RMG: Real-Time Expressive Motion Generation with Self-collision Avoidance for 6-DOF Companion Robotic Arms

Jiansheng LI[1], Haotian SONG[1], Jinni ZHOU[1*], Qiang NIE[1*], and Yi CAI[1*]

***Abstract*— The six-degree-of-freedom (6-DOF) robotic arm has gained widespread application in human-coexisting environments. While previous research has predominantly focused on functional motion generation, the critical aspect of expressive motion in human-robot interaction remains largely unexplored. This paper presents a novel real-time motion generation planner that enhances interactivity by creating expressive robotic motions between arbitrary start and end states within predefined time constraints. Our approach involves three key contributions: first, we develop a mapping algorithm to construct an expressive motion dataset derived from human dance movements; second, we train motion generation models in both Cartesian and joint spaces using this dataset; third, we introduce an optimization algorithm that guarantees smooth, collision-free motion while maintaining the intended expressive style. Experimental results demonstrate the effectiveness of our method, which can generate expressive and generalized motions in under 0.5 seconds while satisfying all specified constraints.**

I. INTRODUCTION

With the advancement of robotic intelligence, robots are increasingly deployed in human-coexisting environments. Enhancing human-robot interaction has become a growing trend, particularly in the field of companion robots [1]. Beyond performing tasks and dialogue intelligently, expressive motions are crucial for physical interaction. Research indicates that expressive motions can convey interaction nuances beyond speech, offering subtle and engaging biological interactions that capture attention and foster acceptance [2]. The 6-DOF robotic arm, with its compact and highly flexible structure, is well-suited for various tasks and environmental integration, making it an excellent platform for home companion robots. Exploring how to utilize the rich postures of robotic arms to create expressive motions for better human interaction is an intriguing research topic, attracting attention from various fields such as robotics,psychology, and art[3].

While some research focus on explicit motion principles [4], these methods often lack generalization. Data-driven learning [5] offers a solution but is hindered by the scarcity of expressive motion data for robotic arms, as most datasets are functional. Cross-modal models [6] can produce high-quality motions but are typically too large for real-time use. Additionally, most methods ignore physical constraints, limiting them to simulations. Overall, current approaches fail to balance generalization, expressiveness, real-time performance, and real-world applicability [7]. Generating expressive motion for robotic arms faces several challenges.

[1]Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China `jli204@connect.hkust-gz.edu.cn`
*Co-corresponding authors with equal contribution.

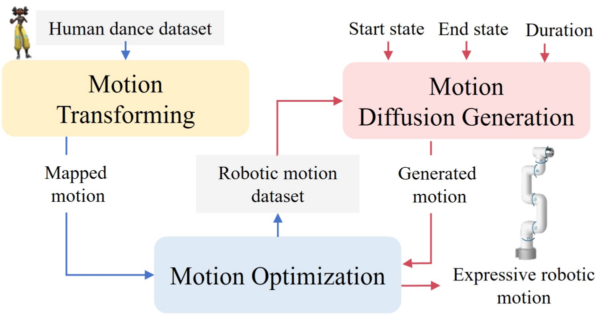
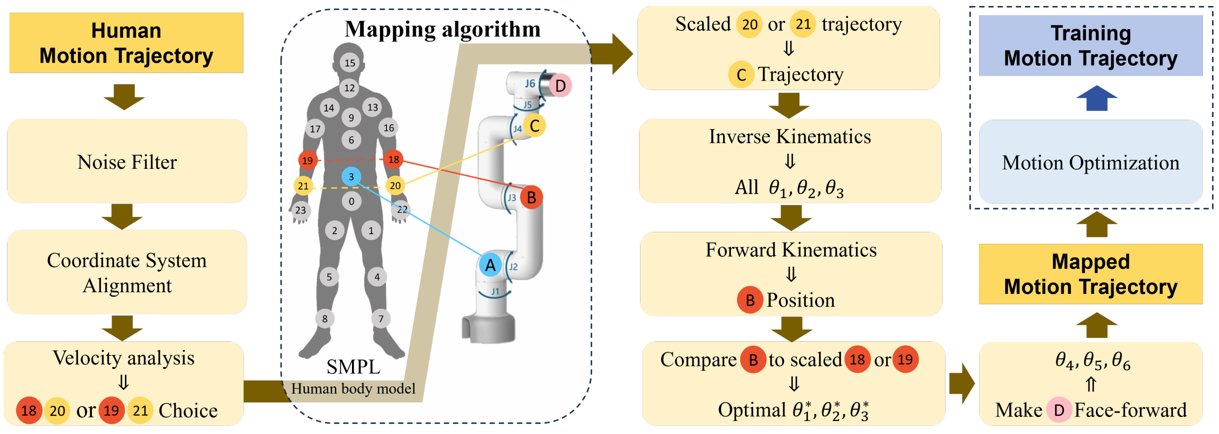


Fig. 1: **Overview of RMG.** Blue arrows indicate the robotic dataset construction pipeline, while red arrows represent the robotic motion generation process.

To address these challenges, this work develops a real-time and expressive motion generation system for 6-DOF robotic arms, enabling motion planning between arbitrary start and end points within specified time constraints. This allows the robot to execute expressive motions while completing tasks. Given the lack of suitable datasets, we designed a motion transforming algorithm that creates an expressive robotic arm dataset through mapping the human dance motions. To generalize these robotic motions, we employ a diffusion model in both Joint Space and Cartesian Space, and we investigate the performance differences between the two models, highlighting the importance of global pose information. To ensure collision-free, feasible, and smooth motions, we developed an optimization algorithm that outperforms existing methods in both real-time performance and motion style preservation. Experimental results demonstrate that our method can generate expressive smooth and collision-free motions within 0.5 seconds. Finally, a prototype showcases how the system integrates with speech and sensors to enhance human-robot interaction.

