ISSN: 2664-9640

Trajectory Planning and Obstacle Avoidance Method based on Informed RRT* and Dynamic Movement Primitives

Song Huang*

University of Shanghai for Science and Technology, Shanghai, China

*huang_song686@163.com

Abstract

Addressing the limitations of traditional Dynamic Movement Primitives (DMP) obstacle avoidance methods-which rely on dynamic coupling terms leading to complex algorithmic structures, sluggish responses, and difficulties in autonomous obstacle avoidance-this study innovatively proposes a robotic arm obstacle avoidance trajectory planning framework that integrates DMP with the Informed RRT* algorithm. This framework uses the Informed RRT algorithm to do local trajectory replanning in areas near obstacles. It keeps the main dynamic features of DMP and creates obstacle-avoidance trajectories that are then smoothly added to the original DMP trajectory. This approach eliminates the need for complex dynamic coupling terms, significantly reducing algorithmic complexity. Its local replanning mechanism not only enables autonomous obstacle avoidance and effectively suppresses trajectory oscillations but also substantially enhances real-time response speed. The main contributions of this study include 1) proposing a decoupled DMP obstacle avoidance method based on local trajectory replanning combined with Informed RRT*, simplifying the algorithm structure; 2) the adopted local replanning strategy significantly shortens response time and improves obstacle avoidance efficiency. The effectiveness of the proposed algorithm was validated through two sets of comparative experiments (against the RRT method) conducted on a collaborative robot platform. Experimental results demonstrate that the algorithm reduces response time for obstacle avoidance by 33.2% and increases success rate by 19.5%, fully proving its potential and applicability in specific industrial scenarios.

Keywords

Dynamic Movement Primitives; Informed RRT*; Obstacle Avoidance; Collaborative Robots.

1. Introduction

With the rapid advancement of automation technology, robotic arms-jointed mechanical structures designed to mimic human arm movements-are increasingly applied across diverse fields, including industry, healthcare, and services. Utilizing robotic arms for manufacturing and processing tasks not only improves working conditions but also effectively ensures task efficiency and quality.

Robot programming can be broadly categorized into traditional programming [1] and teach-and-learn programming [2]. Traditional programming enables predefined motions to instruct robots to complete tasks, but it demands high expertise from operators and is difficult to modify. Teach-and-learn programming not only executes predetermined actions but also dynamically adapts to varying work environments or scenarios. Teach-and-learn methods typically extract motion features from single or few taught trajectories and reproduce these features in new scenarios [3]. Ti et al. [4] developed a method combining DMP with GMM/GMR to smooth regression trajectories and generalize to new targets. While it generates smooth trajectories, it

ISSN: 2664-9640

lacks obstacle avoidance capability. Pairet et al. [5] employed hyperquadratic surface envelopes to handle volumetric obstacles, achieving successful avoidance but relying heavily on sensor accuracy. Kulvicius et al. [6] extended DMP by introducing feedback interaction forces for more flexible trajectory adjustment, requiring additional coupling terms that significantly increased algorithmic complexity. Nevertheless, these approaches suffer from slow obstacle avoidance response times and high model complexity. Conventional robotic arm trajectory planning often employs the RRT algorithm. However, the traditional RRT algorithm suffers from low search efficiency in complex scenarios, excessive redundant points, and high trajectory cost.

To address the aforementioned issues, this paper proposes a trajectory planning algorithm that integrates Informed RRT* with DMP. Building upon RRT*, this algorithm first improves the step size determination mechanism by replacing manually set fixed step sizes with global adaptation. Second, it employs a restricted sampling point region strategy to reduce redundant nodes in the search tree by minimizing the scope of repeated areas, thereby enhancing search efficiency. Concurrently, it performs trajectory optimization on the planned route to reduce trajectory cost and trajectory nodes. Furthermore, the algorithm optimizes the collision detection process for identical objects. Finally, its effectiveness is validated through collaborative robot grasping experiments.

2. GMM/GMR Preprocessing

When deploying collaborative robots in industrial settings, multiple teaching passes are often required to ensure task completion success rates and accuracy. Traditional DMP models cannot fit multiple teaching trajectories, necessitating trajectory preprocessing to enhance teaching trajectory quality. This process synthesizes multiple trajectories into a single trajectory for input into the DMP model.

