2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Diffusion Model and RRT*-Based Methods for Reflected Mass Optimization in Motion Planning

David Gutierrez-Moreno[1], Simon Armleder[1], Yuhwan Kwon[2], Takumi Hachimine[2], Yoshihisa Tsurumine[2], Takamitsu Matsubara[2], Gordon Cheng[1]

***Abstract*— Optimizing a robot's posture can be advantageous for managing interaction forces with the environment. By optimizing the Reflected Mass (RM) along entire trajectories, the robot's posture can be adjusted to minimize impact forces for safety or maximize them for tasks that require high force, such as pushing or striking. However, the integration of RM optimization within motion planning remains under-explored. To address this, we introduce two new approaches for optimizing RM in motion planning: a probabilistic generative model based on diffusion techniques and a sampling-based method using Rapidly-Exploring Random Trees Star (RRT*). Both methods optimize the RM within the motion planning framework, enabling new strategies for enhancing robot interactions in diverse and dynamic environments. Experimental validation in simulation and on a UR5 robot demonstrates the effectiveness of these approaches in controlling RM, offering promising directions for future research and applications.**

I. INTRODUCTION

As robots are increasingly deployed in scenarios involving close human proximity, their interaction with the environment becomes more complex. Careful motion planning is crucial in domains like healthcare and service robotics to prevent injuries and maintain trust [1]. In these scenarios, optimizing the robot's posture to minimize the impact forces resulting from a potential collision can reduce the risk of injury [2]. Conversely, in applications where robots need to exert significant impact forces, such as pushing heavy objects or delivering controlled impacts, optimizing posture to maximize force output is essential for effective task execution [3]. This need to balance minimizing and maximizing impact forces highlights the importance of strategies that can dynamically regulate a robot's posture based on the specific requirements of the task and environment.

One key factor influencing a robot's posture during interactions is Reflected Mass (RM), which represents the effective mass perceived at the end-effector (EE) in specific directions during impact [4]. By adjusting the robot's posture to manage RM along a trajectory, it is possible to either minimize impact forces for safer interactions or maximize them for tasks that require high impact, such as pushing or striking.

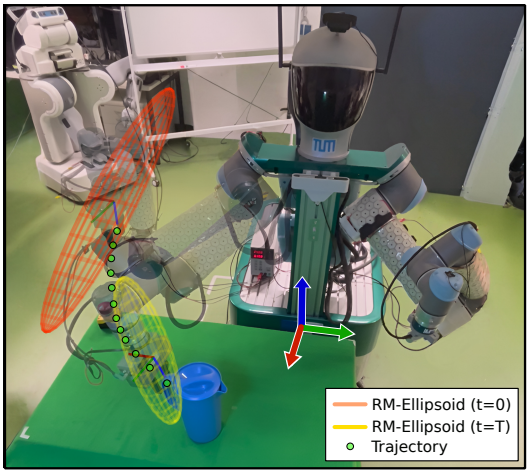

Fig. 1. Optimizing reflected mass along the x-axis while reaching for a bottle on the table.

Previous research has investigated the role of RM and the robot's velocity during a collision to determine the maximum safe velocity a robot can move without causing injury to a human [2], [5]. These methods often do not address the entire trajectory of motion. Control-based approaches are typically reactive, optimizing the robot's dynamics only in response to immediate inputs or disturbances.

In contrast, optimizing RM throughout the entire trajectory in motion planning results in EE movements better suited to ensure safety and performance from the start. This is particularly beneficial in dynamic environments where impact forces must be optimized despite the external object velocities not being controllable, and pre-planned postures are essential for the task.

Contribution: Our work aims to fill a significant gap in the field of robotics by introducing two effective approaches for RM optimization within motion planning. These include a probabilistic generative model based on diffusion techniques and a sampling-based method utilizing Rapidly-Exploring Random Trees Star (RRT*). We validate these approaches experimentally in a simulation study and on a real UR5 industrial robot arm, demonstrating their effectiveness in optimizing RM throughout whole trajectories.