In summary, our approach enables expressive, collision-free, and smooth motion for robotic arms, inspired by human dance styles. The method is applicable to diverse robotic systems equipped with arms, enhancing human-robot interaction.

- A motion transforming algorithm that maps human dance to expressive robotic motions, creating a high-quality robotic dataset, which will be open-source.
- A real-time motion planner based on a diffusion model, balancing generalization and expressiveness.
- A real-time motion optimization algorithm that preserves the style of the motion trajectory.



Human Motion Trajectory
Noise Filter
Coordinate System Alignment
Velocity analysis ⇓ 18 20 or 19 21 Choice
Mapping algorithm
SMPL
Human body model
J6 D
J5
J4 C
J3 B
J2 A
J1
Scaled 20 or 21 trajectory ⇓ C Trajectory
Inverse Kinematics ⇓ All $\theta_1, \theta_2, \theta_3$
Forward Kinematics ⇓ B Position
Compare B to scaled 18 or 19 ⇓ Optimal $\theta_1^*, \theta_2^*, \theta_3^*$
$\theta_4, \theta_5, \theta_6$ ⇑ Make D Face-forward
Mapped Motion Trajectory
Motion Optimization
Training Motion Trajectory

Fig. 2: **Motion transforming Process.** After filtering and coordinate aligning the human motion trajectory, the total velocity at Points 20 and 21 decide the left or right trajectory. The chosen trajectory is then scaled and mapped into the robot arm's joint space. Finally, motion optimization generates a smooth, collision-free, and feasible trajectory.

II. Related Work

A. Motion Design and Mapping by Experts

Traditional approaches to designing expressive robot motions rely on expert-driven methods. Experts use animation principles [2], Laban analysis [8], and biomechanics [4] to generate pose-to-pose motions. While effective, these methods are labor-intensive. Another approach maps human motions to robotic motions, enabling rapid generation but requiring deep motion morphology understanding. For robotic arms, RoboGroove [9] maps human torso angles to arm joints, but focusing solely on the torso may reduce expressiveness. Saviano et al. [10] use PCA for human-robot motion mapping, which is innovative but lacks interpretability and controllability. These methods are also limited by predefined motion sets and fixed start and end points, hindering generalization.

B. Learning-based Motion Generation

Recent methods use multimodal inputs (e.g., music, sound, text) and techniques like diffusion models and transformers [6, 11–13] to generate high-quality motions. However, they are mostly limited to human or biological motion due to data availability. Additionally, their reliance on computationally intensive models restricts real-time performance. For robotic arms, the lack of expressive datasets has led to learning-based approaches, such as autoencoders [5] and GANs [10], to reconstruct robotic motions from human motion dataset. However, these techniques often rely on randomly generated or industrial datasets as base robotic motions, which may lack expressiveness and compromise the final results.

C. Motion Optimization Methods

To enable realistic robot interaction, expressive motions must be optimized for collision-free, dynamically feasible execution while preserving style. Methods like TrajOpt [14], STOMP [15], and CHOMP [16] optimize paths through functional constraints but often lose motion style due to global optimization. Whole-Body Control (WBC) [17] enables real-time local obstacle avoidance and style preservation but requires high-performance hardware and complex modeling, limiting its use in low-cost robots. In this work, we propose a novel method that enables real-time obstacle avoidance on low-cost robots while effectively preserving trajectory shape and motion style.

To achieve real-time and generalizable expressive motion generation for 6-DOF companion robotic arms in real-world environments, Our approach addresses these challenges by explicitly mapping human dance data to robotic motions, leveraging diffusion models for generalization, simplifying the network for real-time performance and optimizing motions with style preservation for feasibility.

III. System Overview

The motion generation framework RMG is illustrated in Fig. 1. Our key insight is to use dance motions to construct the robotic dataset, as dance embodies well-designed, widely accepted expressive movements. The framework consists of three modules: (1) the Motion Transforming module, which maps human dance motions to robotic arm motions; (2) the Motion Diffusion Generation module, which learns to generate motions based on start, end, and duration conditions; and (3) the Motion Optimization module, which post-processes motions from the preceding modules to ensure smoothness, feasibility, and collision avoidance. We implemented this framework on the Mycobot 280, a low-cost ($1098) UR desktop robotic arm with a maximum reach of 280 mm, making it ideal for human-robot interaction scenarios.

IV. Motion Transforming

To construct the expressive motions dataset for robotic arm , we utilized the open-source human dance dataset, Popdancet [6], which comprises over 12,819 seconds of high-quality human dance motion data. The dataset is formatted in the SPML human body model in 60 FPS.

A. Transforming Process

The process, illustrated in Fig. 2, maps human motion to robotic arm motion by focusing on expressive aspects such as regularity, fluidity, and interactivity. Given the robotic arm's limited degrees of freedom, we prioritize replicating the most interactive trajectory. The hand, with its high motion density and expressive significance, is central to this process. By concentrating on the wrist trajectory (the root of the hand), we simplify the motion while preserving its essential features. We then analyze the total velocities of the trajectories of Points 20 and 21, selecting the one with the higher magnitude as t_{wrist}, as it best captures the dynamic and interactive nature of the motion.

To map t_{wrist}, we select Point C on the robotic arm. This point is controlled by the first three joints (J1, J2, J3) and determines the arm's primary posture.The base of the robotic arm (Point A) is aligned with Point 3 in the human system, ensuring the mapped trajectory t_C covers most of the workspace while avoiding extreme spatial concentrations. t_C is a scaled version of t_{wrist}, with Point 3 as the origin, ensuring solvability within the robotic arm's workspace.