The mathematical formula for the standard Gaussian Mixture Model (GMM) is as follows:

$$P_{gmm}(p_j) = \sum_{i=1}^{K} \alpha_i \, N_i(p_j|\mu_i, \sigma_i) \tag{1}$$

Here, p_j denotes the set of teaching data points; K denotes the number of Gaussian clusters; α_i represents the weight assigned to the i-th Gaussian distribution in the mixture model, i.e., the prior probability of selecting the i-th cluster from all Gaussian clusters, with the sum of probabilities for all selected distributions equaling 1; $N_i(p_j|\mu_i, \sigma_i)$ is the i-th Gaussian distribution; μ_i is the mean of the i-th Gaussian distribution, determining the center position of the Gaussian cluster; σ_i represents the covariance matrix, whose eigenvalues and eigenvectors respectively determine the length and orientation of the major and minor axes of the Gaussian cluster.

The primary parameters to be determined for the GMM include the mean, variance, and weights. In the following section, the Expectation-Maximization (EM) algorithm is employed to solve for this set of parameters as shown below.

$$\theta = \{\alpha_1, \alpha_2, \dots, \alpha_K, \mu_1, \mu_2, \dots \mu_K, \sigma_1, \sigma_2, \dots, \sigma_K\} \tag{2}$$

The steps for establishing GMR are as follows:

Calculate the posterior probability, representing the posterior probability that time t belongs to the i-th Gaussian cluster, as shown in the following equation:

ISSN: 2664-9640

$$P(i|t) = \frac{\alpha_i N(t; \mu_{t,i}, \sigma_{tt,i})}{\sum_{j=1}^{K} \alpha_j N(t; \mu_{t,j}, \sigma_{tt,j})} \tag{3}$$

Here, $\mu_{t,i}$ denotes the temporal mean of the i-th Gaussian cluster; $\sigma_{tt,i}$ denotes the temporal variance; $N(t; \mu_{t,i}, \sigma_{tt,i})$ denotes the temporal marginal distribution of the i-th Gaussian cluster; α_i denotes the weight of the i-th Gaussian cluster.

The final output distribution of GMR predictions is a weighted mixture of the conditional distributions of each Gaussian cluster:

$$P(p|t) = \sum_{i=1}^{K} P(i|t)P(p|t,i) = \sum_{i=1}^{K} P(i|t)N(p; \mu_{p|t,i}, \sigma_{p|t,i}) \tag{4}$$

The predicted value is obtained by taking the expected value of the conditional distribution:

$$\hat{p} = E[p|t] = \sum_{i=1}^{K} P(i|t)\mu_{p|t,i} \tag{5}$$

Here, denotes the mathematical expectation, represents the posterior probability at time t, and is the conditional mean.

As shown in Fig. 1, fitting multiple trajectories using GMM preserves the original teaching trajectory characteristics while enhancing trajectory smoothness.

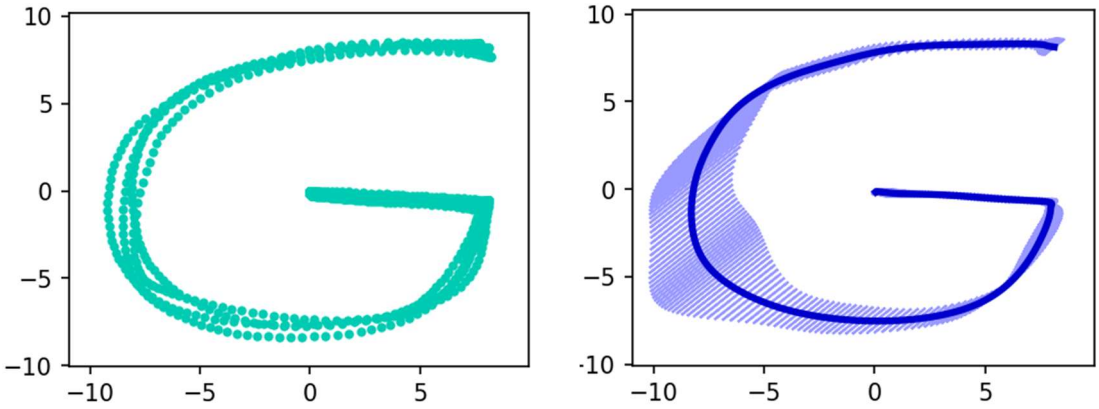


Fig. 1 GMM/GMR Fitting Regression Verification of Multiple Teaching Trajectories

3. Trajectory Learning and Obstacle Avoidance

After preprocessing the teaching trajectory, it is first necessary to effectively encode and learn specific trajectories to ensure precise reproduction of key motion information from the teaching process. For robotic arm operations in complex environments, simple trajectory reproduction often fails to meet obstacle avoidance requirements. Therefore, further correction and optimization of the reproduced trajectory are essential. Specifically, the correction process must account for the end-effector's motion constraints and trajectory safety during actual operation while dynamically adjusting the trajectory to avoid interference and collisions with obstacles. By implementing efficient trajectory correction methods, robotic arms can execute tasks in complex dynamic environments with guaranteed trajectory