The rest of the paper is organized as follows: Section 2 reviews related work in RM for control and motion planning. Section 3 introduces the theoretical foundations of

[1]D. Gutierrez-Moreno, S. Armleder, and G. Cheng are with the Institute for Cognitive Systems, Department of Electrical and Information Technology, Technical University of Munich (TUM), Munich, Germany {d.gutierrez-moreno, simon.armleder, gordon}@tum.de

[2]Y. Kwon, T. Hachimine, Y. Tsurumine and T. Matsubara are with the Graduate School of Information science, Nara Institute of Science and Technology (NAIST), Nara, Japan {y-kwon, hachimine.takumi.hs8, tsurumine.yoshihisa, takam-m}@is.naist.jp

the algorithms. Section 4 presents the problem formulation. Section 5 includes the experimental validation and discusses the results. Section 6, concludes the paper.

II. Related work

A. RM in control

Research in control-based approaches has used RM in redundant robots to optimize task performance while remaining safe. Haddadin et al. [2] devised a safety framework that supervises the RM perpendicular to the impactor surface's points of interest at each time step. It then scales the velocity down, if necessary, to be within human safe limits. However, the approach does not guarantee an optimized posture over the whole movement. In [5], an algorithm was introduced that employs nullspace motion for posture optimization in redundant robots. This method exploits the robot's additional degrees of freedom to change the robot's configuration to reduce the RM measure at each control step. However, this approach requires knowing the local minimum of the optimized posture at a certain Cartesian pose. In addition, they optimize the posture on each trajectory segment locally so that the whole trajectory is not jointly optimized. Recently, Khurana et al. [6] developed a motion planner that frames the robot task as a dynamical system. It then utilizes an inverse kinematics (IK) controller and additional optimization-based controllers to regulate the RM for object pushing or hitting.

The methods above adapt RM online for redundant robots without considering optimizing it across the entire trajectory during the planning phase.

B. RM in motion planning

Robot motion planning problems traditionally focus on generating collision-free trajectories while often optimizing for primary objectives such as shortest path or smoothness. Integrating RM into motion planning has been explored in limited capacity. A method for incorporating RM into motion planning was introduced in [7]. In this approach, RM is utilized to guide a graph search algorithm in finding a safe time-optimal trajectory in the discretized robot's workspace. However, the primary focus is on maximizing the robot's velocity, rather than optimizing RM across the entire path.

III. Preliminaries

This section provides a brief overview of the Rapidly-Exploring Random Trees Star (RRT*) algorithm, diffusion models (DM), and the definition of RM in robotic systems.

A. RRT*

Rapidly-exploring Random Trees (RRT) [8] are a popular algorithm class for path planning. The original RRT algorithm does not guarantee convergence to optimal paths. RRT* [9] enhances the RRT algorithm by incorporating a rewiring step that continually improves the tree structure, leading to an asymptotically optimal solution. This improvement allows RRT* to find paths that are not only feasible but also optimal in terms of a given cost function, making it a powerful tool for complex path planning problems.

B. Diffusion Models

Diffusion models are state-of-the-art generative models known for their ability to fit complex, multimodal distributions with robust sampling performance [10]. In this work, we utilize diffusion models to fit a complex distribution of robot trajectories within its workspace.

Let a $\boldsymbol{\tau}$ denote a random variable for a sequence of robot joint states $(\boldsymbol{\theta}_j)_{j=1}^{H}$ over a horizon H. Let $\mathcal{D} := \{\boldsymbol{\tau}^{(i)}\}_{i=0}^{D}$ be a dataset of trajectories from an unknown distribution, such that $\boldsymbol{\tau}^{(i)} \sim \mathrm{q}(\boldsymbol{\tau})$. Diffusion models can approximate $\mathrm{q}(\boldsymbol{\tau})$ by learning a parameterized model $\mathrm{p}_\theta(\boldsymbol{\tau})$ through a Markov chain of T diffusion steps. To sample from $\mathrm{p}_\theta(\boldsymbol{\tau})$, diffusion models employ a reversed Markov chain of denoising steps.