Using inverse kinematics, we analytically solve t_C to derive all feasible $\theta_1, \theta_2, \theta_3$ for J1, J2 and J3. To resolve the multi-solution problem, we use the human elbow trajectory t_{elbow} as a reference, mapping the human elbow (Point 18 or 19) to Point B on the robotic arm. We compute t_B using forward kinematics and minimize the following function:

$$\begin{aligned} \theta_1^*, \theta_2^*, \theta_3^* = \arg \min_{\theta_1,\theta_2,\theta_3} (|t_B - scale \cdot t_{\text{elbow}}|) \\ \text{subject to} \quad t_C = scale \cdot t_{\text{wrist}}. \end{aligned} \tag{1}$$

This ensures that t_B closely mimics t_{elbow}, promoting smooth and continuous motion. It also enhances expressivity by capturing the spatial relationships between the wrist, elbow, and Point 3.

To enhance interactivity, we use J4, J5, and J6 (which determine Point D's orientation) to mimic the orientation of the human head (Point 15). Additionally, we constrain Point D to face forward most of the time by controlling J4 and J5:

$$\theta_4 = -(\theta_2 + \theta_3), \quad \theta_5 = -\theta_1 \tag{2}$$

Due to morphological differences, the mapped angles may cause self-interference. Motion optimization (Sec. VI) refines the mapped trajectories to ensure they are collision-free and smooth. This process generates a dataset of over 10,000 collision-free, compliant, and highly expressive motion trajectories for the robotic arm.

V. Motion Diffusion Generation

Given the robotic motion dataset from motion transforming (Sec. IV), we train two diffusion models: the J Model for motion generation in joint space and the C Model for Cartesian space. The pipeline is shown in Fig. 3. Given an arbitrary start state x_{start} , end state x_{end}, and motion duration (represented by the l_{valid}), these models synthesize smooth and expressive motion trajectories.

A. Data Preprocessing and Postprocessing

a) Preprocessing: We preprocess motion data from our robotic arm dataset, initially recorded in joint space. The raw data is segmented into motion sequences of 120 to 400 frames based on zero-velocity points. For real-time applicability, we downsample the data from 60 Hz to 12 Hz and pad all trajectories to 80 time steps using a masking strategy. In the C Model, positions of t_C is extracted as input, which contain the message of θ_1, θ_2, θ_3. By focusing exclusively on Point C, we streamline the x_{start} and x_{end} using only Point C's position and enhance computational efficiency. In the J Model, θ_1, θ_2, θ_3 are directly used as input features, which also inherently define t_C. To enhance temporal consistency and smoothness, we compute velocity components via finite differences and concatenate them with the original data, yielding the following structured representations:

Joint Space Representation:

$$X_{\text{angle}} \in \mathbb{R}^{l\times 6} : [\theta_1, \theta_2, \theta_3, V_{\theta_1}, V_{\theta_2}, V_{\theta_3}].$$

Cartesian Space Representation:

$$X_{\text{cartesian}} \in \mathbb{R}^{l\times 6} : [X, Y, Z, V_x, V_y, V_z].$$

In this work, l refers to 80.

b) Postprocessing: Given the denosied sequence, we first extract the valid positional segment $X' \in \mathbb{R}^{l_{\text{valid}}\times 3}$. The start and end points are then precisely aligned through interpolation to ensure trajectory continuity. For the C Model, joint angles are computed using inverse kinematics. To simplify computation and maximize interactivity, angles for J4, J5, and J6 are configured to maintain the end effector's forward-facing orientation. Finally, the angular motions $X \in \mathbb{R}^{l_{\text{valid}}\times 6}$ are optimized using the algorithm described in Sec. VI to ensure smoothness and collision avoidance.

B. Diffusion Framework

Diffusion models are widely used for high-quality motion generation due to their ability to capture spatiotemporal relationships in long sequences. In this study, we employ Denoising Diffusion Implicit Models (DDIM) [18], which improves inference efficiency by reducing the number of required time steps through implicit sampling. However, DDIM's limited consideration of variance can restrict sample diversity. To address this, we integrate the improved Denoising Diffusion Probabilistic Models (iDDPM) method [19] during training, which incorporates a variational lower bound loss (vlb). The vlb loss minimizes the discrepancy between the generated and true data distributions:

$$\begin{aligned} L_{vlb}(t) = \mathbb{E}_{x_t\sim q} [-\log p(x_t|x_{t-1})] \\ + D_{KL}(q(x_{t-1}|x_t) \| p(x_{t-1}|x_t)) \end{aligned} \tag{3}$$

The first term is the negative log-likelihood, which evaluates the model's prediction accuracy, while the second term is the Kullback-Leibler divergence, ensuring the generated distribution aligns closely with the true data distribution. By optimizing both mean and variance, this approach enables the generation of more diverse and high-quality samples.

C. Encorder block

The encoder block is designed to integrate conditional features. Initially, we sequentially concatenate the conditional vectors and process them through MLP to extract

