonstrations is uneven, and EditMP tends to fail where the demonstrations are sparse. Therefore, uniformly distributed demonstrations can improve the adaptability of EditMP.

C. Controllably movement editing

A key goal of MPs is to adapt learned movements to new task scenes, which have different target positions, obstacle positions, and obstacle shapes. The movement editing should be controllable and interpretable since the robot has to reach the target position and avoid hitting obstacles (e.g., passing between two obstacles).

In this section, we evaluate the controllably movement editing with drawing rectangle and obstacle avoidance tasks. Fig. 8 visualizes the results of the latent space disentanglement in the drawing rectangle task. As introduced in Sec. III-B, we disentangle the latent space to explore the projection directions which correspond to the movement variables. When we sample the latent codes along the projection direction, the variation of the variable of the generated movement is approximately linear. For Fig. 8, we first explore the projection directions (d_{height} and d_{width}), which correspond to the height and width of the movement. Then we sample z_{00}. We sample $z_{i0}, i = 1, 2, 3$ along the d_{width} with z_{00} as the starting point. $z_{ij}, j = 1, 2, 3$ is sampled along the d_{height} with z_{i0} as the starting point. The movements in Fig. 8 are generated using

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

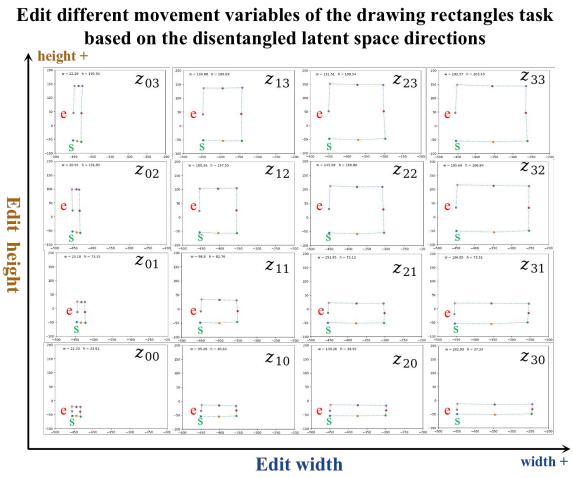

Fig. 8. This figure visualizes the results of the latent space disentanglement in the drawing rectangle task. We disentangle the latent space to find the projection directions (d_{height} and d_{width}) which correspond to the height and width of the movement. The movements are generated with the latent codes z that are sampled along these projection directions. "s" represents the starting point. "e" represents the ending point.

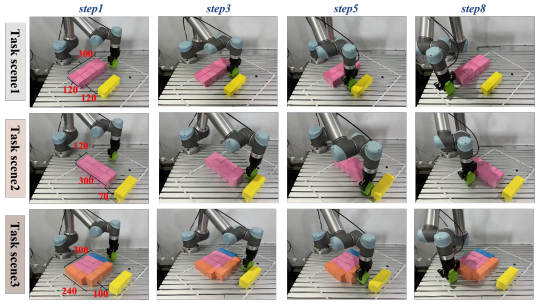

Fig. 9. This figure shows the results of the movement editing in obstacle avoidance tasks that have different obstacles. The red number in the figure indicate the size of the obstacle, which is measured in millimeters.

the sampled z, $\tau = G(z)$. When z moves along the d_{width}, the width of the movements become larger. Moreover, when different z move along the d_{height}, all the generated movements' heights become larger. This proves that the projection directions are generalized.

Notably, the linearity here is approximate linearity (We introduce this in section III-B). Therefore, when the latent code varies linearly, the change in the trajectory's height is approximately linear. Moreover, the changes in the movement variables are limited. The variation limits of different latent codes are different. We discuss the variation limits in section IV-D.