1) Forward diffusion process: A sample $\boldsymbol{\tau}_0^{(i)} \sim \mathrm{q}(\boldsymbol{\tau})$ is perturbed by Gaussian noise $\boldsymbol{\epsilon}_t$ at each step t, transforming the sample into a multivariate standard Gaussian distribution after T diffusion steps, $\boldsymbol{\tau}_T^{(i)} \sim \mathcal{N}(\mathbf{0}|\boldsymbol{I})$. This is called the forward diffusion process, $\boldsymbol{\tau}_0^{(i)} \xrightarrow{\boldsymbol{\epsilon}_1} \boldsymbol{\tau}_1^{(i)} \xrightarrow{\boldsymbol{\epsilon}_2} \boldsymbol{\tau}_2^{(i)} \ldots \xrightarrow{\boldsymbol{\epsilon}_T} \boldsymbol{\tau}_T^{(i)}$. We employ a Markov diffusion kernel[1] defined as

$$\mathrm{q}(\boldsymbol{\tau}_t|\boldsymbol{\tau}_{t-1}) = \mathcal{N}(\boldsymbol{\tau}_t|\sqrt{1-\beta_t}\boldsymbol{\tau}_{t-1}, \beta_t\mathbf{I}), \tag{1}$$

where the parameter β_t is the diffusion rate chosen by a variance scheduler [11]. In this work we use a cosine schedule. The noisy samples obtained are then used to train a neural network model to predict the noise added to the data point at each time step.

2) Reverse denoising process: The goal is to reconstruct the original information by denoising a Gaussian sample $\boldsymbol{\tau}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. When β_t is sufficiently small, the functional form of the Markov diffusion kernel is preserved; that is, each transition can be approximated as some data point centered at $\boldsymbol{\mu}_t$, slightly perturbed by small noise [10]. This allows to define the reverse Markov transition kernel also as Gaussian. Thus, the entire reverse denoising process can be computed utilizing a parameterized model $\mathrm{p}_\theta(\boldsymbol{\tau}_{t-1}|\boldsymbol{\tau}_t)$ as the reverse Markov transition kernel such that

$$\mathrm{p}_\theta(\boldsymbol{\tau}_{t-1}|\boldsymbol{\tau}_t) = \mathcal{N}(\boldsymbol{\tau}_{t-1}|\boldsymbol{\mu}_\theta(\boldsymbol{\tau}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{\tau}_t, t)), \tag{2}$$

where the mean function $\boldsymbol{\mu}_\theta(\boldsymbol{\tau}_t, t)$ can be written with respect to the current diffusion step and a predicted noise

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\boldsymbol{\tau}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_t, t)\Big). \tag{3}$$

3) Training: The noise of the mean function in Eq. 3 can be learned using the simplified objective from [12],

$$L(\theta) := \mathbb{E}_{t,\boldsymbol{\tau},\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\tau} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right]. \tag{4}$$

4) Guided sampling: To obtain trajectories from $\mathrm{p}_\theta(\boldsymbol{\tau})$ that meet specific goals or minimize a predefined cost, we use classifier guidance [13]. Here, the gradient of the log-likelihood of the desired objective is used to modify the reverse Markov transition kernel. The conditioned Markov kernel for guided sampling is defined as:

$$\mathrm{p}_\theta(\boldsymbol{\tau}_t|\boldsymbol{\tau}_{t+1}, \boldsymbol{o}) = \mathcal{N}(\boldsymbol{\tau}_t|\boldsymbol{\mu}_{t+1} + \boldsymbol{\Sigma}\boldsymbol{g}, \boldsymbol{\Sigma}), \tag{5}$$

[1]Note that we dropped the superscript i denoting the i-th sample of the distribution to avoid cluttering notation

where $\mathbf{o}$ is a random variable denoting the fulfillment of task-specific goals or costs, and the guidance term $\boldsymbol{g}$ is defined as $\boldsymbol{g} = \nabla_{\boldsymbol{\tau}_{t+1}} \log \mathrm{p}_\theta(\boldsymbol{o}|\boldsymbol{\tau}_{t+1})|_{\boldsymbol{\tau}_{t+1}=\boldsymbol{\mu}_{t+1}}$. $\boldsymbol{\mu}_{t+1}$ denotes the mean function of the reverse transition kernel in (2) at step $t+1$, and $\boldsymbol{\Sigma}$ is its covariance.

C. Reflected Mass (RM)

The equations of motion of a rigid body manipulator can be written in the following matrix vector form

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{g}(\boldsymbol{\theta}) = \boldsymbol{\tau}, \tag{6}$$

where $\boldsymbol{\theta} \in \mathbb{R}^n$ is the robot's generalized coordinates, $\mathbf{M}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n}$ is the configuration dependent inertia matrix, $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \in \mathbb{R}^n$ is the Coriolis and centrifugal joint torques vector, and $\mathbf{g}(\boldsymbol{\theta}) \in \mathbb{R}^n$ the gravity joint torque vector. $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of generalized joint torques.