ISSN: 2664-9640

feasibility, enhanced operational safety, and improved stability. The key to this process lies in the effective integration of adaptive trajectory correction and obstacle avoidance strategies, ensuring the robotic arm can smoothly and precisely complete tasks within complex environments.

3.1. Demonstration Trajectory Learning

Dynamic Movement Primitives (DMP) is a classical method for generating replay trajectories and control in robotics. DMP can be viewed as the superposition of a PD controller and a trajectory shape learner. The complete formula for the DMP system is as follows:

$$\tau^2\ddot{y} = \alpha_y\left(\beta_y(g-y) - \dot{y}\right) + f(s) \tag{6}$$

Here, y represents the robot's current position, while $\dot{y}$ and $\ddot{y}$ denote the first- and second-order derivatives of y, i.e., velocity and acceleration. g is the target state, and α_y and β_y are the damping coefficients.

The construction of the nonlinear function $f(s)$ requires a time-independent phase variable, which is derived from the regular system:

$$\tau\dot{s} = -\alpha_s s \tag{7}$$

The phase variable s ultimately approaches 0 as the trajectory concludes, eliminating dependence on time and thereby enabling time-independent trajectory generation.

$f(s)$ is composed of a weighted combination of multiple Gaussian basis functions, each corresponding to a distinct phase point. Radial basis functions (RBFs) are used to fit the trajectory's characteristics, with the formula defined as:

$$f(s) = \frac{\sum_{i=1}^{N} w_i\psi_i(s)}{\sum_{i=1}^{N}\psi_i(s)} s(g-y_0) \tag{8}$$

In the equation, $\psi_i(s)$ represents a basis function (such as a Gaussian function), N denotes the number of basis functions, and w_i is the weight parameter. Here, $\psi_i(s)$ denotes the i-th Gaussian function, w_i is the corresponding weight, y_0 is the starting position, and g is the target position. The Gaussian radial basis function formula is as follows:

$$\psi_i(s) = \exp(-h_i(s-c_i)^2) \tag{9}$$

Here,c_i denotes the center of the basis function, uniformly distributed over $s \in [0,1]$, while h_i represents the width of the basis function. These parameters are determined through DMP learning of the taught trajectory.

During the learning phase, the input initially consists of the teaching trajectory $\{y_{\text{demo}}(t), \dot{y}_{\text{demo}}(t), \ddot{y}_{\text{demo}}(t)\}$, representing the displacement, velocity, and acceleration of the teaching trajectory, respectively. Using equation (2), the target force f_{target} is then solved via the inverse dynamics equation:

$$f_{\text{target}} = \tau^2\ddot{y}_{\text{dem}} - \alpha_y\left(\beta_y(g - y_{\text{demo}}) - \tau\dot{y}_{\text{demo}}\right) \tag{10}$$

ISSN: 2664-9640

Next, the weights are fitted using locally weighted regression (LWR) to minimize $\left|f(s) - f_{\text{target}}\right|^2$, thereby solving for w_i in equation (3).

In summary, after learning the preprocessed teaching trajectory, the DMP can generate a reproduction trajectory, as shown in Fig. 2. The reproduction trajectory strictly converges to the target endpoint while preserving shape characteristics, exhibiting smooth motion without oscillation.