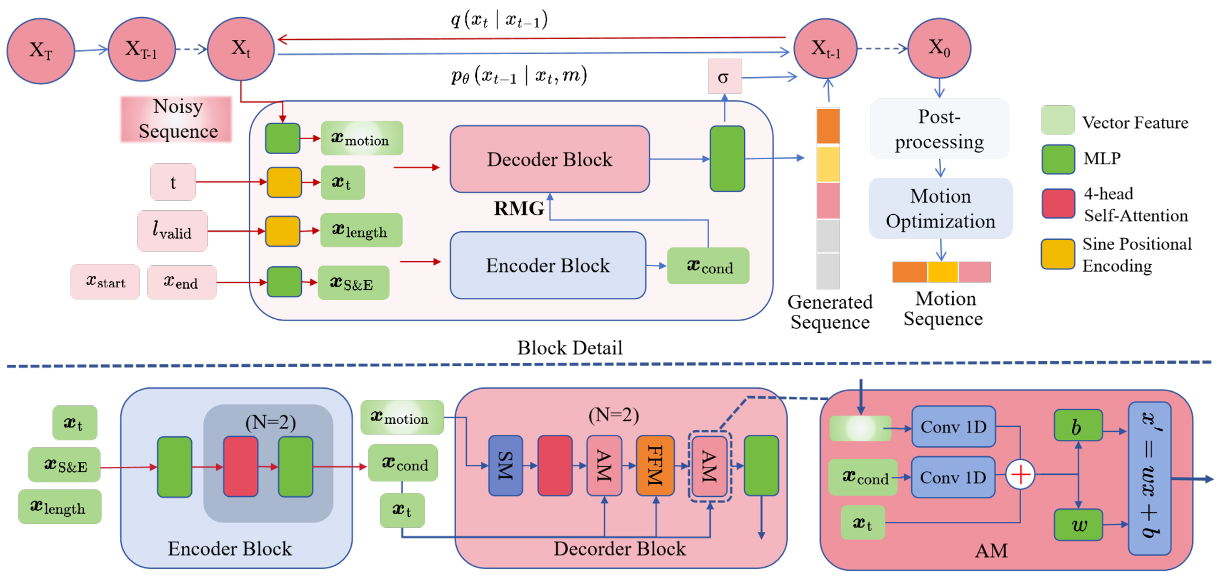


Fig. 3: **Motion Diffusion Generation Overview.** The J Model and C Model share the same model structure. This system learns to denoise motion sequences from time $t = T$ to $t = 0$, where t represents the diffusion steps. l_{valid} is the valid sequence length to determine the motion time. In the C Model, start state x_{start} and end state x_{end} denote coordinate values, while in the J Model, they correspond to θ_1, θ_2, θ_3. All input conditions are converted into vector feature $x \in \mathbb{R}^{b\times h}$, and the noise sequence is converted to $x_{\text{motion}} \in \mathbb{R}^{b\times l\times h}$, where L is the whole sequence length and h is set to 256 as the hidden layer dimension. N refers to the stack number. The postprocessing of the denoised sequence can be found in Sec. V-A.0.b.

and combine essential features from these vectors. Following this, we apply a 4-head self-attention layer to capture the intricate relationships between the extracted features, thereby enhancing the model's expressive capability.

D. Decoder Block

The decoder block is responsible for decoding action sequences with style characteristics under specific conditions, utilizing various modules.

1) Spatial Module (SM): SM is designed to capture spatial relationships between different directions or joints using a 1D convolution. The input vector $x_{\text{motion}} \in \mathbb{R}^{b\times l\times h}$ is first transposed before being passed into the SM. This transformation generates an attention map that focuses on the spatial aspects of the data.

2) Self-attention: To capture temporal relationships within the input sequence, a 4-head self-attention layers is employed. The positionally encoded x_{motion} serves as both the query and key, while the original x_{motion} acts as the value. These components are processed through the classical attention mechanism [20].

3) Alignment Module (AM): We adopted the AM in POPDG [6], as illustrated in Fig. 3. It integrates motion features with conditions via 1D convolution and linear transformation, enhancing feature-condition alignment and improving the accuracy and coherence of generated sequences.

4) Feature Fusion Module (FFM): Shown in Fig. 4, FFM deeply integrates action features with conditional features. To ensure a smooth transition in the generated motion sequence from a start point to an endpoint with minimal deviation, the module places special emphasis on the sequence points near these critical points during the fusion process. It utilizes a dynamic weighting mechanism that adjusts the fusion of

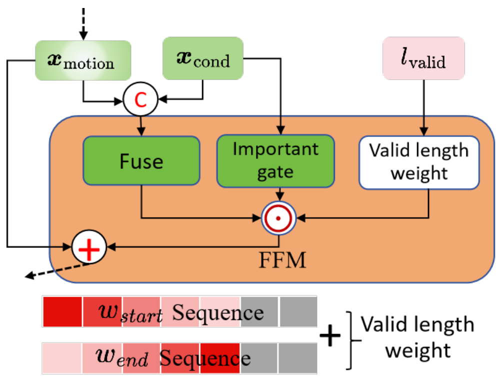


Fig. 4: **Feature Fusion Module.** In the w sequence, the gray areas indicate invalid parts, and the color intensity in the valid sections reflects the weight. The sum of the two sequences gives the valid length weight.

features based on l_{valid}. Specifically, the module calculates the sorting distance from each sequence point to both the start and end of the valid sequence, and then applies a Gaussian function to generate the corresponding weights:

$$\begin{aligned} w_{\text{end}} &= \exp\left(-\frac{2\cdot \text{end\_distance}^2}{\sigma^2}\right) \\ w_{\text{start}} &= \exp\left(-\frac{2\cdot \text{start\_distance}^2}{\sigma^2}\right) \end{aligned} \tag{4}$$

where σ is a parameter related to the valid sequence length. The closer a position is to either the start or end point, the higher the corresponding weight will be. The final valid length weight is the sum of w_{start} and w_{end}.

Furthermore, the importance gate processes the conditional features to enhance focus on key attributes. Subsequently, the condition and sequence features are fused through a transformation network. The final output combines the original sequence features with the weighted fused features, thereby enhancing the model's adaptability, expressiveness, and comprehension of complex temporal data, while also providing improved interpretability.

E. Loss Function

During training, additional losses complement the VLB loss (Eq. 3). Let $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}_{\theta,t}$ denote the original and generated data at diffusion step t, with [pad] and [valid] representing masked and valid portions, respectively.