Reflected mass, in the context of robot manipulation, refers to the effective mass perceived at the robot's end effector when colliding with an external object, e.g., a human [4]. It can be computed from the Cartesian mass matrix, derived from the definition of kinetic energy as shown by [4]. Consequently, we can write the Cartesian mass matrix as follows,

$$\boldsymbol{\Lambda}(\boldsymbol{\theta}) = \left(\mathbf{J}(\boldsymbol{\theta})\mathbf{M}(\boldsymbol{\theta})\mathbf{J}^T(\boldsymbol{\theta})\right)^{-1}, \tag{7}$$

where $\mathbf{J}(\boldsymbol{\theta}) \in \mathbb{R}^{6\times n}$ is the Jacobian matrix that maps joint velocities to the EE velocities. As shown in [4], the Cartesian mass matrix can be decomposed into a linear v and angular w component,

$$\boldsymbol{\Lambda}^{-1}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{\Lambda}_v^{-1}(\boldsymbol{\theta}) & \boldsymbol{\Lambda}_{wv}^{-1}(\boldsymbol{\theta}) \\ \boldsymbol{\Lambda}_{vw}^{-1}(\boldsymbol{\theta}) & \boldsymbol{\Lambda}_w^{-1}(\boldsymbol{\theta}) \end{bmatrix}. \tag{8}$$

Finally, given a unit direction $\mathbf{u} \in \mathbb{R}^3$ of potential collision the RM at configuration $\boldsymbol{\theta}$ can be written as,

$$m_u(\boldsymbol{\theta}) = \left(\mathbf{u}^T \boldsymbol{\Lambda}_v^{-1}(\boldsymbol{\theta})\mathbf{u}\right)^{-1}, \tag{9}$$

where $m_u(\boldsymbol{\theta}) \in \mathbb{R}$ represents the RM of the EE in the direction $\mathbf{u}$ of collision.

IV. Problem Formulation

In this work, we aim to plan trajectories that reach a desired EE pose while optimizing RM in a given direction $\mathbf{u}$. Additionally, the algorithm should achieve it independently of the robot's redundancy resolution capabilities. To address this, we propose two distinct motion planning algorithms.

A. RRT*-RM

We present a novel adaptation of the RRT* algorithm specifically designed for RM optimization. Drawing inspiration from [14], our method incorporates a modified cost function to prioritize paths within the tree that achieve desired RM values. The cost function is defined as:

$$C(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}_{\text{new}}) = \beta C_{\text{d}}(\boldsymbol{x}_{\text{new}}) + (1-\beta) C_{\text{RM}}(\boldsymbol{\theta}_{\text{new}}), \tag{10}$$

where $C_{\text{d}} : \mathbb{R}^3 \to \mathbb{R}$ represents the accumulated euclidean distance cost function and $C_{\text{RM}} : \mathbb{R}^n \to \mathbb{R}$ denotes the average RM cost function up to the current explored node. The parameter β is a weighting factor that balances the contributions of both terms, with $\beta = 0.5$ in our study. To enforce RM reduction at each Cartesian space node of the tree, we employ a closed-loop inverse kinematic solver (CLIK). Following the approach in [15], we use the robot's Jacobian pseudo-inverse and its nullspace projector to generate the desired configurations. Since the UR5 is a non-redundant robot, we truncate the last three rows (orientation part) of $\mathbf{J}(\boldsymbol{\theta}) \in \mathbb{R}^{6\times n}$. By removing the angular Jacobian $\mathbf{J}_w$, we obtain a three-dimensional null space $\mathcal{N}(\mathbf{J}_v)$ onto which the objective function can be projected using the nullspace projector $\mathbf{N} = \mathbf{I} - \mathbf{J}^\dagger\mathbf{J}$, where $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix, and $\mathbf{J}^\dagger \in \mathbb{R}^{n\times 3}$ is the right pseudoinverse. Using differential kinematics, the updated joint configuration $\boldsymbol{\theta}_{new}$ can be obtained via forward Euler integration of the joint space velocity

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^\dagger \boldsymbol{e}_x + \mathbf{N}\dot{\mathbf{g}}_0, \tag{11}$$

where $\boldsymbol{e}_x = \boldsymbol{x}_{new} - \boldsymbol{x}$ is the Cartesian position error. The second term denotes the projection of the secondary joint velocity $\dot{\mathbf{g}}_0$ onto $\mathcal{N}(\mathbf{J}_v)$, which is determined by the negative gradient of the RM equation

$$\dot{\mathbf{g}}_0 = -\alpha \left(\frac{\partial m_u(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right)^T \in \mathbb{R}^n, \tag{12}$$

where α is step size for the gradient descent.