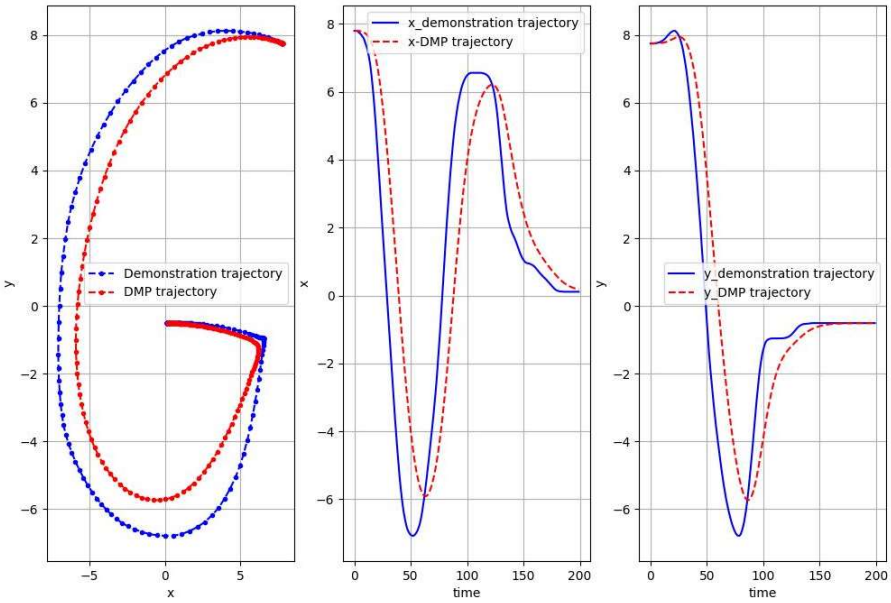


Fig. 2 Comparison of DMP-generated trajectory and trajectory parameters

3.2. Trajectory Correction

In certain complex operating conditions, a single DMP system lacks obstacle avoidance capabilities. This paper introduces the Informed RRT* algorithm, which modifies the local trajectory when the DMP trajectory encounters an obstacle to generate an obstacle-avoidance trajectory. The steps for generating an obstacle-avoidance trajectory using Informed RRT* are as follows:

Step 1: Define the elliptical sampling region

As shown in Fig. 3, the elliptical region typically centers on the starting point x_{start} and the endpoint x_{goal}. The major axis of the ellipse is determined by the current optimal trajectory length c_{best}, while the minor axis is calculated using the formula $\frac{\sqrt{{c_{\text{best}}}^2 - {c_{\min}}^2}}{2}$, where $c_{\min}$ represents the distance from the starting point to the endpoint. After the initial trajectory is generated, the algorithm restricts the sampling area to this ellipse (the black dashed ellipse in the figure). All subsequent sampling points (such as x_{rand}) are located within the ellipse.

Step 2: Node Expansion and Trajectory Generation

In Fig. 3, x_{rand} denotes a randomly sampled target point within the ellipse, x_{nst} represents the node in the current tree closest to x_{rand}, and x_{new} indicates the new node generated by expanding from x_{nst} toward x_{rand}. After generating x_{rand} within the ellipse, locate the nearest tree node x_{nst}. Generate x_{new} along the $x_{\text{nst}} \rightarrow x_{\text{rand}}$ direction using a fixed step size t, then connect x_{nst} and x_{new} to form a new trajectory branch.

Step 3: Parent Node Re-election

ISSN: 2664-9640

In Fig. 3, within the neighborhood (X_{free}) centered at x_{new} with radius r, candidate parent nodes (e.g., x_{pa1}, x_{pa2}) may replace the original parent node x_{nst}. The trajectory cost from all nodes within the neighborhood to x_{new} is computed (e.g., whether the distance from x_{pa1} to x_{new} is shorter than that from x_{pa2} to x_{new}). The node with the lowest cost is selected as the new parent node.

Step 4: Elliptical Dynamic Contraction and Convergence

In Fig. 3, when a better trajectory is detected, the major axis of the ellipse is updated to the new c_{best}, gradually narrowing the sampling area toward the vicinity of the theoretically optimal trajectory. As iterations proceed, the elliptical region progressively focuses around the better trajectory, ultimately generating a smooth trajectory that closely approximates the shortest route (the solid red trajectory in the figure).