Loss for the masked portion $\mathcal{L}_{\text{pad}}$:

$$\mathcal{L}_{\text{pad}} = \mathbb{E}_{\boldsymbol{x},t}\left[\|\boldsymbol{x}[\text{pad}] - \hat{\boldsymbol{x}}_{\theta,t}[\text{pad}]\|_2^2\right] \tag{5}$$

Loss for the valid portion $\mathcal{L}_{\text{pv}}$: where the first three dimensions represent position loss and the last three dimensions represent velocity loss:

$$\mathcal{L}_{\text{pv}} = \mathbb{E}_{\boldsymbol{x},t}\left[\|\boldsymbol{x}[\text{valid}] - \hat{\boldsymbol{x}}_{\theta,t}[\text{valid}]\|_2^2\right] \tag{6}$$

By differentiating the valid sequence portion, we obtain velocity and acceleration losses $\mathcal{L}_{\text{va}}$. Unlike the velocity loss in $\mathcal{L}_{\text{pv}}$, this velocity loss focuses on the smoothness and continuity of the position part:

$$\mathcal{L}_{\text{va}} = \mathbb{E}_{\boldsymbol{x},t}\left[\left\|\boldsymbol{x}'[\text{valid}] - \hat{\boldsymbol{x}}'_{\theta,t}[\text{valid}]\right\|_2^2\right] \tag{7}$$

The start and end errors of the valid sequence portion are treated as start and end losses $\mathcal{L}_{\text{start}}$ and $\mathcal{L}_{\text{end}}$:

$$\mathcal{L}_{\text{start}} = \mathbb{E}_{\boldsymbol{x},t}\left[\|\boldsymbol{x}[\text{valid}][0] - \hat{\boldsymbol{x}}_{\theta,t}[\text{valid}][0]\|_2^2\right] \tag{8}$$

$$\mathcal{L}_{\text{end}} = \mathbb{E}_{\boldsymbol{x},t}\left[\|\boldsymbol{x}[\text{valid}][-1] - \hat{\boldsymbol{x}}_{\theta,t}[\text{valid}][-1]\|_2^2\right] \tag{9}$$

The final total loss is expressed as:

$$\mathcal{L}_{\text{total}} = w_0\mathcal{L}_{\text{vlb}} + w_1\mathcal{L}_{\text{pv}} + w_2\mathcal{L}_{\text{va}} + w_3\mathcal{L}_{\text{pad}} + w_4\mathcal{L}_{\text{start}} + w_5\mathcal{L}_{\text{end}} \tag{10}$$

w denotes the weights of the losses.

VI. Motion Optimization

To ensure that the robotic arm's motions are collision-free, smooth while preserving the motion style. we optimized the motion trajectories generated by mapping algorithms and diffusion models.

We developed an obstacle avoidance algorithm using Particle Swarm Optimization (PSO) [21], integrated with Curobo [22] for efficient parallel collision detection in joint space. The algorithm evaluates collision costs $Curobo(x_n)$ for each point x_n in a motion sequence $X \in \mathbb{R}^{N\times 6}$ to identify collision points.For each of these points, we update 50 particles and compute their fitness f:

$$f = 10000 \times Curobo(x_n) + \Delta\theta \tag{11}$$

The fitness function is designed to strictly avoid collisions while minimizing changes in joint angles to preserve motion style. Particle positions are updated iteratively using the PSO algorithm until the fitness values of all collision points fall below a predefined threshold. The specific algorithm implementation can be summarized by Algorithm 1.

Algorithm 1 PSO for Collision Avoidance

```
1:  Input: $X \in \mathbb{R}^{N\times 6}, w, c_1, c_2$
2:  $r_1, r_2 \sim$ Uniform$(0,1), f_{gbest} = \infty$
3:  for each $n \in C = \{n \mid Curobo(x_n) > 0\}$ do
4:      Initialize particles: $p_i, v_i, pbest_i = p_i, f_{pbest_i} = \infty$
5:      repeat
6:          for $i = 1$ to $n$ do
7:              $f_i = 10000 \times Curobo(p_i) + \Delta\theta_i$
8:              if $f_i < f_{pbest_i}$ then
9:                  $pbest_i = p_i, f_{pbest_i} = f_i$
10:             end if
11:             if $f_i < f_{gbest}$ then
12:                 $gbest = p_i, f_{gbest} = f_i$
13:             end if
14:             $v_i = w \cdot v_i + c_1 \cdot r_1 \cdot (pbest_i - p_i)$
15:                 $+c_2 \cdot r_2 \cdot (gbest - p_i)$
16:             $p_i = p_i + v_i$
17:         end for
18:     until $f_{gbest} <$ threshold
19: end for
```

If the fitness values remain above the threshold and collisions persist after 30 iterations, problematic points are removed. We then apply B-spline fitting and Gaussian filtering to smooth the trajectory, followed by a recheck for collisions. Finally, we use the TOPP-RA algorithm [23] from MoveIt to replan the motion timing, avoiding sudden accelerations or decelerations and ensuring smooth execution within the robotic arm's velocity and acceleration limits.

VII. Experiments and Analysis

In this section, we conduct experiments and analyze each module separately. All training and computations are performed on a computer with an Nvidia 4070 GPU, with results demonstrated in the Gazebo simulation environment.

A. Motion Optimization Effects

We compare our motion optimization algorithm (Sec. VI) with Curobo's TrajOpt in terms of speed and motion style preservation. Motion style preservation is evaluated by comparing the similarity between the optimized robotic trajectories and the original human trajectories. Specifically, we compare the robot's t_C with the human's scaled t_{wrist} and the robot's t_B with human's scaled t_{elbow}, as these represent the primary motions. Comparing with the original human trajectories provides a unified benchmark for subsequent experiments.

Motion features are assessed using a vector $X \in \mathbb{R}^{35}$, derived from frequency characteristics, motion features, and trajectory morphology. Similarity in motion features is measured using Frechet Inception Distance (FID) [24] for distribution distance, while trajectory sequence similarity is evaluated using average Dynamic Time Warping (DTW) [25]. DTW finds the optimal matching path between two trajectories and calculates the minimum cumulative distance after normalization and length effect elimination.