The RRT*-RM solution's smoothness is enhanced by fitting cubic splines through the Cartesian nodes and a trajectory is created with a trapezoidal velocity profile.

B. Diffusion model with RM optimization (DM-RM)

We train a diffusion model, denoted as $\mathrm{p}_\theta(\boldsymbol{\tau})$, on a dataset $\mathcal{O}$ composed of robot trajectories represented in joint coordinates.

The neural network architecture employed for training the noise predictor $\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_t, t)$ is a Temporal U-Net, as described in [16].

1) Initial robot configuration $\boldsymbol{\theta}_{init}$: To ensure that the sampled trajectories are consistent with the initial robot configuration $\boldsymbol{\theta}_{init}$, we use a Dirac delta function to guide the denoising process defined as

$$\delta_{\boldsymbol{\theta}_{init}}(\boldsymbol{\tau}) = \begin{cases} -\infty, & \text{if } \boldsymbol{\theta}_0 = \boldsymbol{\theta}_{init} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where $\delta_{\boldsymbol{\theta}_{init}}(\boldsymbol{\tau})$ imposes a cost of $-\infty$ when the state $\boldsymbol{\theta}_0$ matches the initial configuration $\boldsymbol{\theta}_{init}$, and zero otherwise. The function is implemented by directly enforcing the initial robot joint configuration on the $\boldsymbol{\tau}$.

2) RM optimization over the trajectory horizon: To optimize the RM during trajectory generation, we guide the denoising process towards regions of the learned distribution corresponding to trajectories with optimized RM. To achieve that we define the following function,

$$J_{RM}(\boldsymbol{\tau}) = \sum_i^H m_u(\boldsymbol{\theta}_i), \tag{14}$$

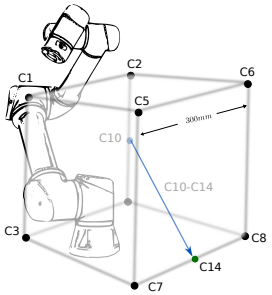

Fig. 2. Performance test cube as per ISO 9283 Standard [17] with points of interest (C1 to C14).

where $J_{RM}(\boldsymbol{\tau})$ represents the cumulative RM along a specified direction $\mathbf{u}$, as introduced in section III-C over the horizon H.

To further condition the sampling process to generate trajectories that bring the robot's EE to a desired goal pose, we define the following likelihood function,

$$J_{Goal}(\boldsymbol{\tau}) = \sum_{i=H-h}^{H} \gamma_1 ||\boldsymbol{x}_{goal} - f(\boldsymbol{\theta}_i)||_2 + \gamma_2 ||\text{LogMap}(\boldsymbol{Q}_i * \boldsymbol{Q}_{goal}^{-1})||_2, \quad (15)$$

where the $\boldsymbol{x}_{goal} \in \mathbb{R}^3$ is the Cartesian goal position vector, $f : \mathbb{R}^n \to \mathbb{R}^3$ is the forward kinematics function that maps the robot's joint angles to the EE's position, and $\boldsymbol{Q}_i$ and $\boldsymbol{Q}_{goal}$ are the EE's unit quaternions at horizon step i and at the goal, respectively. The LogMap operation maps elements of the SU(2) manifold to $\mathfrak{so}(3)$ Lie-algebra vectors. The parameters $\gamma_1, \gamma_2 \in \mathbb{R}$ are weighting factors to balance the position and orientation cost. Finally, we use the last 5 time steps to attract the robot towards the goal position ($h = 5$).

3) Trajectory generation: The gradient of the likelihood functions is computed and utilized to shift the mean value $\boldsymbol{\mu}_t$ at each time diffusion step. Therefore, we can directly denote the gradient of the log-likelihood as

$$\nabla_{\boldsymbol{\tau}_t} \log\, \mathrm{p}_\theta(\boldsymbol{o}|\boldsymbol{\tau}_t) = \lambda_1 \nabla_{\boldsymbol{\tau}_t} J_{RM}(\boldsymbol{\tau}_t) + \lambda_2 \nabla_{\boldsymbol{\tau}_t} J_{Goal}(\boldsymbol{\tau}_t), \quad (16)$$

where the parameters λ_1, λ_2 are the importance weights or likelihood temperatures. Algorithm 1 summarizes the process of generating optimized RM trajectories using guided denoising.