TABLE III
COVERAGE DISTRIBUTION OF GENERATED TRAJECTORIES

Precision	10 mm	5 mm
Coverage rate	99.61%(510/512)	83.98%(429/512)

Fig. 9 shows the results of the movement editing in the obstacle avoidance task. The white rectangular box indicates the reachable space of the robot's end-effector in this task ($-300mm <= x <= -580mm, -250mm <= y <= 250mm$ when $z = 106mm$). The red number in the figure indicates the obstacle's size, measured in millimeters. In the obstacle avoidance task, the robot is asked to grab the square and pass between the two obstacles (colored obstacle and yellow obstacle), and the square must not leave the table. The environment variables of this task are the width and the height of the colored obstacle and the distance between the yellow obstacle and the colored obstacle. All environment variables for *Task scene1* to *3* are significantly different. For example, in *Task scene1*, the height of the colored obstacle is 120 mm, and the width is 300 mm. In *Task scene2*, these two values are swapped. The height of the colored obstacle is 300 mm, and the width is 120 mm. Fig. 9 shows that EditMP can adapt to these new task scenes while ensuring compliance with the task's requirements.

TABLE IV
THE EDITING BOUNDARIES OF MOVEMENTS IN THE OBSTACLE AVOIDANCE TASKS

$W\ (mm)$	40	60	80	100	120
$L^D_{max}(mm)$	174	166	140.88	178.89	103.32
$L^G_{max}(mm)$	260.78	250.36	254.41	244.77	256.48
$W\ (mm)$	140	160	180	total	\
$L^D_{max}(mm)$	172.85	183.37	157.99	198	\
$L^G_{max}(mm)$	240.24	260.35	256.28	260.78	\

D. Boundaries of the movement editing

In this section, we evaluate the generality of EditMP with the obstacle avoidance task. There are two critical questions about the generality of EditMP. Can the distribution of the generated trajectory override the distribution of the demonstration trajectory? Where are the boundaries of movement editing?

To answer the first question, we train a generator with a dataset that contains 512 demonstration movements which are collected from different scenes with the same way as in section IV-B. Then, we test whether the EditMP can regenerate every movement of the dataset. We calculate the spatial distance between the end-effector of each step of the generated trajectory and the demonstration trajectory. When the average value of distances is less than a particular value (5mm or 10mm), we consider them to be the same trajectory. Table. III shows the results. The coverage rate equals the number of regenerated movements divided by the number of all demonstration movements. When the precision is 10 mm, the coverage rate is 99.61%. Only two demonstration movements cannot be regenerated. Notably, this does not mean that the generator cannot generate these two movements, only that the error is greater than 10mm. For the 5 mm precision, the coverage rate is 83.98%. These results prove that the generated movements' distribution overrides the demonstration movements' distribution.

For the second question, we show the editing boundaries of the movement length editing in Table. IV. W indicates the width of the obstacle. L^D_{max} and L^G_{max} are the maximum length of the obstacle that the demonstration (collected to train the EditMP) and generated movement can cross when the width of the obstacle is fixed. For each fixed obstacle width, we traverse the entire demonstration data set to find the maximum length of the obstacle that the demonstration movement can cross. For the L^G_{max}, we edit the movement to

find the maximum length of the obstacle that the movement can cross. The $total$ represents the maximum length of the obstacle the demonstration trajectories or the generated movements can cross.

From the Table. IV, we can see that L^G_{max} is larger than L^D_{max} in each fixed W. The $total$ of the demonstration is 198 mm, and that of the edited movement is 260.78 mm. The boundaries of movement editing are almost the same as the reachable space of the robot. This shows that EditMP is not just simply imitating demonstration movements. EditMP learns the intent of demonstration movements.

V. Conclusion

This work presents the EditMP, a transformer and GAN-based MP approach. Experiments demonstrate that EditMP can learn movements from high-variance demonstrations and has state-of-the-art adaptability. EditMP learns the intent of the demonstration movements instead of remembering the demonstration movements. For complex and large obstacles, EditMP can controllably and interpretably edit the learned movement at the joint level to avoid hitting obstacles since EditMP captures the complex cooperation between robot joints.