Method	FID$_C$ ↓	FID$_B$ ↓	DTW$_C$ ↓	DTW$_B$ ↓	T_{opt} ↓
TrajOpt [14]	159.97	473.34	0.38	1.31	0.65 s
RMG(ours)	**6.17**	**94.72**	**0.11**	**0.75**	**0.29 s**

TABLE I: **Algorithm Comparison.** Subscript C denotes the robot's t_C metrics and subscript B denotes robot's t_B metrics. TrajOpt [14] is an algorithm in Curobo [22] and tends to prioritize minimum-time collision-free paths, resulting in a significant loss of human stylistic elements. T_{opt} represents the optimization time for trajectories with an average of 20% collision points and a sequence length of 200. Additionly, RMG's average optimization time is 105 seconds without parallel processing.

Results in Table I show that, unlike TrajOpt—which optimizes the entire trajectory simultaneously, altering previously collision-free points and degrading path shape and style—our approach better preserves trajectory style, as evidenced by significantly lower FID and DTW scores. Furthermore, leveraging GPU parallel computing, our method achieves a 300× speedup over serial computation and is over two times faster than TrajOpt, fully meeting real-time requirements.

B. Motion Transforming Effects

The quality of motion transforming in Sec. IV depends on motion expressivity and mapping similarity.

1) Similarity: Table I also reflects the similarity between the final robotic transformed motions and the original human motions, as t_C vs. t_{wrist} and t_B vs. t_{elbow} represent the primary mapping relationships. Our method achieves significantly lower FID and DTW values compared to the TrajOpt group, which optimizes the same mapped motions and serves as a baseline for non-similarity. This demonstrates its ability to preserve dance characteristics. The FID and DTW values for t_C are notably lower than those for t_B, indicating that t_C (mapped from the wrist) closely resembles the original motion. In contrast, t_B captures only partial elbow characteristics because t_{elbow} serves as an auxiliary reference and is not always reachable due to structural constraints, which explains its higher FID and DTW values.

2) Expressivity: We evaluate motion expressivity by comparing the our method with RoboGroove [9], which uses torso joint angles (Points 0, 6, 15) as robotic joint angles. Motion diversity is a key indicator of expressivity. To quantify this, we compute the average Euclidean distance between 40 mapped robotic motions in the feature space of t_B and t_C (the same as used for FID in Sec. VII-A).

Smoothness is another key characteristic, measured using the average Curvature Change Rate (CCR) of t_C. CCR quantifies the rate of curvature variation along a trajectory, where a lower value indicates smoother motion. It is calculated as:

$$\text{CCR} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{d^2\mathbf{r}(s_i)}{ds^2}\right| \tag{12}$$

where $\mathbf{r}(s_i)$ is the position at arc length s_i, and N is the number of sampled points.

Other aspects of expressivity are assessed through a user study with 20 participants. Participants rate the generated motions form both methods on biologicality (lifelike and fluid motions), playfulness (dynamic, graceful, and varied motions), and interactivity (engagement with a hypothetical person) on a scale of 1 to 5.

Method	Dist↑	CCR ↓ [m^{-2}]	Bio-logicality ↑	Playful-ness ↑	Inter-activity ↑
Robo-Groove[9]	5.52	**80.02**	3.2	2.6	2.75
RMG(ours)	**8.06**	91.63	**4.2**	**4.25**	**4.3**

TABLE II: **Expressiveness Evaluation For Motion Transforming.**

Results in Table 2 show that RMG outperforms RoboGroove in all categories except CCR, where RMG's score is slightly higher. However, this is acceptable because RMG achieves a higher Dist score, indicating increased motion complexity. This demonstrates that hand-based mapping yields better expressivity than torso-based mapping, while also achieving a better balance between motion complexity and smoothness.

C. Motion Diffusion Generation Effects

In motion diffusion generation, both the J Model and C Model were trained for 2k epochs, including a 20-epoch warm-up phase. Optimization used the Adan optimizer with a learning rate of 2e-4 and a weight decay rate of 1.2e-4.

Since our model focuses on generating expressive motion trajectories rather than traditional path planning tasks aimed at finding optimal paths, we evaluate overall performance in both planning ability and expressivity. Similar works include ELEGNT [2] and Osorio et al.'s work [5], but neither is open-source, and their tasks are different from ours. We only compare and analyze the J model and C model by generating robot motions for 40 test set conditions, using the corresponding robotic dataset motions obtained from motion transforming as the oracle. The results are presented in TABLE III.

1) Planning Ability:

a) Errors in Condition: Both models achieve minimal errors in start and end points (Δ_{start} and Δ_{end}). Additionally, the motion duration deviation ΔT_{Motion} in the J Model is only 0.34 seconds from the dataset's reference motion time, 6 times smaller than the C Model's, demonstrating superior temporal control.

b) Optimization Rate: the raw motion sequences contain points that require optimization due to collisions and inverse kinematics failures, quantified by the optimization rate η_{opt}. The J Model achieves a 4.4× lower η_{opt}, indicating a superior understanding of the free configuration space. In contrast, the C Model relies more heavily on post-processing optimization, likely due to its training data being restricted to the Point C trajectory without global information.

c) Speed: Both models achieve real-time performance with T_{gen} under 0.5s. However, the C Model requires 1.4 times longer due to additional kinematic processing.

Overall, the J Model outperforms the C Model in planning ability, demonstrating superior temporal control, lower optimization requirements, and faster motion generation.