Algorithm 1 Guided diffusion

Input: Pre-trained noise predictor $\boldsymbol{\epsilon}_\theta$, start state ($\theta_{init}$, temperatures λ_i, scheduling terms $(\alpha_t, \bar{\alpha}_t, \sigma_t)$, Batch size B.

Sample a batch of trajectories
$\mathcal{T}_N = \{\boldsymbol{\tau}_N^{(j)}\}_{j=1}^{B} \sim \mathcal{N}(0, I)$

Hard set start state (apply $\delta_{\boldsymbol{\theta}_{init}}(\boldsymbol{\tau}_N)$)
$\boldsymbol{\theta}_{i=0} \leftarrow \boldsymbol{\theta}_{init}$

for $t = N, \ldots, 1$ **do**
  Compute the diffusion step mean
  $\boldsymbol{\mu}_t = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{\tau}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_t, t) \right)$
  Compute weighted gradient of costs
  $\boldsymbol{g} = -\lambda_1 \nabla_{\boldsymbol{\tau}} J_{RM}(\boldsymbol{\tau}_t) - \lambda_2 \nabla_{\boldsymbol{\tau}} J_{Goal}(\boldsymbol{\tau}_t)$
  Sample from the diffusion step posterior
  $\boldsymbol{\tau}_{t-1} = \boldsymbol{\mu}_t + \boldsymbol{g} + \sigma_t \boldsymbol{z}, \quad \boldsymbol{z} \sim \mathcal{N}(0, I)$
  Hard set start state (apply $\delta_{\boldsymbol{\theta}_{init}}(\boldsymbol{\tau}_{t-1})$)
  $\boldsymbol{\theta}_{i=0} \leftarrow \boldsymbol{\theta}_{init}$
end for

Compute the MAP estimate
$\boldsymbol{\tau}_0^* = \arg\max_{\boldsymbol{\tau}_0 \in \mathcal{T}_0} \mathrm{p}(\boldsymbol{\tau}_0|\boldsymbol{o})$

Output: Probabilistically optimal trajectory $\boldsymbol{\tau}_0^*$

V. Experimental Validation

This section presents the validation experiments conducted with the RRT*-RM and DM-RM algorithms. It begins with a description of the simulation used to evaluate the proposed methods, followed by an explanation of the data generation process for training the diffusion model. Next, the implementation and experiment description are introduced. Finally, the results of each method are analyzed and discussed.

A. Simulation

We aim to study the application of the proposed methods to a non-redundant robot. A simulation environment was developed using CoppeliaSim (formerly V-REP) to emulate a service robot scenario. A UR5 industrial robot arm is mounted on a workbench with a cup and a human hand, where the robot needs to reach a target position while minimizing RM in desired directions.

B. Dataset

The standard ISO 9283 [17] was used to define a cubical volume inside the reachable robot's workspace to generate standardized trajectories for training the generative model along predefined Cartesian point pairs within the cube. The chosen side length of the test cube for our experiments was $300\,mm$. Figure 2 depicts examples of reference points chosen for trajectory generation.

Using cubic splines, we recorded 50,000 joint state trajectories generated by CoppeliaSim's inverse kinematics solver between these randomly selected point pairs. Moreover, the orientation and position of the spline's support points were randomly varied to achieve an extensive coverage of the robot's configuration space. We aimed to produce such a high number of trajectory observations to create a sufficiently rich dataset for training the diffusion model. The Diffusion model was trained using 25 diffusion steps and a horizon $H = 48$.