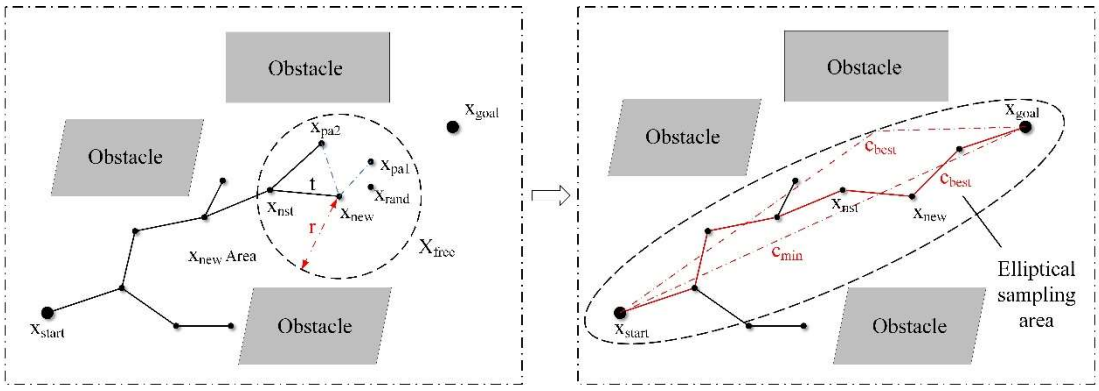


Fig. 3 Schematic Diagram of Informed RRT* Obstacle Avoidance trajectory Generation

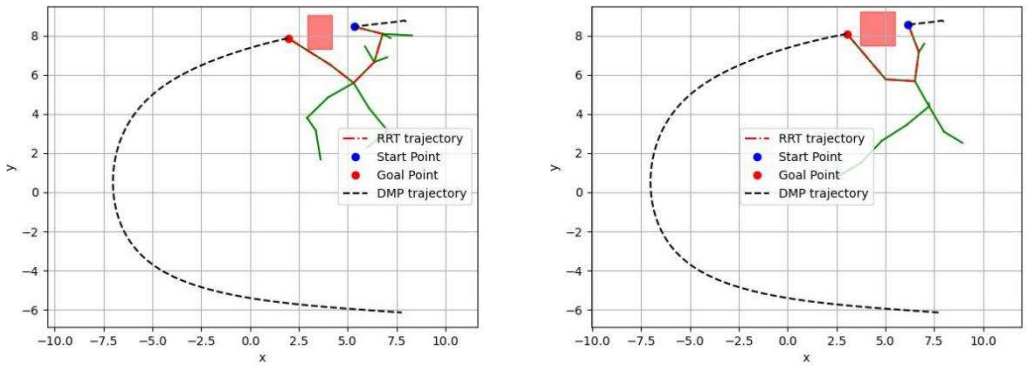
4. Experimental Results

To validate the superiority of the DMP-Informed RRT* model, this paper compares it with a commonly used trajectory correction method based on the RRT algorithm in a simulated environment. The 'G'-shaped trajectory from the handwritten character dataset was employed, simulating a scenario where a square obstacle with dimensions fluctuating between 0 and 2 appears randomly on the trajectory reproduced by the DMP. For the comparison algorithms, the expansion step size was uniformly set to 0.01, the maximum number of nodes was set to 200, and all other corresponding parameters were identical. To prevent direct contact between trajectory points and the obstacle, a collision margin of 0.02 was applied.

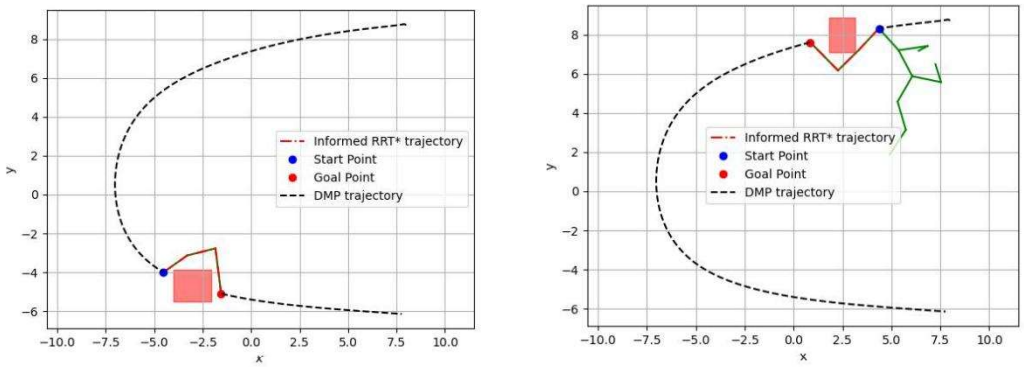
Fig. 4 illustrates the trajectory planning performance of DMP-Informed RRT* and DMP-RRT in this environment, where cuboid blocks represent obstacles. It is evident that due to the blind nature of randomly expanding nodes, the trajectory shape and direction planned by the DMP-RRT algorithm cannot be controlled. This significantly impacts the actual operation of the robotic arm, such as when the planned trajectory exceeds the arm's working range or approaches singular points. In contrast, the corrected trajectory planned using the framework proposed in this paper continuously provides accurate guidance information to the RRT expansion tree by learning from the manually taught DMP model. Consequently, the corrected trajectory remains relatively similar to the original trajectory. Regarding algorithmic efficiency and stability, considering the randomness in RRT tree node selection, trajectory planning from start to goal was repeated 20 times in this obstacle environment. The final computational time comparison is shown in Table 1. The proposed method reduces the average response time by 61.38% and shortens the average trajectory length by 7.48% compared to the DMP-RRT algorithm.