Task scenes in daily life are more complex and unstructured. Although, there are rules for defining environmental variables, often obstacle shape, obstacle position, and target location, VMs and VEs are task-dependent and need to be designed by users. Moreover, the adaptability of one movement cannot cover such a large gap between task scenes and demonstration scenes. The following points are the direction of our future work: (1) a more efficient way to encode environments; (2) design the MP library approach and determine the adaptability boundaries of each movement.

References

[1] Thibaut Kulak, Hakan Girgin, Jean-Marc Odobez, and Sylvain Calinon. Active learning of bayesian probabilistic movement primitives. *IEEE Robotics and Automation Letters*, 6(2):2163–2170, 2021.

[2] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14):1419–1434, 2021.

[3] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[4] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. *Advances in neural information processing systems*, 26, 2013.

[5] Sebastian Gomez-Gonzalez, Gerhard Neumann, Bernhard Schölkopf, and Jan Peters. Adaptation and robust learning of probabilistic movement primitives. *IEEE Transactions on Robotics*, 36(2):366–379, 2020.

[6] Emmanuel Pignat, Hakan Girgin, and Sylvain Calinon. Generative adversarial training of product of policies for robust and adaptive movement primitives. In *Conference on Robot Learning*, pages 1456–1470. PMLR, 2021.

[7] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

[8] Yanlong Huang, Leonel Rozo, Joao Silvério, and Darwin G Caldwell. Kernelized movement primitives. *The International Journal of Robotics Research*, 38(7):833–852, 2019.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[11] Amit H Bermano, Rinon Gal, Yuval Alaluf, Ron Mokady, Yotam Nitzan, Omer Tov, Oren Patashnik, and Daniel Cohen-Or. State-of-the-art in the architecture, methods and applications of stylegan. In *Computer Graphics Forum*, volume 41-2, pages 591–611. Wiley Online Library, 2022.

[12] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[13] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[14] Mojtaba Sharifi, Javad K Mehr, Vivian K Mushahwar, and Mahdi Tavakoli. Autonomous locomotion trajectory shaping and nonlinear control for lower limb exoskeletons. *IEEE/ASME Transactions on Mechatronics*, 27(2):645–655, 2022.

[15] Matteo Saveriano, Fares J Abu-Dakka, Aljaz Kramberger, and Luka Peternel. Dynamic movement primitives in robotics: A tutorial survey. *arXiv preprint arXiv:2102.03861*, 2021.

[16] Rudolf Lioutikov, Gerhard Neumann, Guilherme Maeda, and Jan Peters. Learning movement primitive libraries through probabilistic segmentation. *International Journal of Robotics Research*, 36(8):879–894, 1 2017.

[17] Yu Zhang, Long Cheng, Ran Cao, Houcheng Li, and Chenguang Yang. A neural network based framework for variable impedance skills learning from demonstrations. *Robotics and Autonomous Systems*, 160:104312, 2023.

[18] Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2018.

[19] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Restyle: A residual-based stylegan encoder via iterative refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6711–6720, 2021.

[20] Yen-Chi Cheng, Chieh Hubert Lin, Hsin-Ying Lee, Jian Ren, Sergey Tulyakov, and Ming-Hsuan Yang. Inout: Diverse image outpainting via gan inversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11431–11440, 2022.

[21] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[22] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8296–8305, 2020.

[23] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14104–14113, 2020.

[24] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4432–4441, 2019.

[25] Junyu Luo, Yong Xu, Chenwei Tang, and Jiancheng Lv. Learning inverse mapping by autoencoder based generative adversarial nets. In *International conference on neural information processing*, pages 207–216. Springer, 2017.

[26] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[27] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[28] Jacob Goldberger, Shiri Gordon, Hayit Greenspan, et al. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *ICCV*, volume 3, pages 487–493, 2003.

[29] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *IEEE transactions on pattern analysis and machine intelligence*, 44(4):2004–2018, 2020.