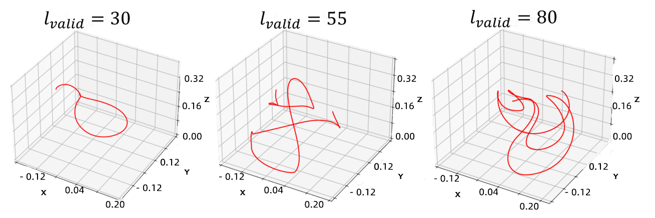


Fig. 5: **Point C trjectory t_C in the J Model**

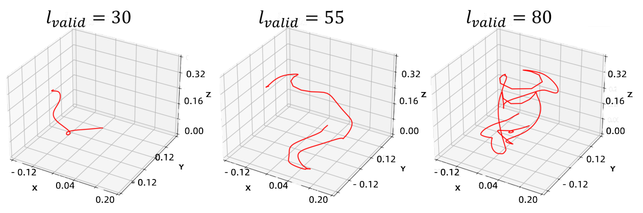


Fig. 6: **Point C trjectory t_C in the C Model**

	Motion Quality		Diversity	Smoothness	Planning Ability				
	$\text{FID}_\text{C}\downarrow$	$\text{FID}_\text{B}\downarrow$	Dist $\uparrow$	CCR$[\text{m}^{-2}]$	$\lvert\Delta_\text{start}\rvert\downarrow$	$\lvert\Delta_\text{end}\rvert\downarrow$	$\eta_\text{opt}\downarrow$	T_{gen}	$\lvert\Delta_{T_{Motion}}\rvert\downarrow$
Robotic Dataset	6.17	94.72	8.06	91.63	-	-	-	-	-
J Model	**19.86**	**114.72**	8.96	**83.10**	0.05 rad	0.22 rad	**4.1%**	**0.37 s**	**0.33 s**
C Model	81.75	287.16	**13.32**	127.8	8.71 mm	13.30 mm	17.9%	0.46 s	1.86 s

TABLE III: **Motion Generation Evaluation.** The robitc dataset is generated from motion transformation (Sec. IV). Subscript C denotes the robot's t_C metrics and subscript B denotes robot's t_B metrics. Δ_start and Δ_end represent the average deviation from the given start and end points before fine tuning. η_opt indicates the proportion of points in the original sampled path that need adjustment. T_gen is the motion generation time. Δ_{T_Motion} is the motion time error in simulation from the original motion.

2) Motion Expressivity:

a) Motion Quality: We evaluated the feature distribution distance between human motions and generated robot motions using FID (Sec. VII-A) to assess the inheritance of human dance motion expressivity. The J Model's FID is slightly higher than the baseline, indicating improved generalization while preserving motion characteristics. In contrast, the C Model exhibits significantly higher FID values, especially for FID_B, likely due to its reliance on post-processing optimization and the lack of Point B data in training.

b) Diversity: Diversity was assessed by computing Dist (Sec. VII-B.2). The J Model exhibits slightly higher Dist than the robotic dataset, reflecting its generalization capability. In contrast, the C Model's Dist value is substantially higher, correlating with its elevated FID. This increased diversity in the C Model results from trajectory optimization required for 17.9% of points, leading to more chaotic trajectories.

c) Smoothness: The CCR of t_C is used to evaluate the motion smoothness. The J Model's CCR closely matches the robotic dataset, demonstrating a better balance of smoothness and motion complexity. However, the C Model exhibits significantly higher CCR values, which may lead to less natural motion transitions.

d) Visualization and User Study: We visualize motions generated under different l_{valid} (30, 55, 80) by plotting t_C. As shown in Fig. 5 and 6, motions become more complex as l_{valid} increases, highlighting the impact of length conditions on motion generation. The J Model produces smoother and more diverse trajectories, while the C Model generates more rigid and less natural motions, aligning with the CCR values.

In the user study, 80% of participants preferred the motions generated by the J Model, a preference consistent with the visual characteristics of the trajectories.

In conclusion, while both models can generate expressive motion planning in real-time, the J Model, with its better understanding of the configuration space during training, produces higher-quality and more expressive motions.

Model	Method	$\text{FID}_\text{C}\downarrow$	$\text{FID}_\text{B}\downarrow$	$\lvert\Delta_\text{start}\rvert\downarrow$	$\lvert\Delta_\text{end}\rvert\downarrow$	$\eta_\text{opt}\downarrow$	$\lvert\Delta_{T_M}\rvert\downarrow$
J Model	RMG	19.86	114.72	0.05 rad	0.22 rad	4.1%	0.33 s
	w/o SM	23.92	118.53	0.08 rad	0.14 rad	9.1%	0.37 s
	w/o FFM	22.72	115.92	0.09 rad	0.66 rad	7.5%	0.40 s
	w/o AM	28.32	124.98	0.24 rad	0.31 rad	12.1%	0.51 s
C Model	RMG	81.75	287.16	8.71 mm	13.3 mm	17.9%	1.86 s
	w/o SM	137.47	380.84	8.52 mm	15.1 mm	36.7%	2.26 s
	w/o FFM	90.85	300.14	10.1 mm	48.3 mm	26.6%	2.54 s
	w/o AM	198.98	458.26	7.93 mm	9.22 mm	39.7%	2.67 s

TABLE IV: **Ablation Study.** Without SM, performance declines across all metrics, with η_opt doubling. Without FFM, Δ_end increases by more than threefold. Without AM, all metrics deteriorate, though the J Model shows less degradation compared to the C Model.

D. Ablation Study for Motion Diffusion Generation

We conducted ablation experiments in our Generation Models to evaluate the effectiveness of the key design components: SM, AM, and FFM. As shown in TABLE IV, the J Model demonstrates greater robustness compared to the C Model. With SM, the model better captures the style and spatiotemporal features of trajectories, leading to improved motion quality (FID) and planning ability (Δ_start, Δ_end, η_opt, Δ_{T_M}). The FFM enhances the model's ability to fuse x_{start}, x_{end} and l_{valid}, significantly reducing Δ_start and Δ_end. Additionally, AM improves the alignment of condition and trajectory characteristics and achieve a better balance between motion quality and planning capability, resulting in overall superior performance.