ISSN: 2664-9640

Table 1 demonstrates that other RRT algorithms not only exhibit significantly longer average computation times than the DMP-Informed RRT* method but also suffer from highly unstable computational costs due to the randomness of node expansion. The computational cost of standard RRT algorithms is highly unstable, with significant variation between maximum and minimum execution times. In contrast, the proposed DMP-Informed RRT* framework consistently achieves trajectory planning within a short timeframe across multiple tests, exhibiting relatively stable computational costs. This stability is advantageous for practical applications demanding high real-time performance and flexibility.



(a) DMP-RRT obstacle avoidance trajectory



(b) DMP-Informed RRT* obstacle avoidance trajectory

Fig. 4 Comparison of DMP-Informed RRT* and RRT Obstacle Avoidance Trajectories

Table 1. Comparison of obstacle avoidance trajectory data generated by two algorithms.

Method	Average Response time (s)	Average trajectory Length
DMP- RRT	1.028	47.68
DMP-Informed RRT*	0.637	44.36

5. Conclusion

To address challenges in robotic arm trajectory planning within complex environments, this paper proposes a concise and efficient method for trajectory generation and obstacle avoidance based on teach-in learning. The approach employs GMM/GMR for preprocessing teach-in data and integrates multiple teach-in trajectories, optimizing them while preserving original trajectory features. Combining DMP and Informed RRT models, this paper designs an effective trajectory learning and intelligent obstacle avoidance framework, enhancing the robot arm's

ISSN: 2664-9640

obstacle avoidance capabilities in complex environments. Specifically, the DMP model encodes optimized trajectories to learn motion features and generate replayable trajectories. In obstacle-laden environments, trajectories generated by DMP are refined using target bias and the Informed RRT algorithm. Trajectory points generated by DMP provide guidance information, decomposing trajectory planning into multiple subtasks. This approach ensures trajectory smoothness and continuity while rapidly and stably locating obstacle-avoidance trajectories. Simulation experiments comparing this framework with traditional RRT algorithms validate its advantages in practicality and computational efficiency. Compared to conventional trajectory planning methods, this approach not only reduces operator requirements but also significantly enhances the robotic arm's capability to navigate complex environments, demonstrating significant application and promotional value. Future research will focus on further optimizing robotic arm trajectory planning algorithms to enhance system practicality and flexibility, providing more reliable technical support for robotic arm applications in increasingly complex environments.

References

[1] R. Li, K. Liu, W. Volk, et al., DemoTeach-robot programming with MoCap system for foundry applications, The International Journal of Advanced Manufacturing Technology,140 (9-10) (2025) 1-16.

[2] A.F. Nannim, E.N. Ibezim, M. Mosia, et al., Project-based learning with arduino robots: impact on undergraduate students' achievement and task persistence in robotics programming, Frontiers in Robotics and AI 12 (2025) 1615427-1615427.

[3] M. Babcinschi, F. Cruz, N. Duarte et al., Offline robot programming assisted by task demonstration: an AutomationML interoperable solution for glass adhesive application and welding, International Journal of Computer Integrated Manufacturing 38 (7) (2025) 864-875.

[4] Ti, B., Gao, Y., Li, Q. and Zhao, J. (2019) Dynamic Movement Primitives for Movement Generation Using GMM-GMR Analytical Method. 2019 IEEE 2nd International Conference on Information and Computer Technologies, Kahului, 14-17 March 2019, 250-254.

[5] È. Pairet, P. Ardón, M. Mistry, et al. Learning generalizable coupling terms for obstacle avoidance via low-dimensional geometric descriptors, IEEE Robot. Autom. Let. 4 (2019) 3979-3986.

[6] T. Kulvicius, M. Biehl, M.J. Aein, et al., Interaction learning for dynamic movement primitives used in cooperative robotic tasks, Robot. Auton. Syst. 61 (2013) 1450-1459.