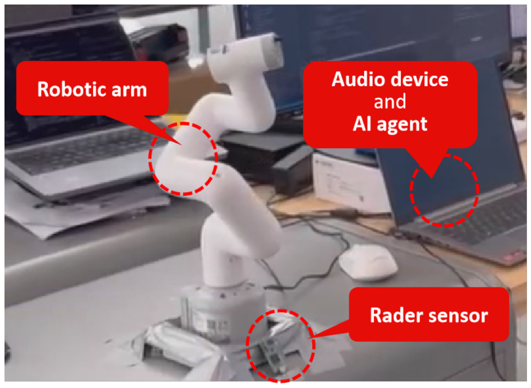


Fig. 7: **Prototype Overview.**

VIII. Deployment

We successfully deployed the generative model on the low-cost desktop robotic arm Mycobot280 for real-time interaction. As shown in Fig. 7, the arm is equipped with a radar sensor for gesture recognition, an AI agent, and audio devices for voice interaction. When the sensor detects a gesture, the arm generates an expressive motion. During AI interactions, the agent estimates speech duration and randomly selects an end state, which is fed into the generation model to create a corresponding expressive motion. With our method, the motion is non-repetitive and can be generated in real-time from any start and end point. The results demonstrate that interactions are engaging, particularly during motion demonstrations accompanied by voice.

IX. Conclusion

In this work, we demonstrated real-time expressive motion interaction using a 6-DOF companion robotic arm. We introduced a mapping method to extract expressive features from human motion and created a corresponding robotic arm expressive motion dataset. Additionally, we developed a motion optimization algorithm to ensure smoothness, avoid self-collisions, and preserve motion style. Furthermore, we designed a iDDPM-based network capable of generating expressive motions for the robotic arm based on a start state, end state, and motion duration. We trained both Cartesian space and joint space models, with both meeting real-time interaction requirements. With global pose information, the joint space model outperformed the Cartesian space model due to its superior understanding of the configuration space.

There are some limitations in our study. In the prototype, due to the robot's limited performance and challenges in maintaining precise speed control, it frequently overshoots, stutters, and struggles to track trajectories accurately. Additionally, motion generation is constrained by predefined trajectory endpoints and durations, lacking emotional classification, which limits its ability to produce contextually appropriate responses. Future work will focus on enabling the robotic arm to autonomously select emotional responses, determine motion endpoints, and adjust motion duration based on contextual analysis, paving the way for more natural and intelligent human-robot interactions.

References

[1] P. Vanc, J. K. Behrens, K. Stepanova, and V. Hlavac, "Communicating human intent to a robotic companion by multi-type gesture sentences," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 9839–9845.

[2] Y. Hu, P. Huang, M. Sivapurapu, and J. Zhang, "Elegnt: Expressive and functional movement design for non-anthropomorphic robot," 2025. [Online]. Available: https://arxiv.org/abs/2501.12493

[3] R. Stock-Homburg, "Survey of Emotions in Human–Robot Interactions: Perspectives from Robotic Psychology on 20 Years of Research," vol. 14, no. 2, pp. 389–411. [Online]. Available: https://doi.org/10.1007/s12369-021-00778-6

[4] G. Xia, R. B. Dannenberg, J. Tay, and M. M. Veloso, "Autonomous robot dancing driven by beats and emotions of music," in *Adaptive Agents and Multi-Agent Systems*, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:11293294

[5] P. Osorio, R. Sagawa, N. Abe, and G. Venture, "A Generative Model to Embed Human Expressivity into Robot Motions," *Sensors*, vol. 24, no. 2, p. 569, Jan. 2024.

[6] Z. Luo, M. Ren, X. Hu, Y. Huang, and L. Yao, "POPDG: Popular 3D Dance Generation with PopDanceSet," May 2024.

[7] G. Venture and D. Kulić, "Robot expressive motions: A survey of generation and evaluation methods," *J. Hum.-Robot Interact.*, vol. 8, no. 4, Nov. 2019. [Online]. Available: https://doi.org/10.1145/3344286

[8] M. Masuda, S. Kato, and H. Itoh, "A Laban-Based Approach to Emotional Motion Rendering for Human-Robot Interaction," in *Entertainment Computing - ICEC 2010*, H. S. Yang, R. Malaka, J. Hoshino, and J. H. Han, Eds. Berlin, Heidelberg: Springer, 2010, pp. 372–380.

[9] A. Rogel, R. Savery, N. Yang, and G. Weinberg, "RoboGroove: Creating Fluid Motion for Dancing Robotic Arms," in *Proceedings of the 8th International Conference on Movement and Computing*, ser. MOCO '22. Association for Computing Machinery, pp. 1–9. [Online]. Available: https://dl.acm.org/doi/10.1145/3537972.3537985

[10] P. Osorio and G. Venture, "Control of a Robot Expressive Movements Using Non-Verbal Features," *IFAC-PapersOnLine*, vol. 55, no. 38, pp. 92–97, Jan. 2022.

[11] R. Li, S. Yang, D. A. Ross, and A. Kanazawa, "AI Choreographer: Music Conditioned 3D Dance Generation with AIST++," Jul. 2021.

[12] S. Wang, J. Zhang, W. Cao, X. Hu, M. Li, X. Ji, X. Tan, M. Li, Z. Xie, C. Wang, and L. Ma, "MMoFusion: Multi-modal Co-Speech Motion Generation with Diffusion Model," May 2024.

[13] M. Zhang, Z. Cai, L. Pan, F. Hong, X. Guo, L. Yang, and Z. Liu, "MotionDiffuse: Text-Driven Human Motion Generation With Diffusion Model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 6, pp. 4115–4128, Jun. 2024.

[14] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Berlin, Germany, 2013, pp. 1–10.

[15] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 4569–4574.

[16] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[17] T. Zhang, X. Peng, F. Lin, X. Xiong, and Y. Lou, "Whole-body Compliance Control for Quadruped Manipulator with Actuation Saturation of Joint Torque and Ground Friction," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 11 124–11 131.

[18] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[19] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[22] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox, "Curobo: Parallelized collision-free robot motion generation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8112–8119.

[23] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 645–659, 2018.

[24] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," Jan. 2018.

[25] S. Salvador and P. K. Chan, "Fastdtw: Toward accurate dynamic time warping in linear time and space," 2004. [Online]. Available: https://api.semanticscholar.org/CorpusID:6226669