C. Implementation

Both methods were implemented in Python 3.8 on a workstation with an 8-core Intel i7 processor running at 4.80 GHz and an NVIDIA RTX A5000 graphics card. The diffusion model is based on the work of [16] and [18]. The robot's forward kinematic functions and mass matrix were implemented using the PyTorch framework [19]. The dynamics parameters utilized were extracted from the robot's URDF. This approach enabled parallel computation on the

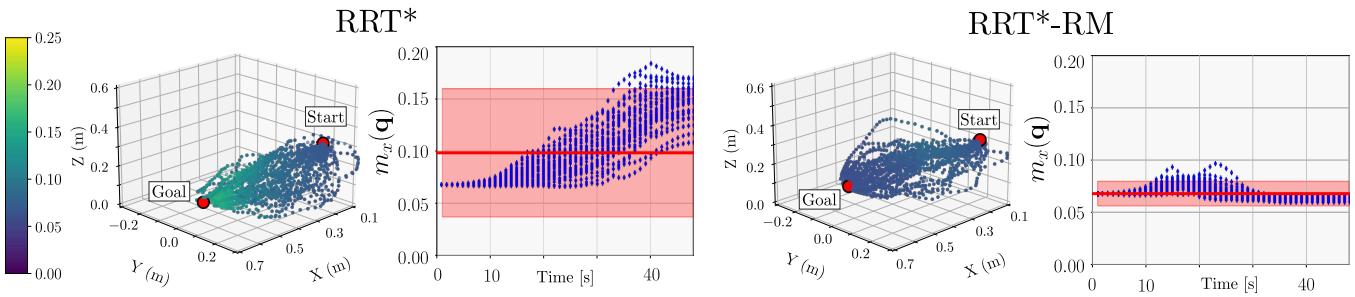

Fig. 3. **RRT*-based trajectory planning:** Trajectories and RM values obtained by RRT* (left side) and RRT*-RM (right side) along the x-axis of the C10-C14 path. RRT*-RM reduces cumulative RM by 31% compared to RRT*. The red line denotes the mean RM value and the shaded area the ± 2 standard deviation.

GPU and facilitated automatic differentiation, allowing for efficient gradient calculations of the guiding functions during each denoising step across a batch of trajectories. For the RRT* case, we used a CaSaDi-based pinocchio framework to compute the gradients of the reflected mass function.

D. Setup

We selected random pairs of points of interest from the ISO-Cube as initial and goal poses to evaluate the proposed methods. For each pair, 100 trajectories were generated using the RRT*-RM and DM-RM methods. In these experiments, the orientation term in Equation 15 was omitted, allowing the methods to optimize the goal orientation freely. Due to space limitations, we present only the study of trajectories from C10 to C14. This path requires the EE to primarily rotate and extend along the x-direction. We compared the trajectory distributions optimized for RM along the robot's base x-axis. This simplification facilitates testing and simulates scenarios where potential collisions do not align with the robot's primary direction of motion.

E. RRT*-RM

Figure 3 illustrates the trajectory distributions for RRT*-based methods in Cartesian space. The RRT*-RM method (right) significantly reduces RM values compared to the RRT* (left). RRT*-RM reduces the cumulative RM by 31%.

To better understand the distribution of RM values and the location of high RM values, Fig. 3 depicts beside the Cartesian plots the RM values over each time step of the trajectory. It is evident that, in the standard RRT*, there is a noticeable upward trend in RM values, particularly between time steps 10 and 30. This period corresponds to the transition phase, where the robot shifts from rotating to extending its arm. During this phase, the robot's configuration changes dynamically, increasing RM. However, RRT*-RM effectively minimizes this transition's impact, maintaining more consistent and lower RM values throughout the trajectory.

F. DM-RM

We sample batches $B = 64$ of Standard normal noise distributions to initialize the denoising processes. We generate 100 trajectories with DM-RM and the unguided variant. In Fig. 4 the Cartesian trajectories generated for the X-Directional extension for the unguided Diffusion model (left) and DM-RM with respect to the robot's base X direction (bottom) are depicted. The RM values in the unguided case are overall higher than in the RRT* case. DM-RM reduces the RM values by 38%.

To investigate the origin of the high RM values, we graph them at each time step in Fig.4 beside the Cartesian plots. It is observed that the majority of the RM values lie within the shaded region around the solid red line, which denote the mean value and the ± 2 standard deviations. The concentration of outliers towards the end of the trajectory is attributed to the influence of the goal cost $J_{Goal}(\boldsymbol{\tau}_t)$ in Eq. 15, which acts on the set of last five points near the goal. In this region, the effects of the RM gradients during sampling are diminished as the priority shifts towards minimizing the distance to the goal rather than RM. This reflects a necessary trade-off, where the importance of accurately reaching the goal position outweighs the RM minimization efforts in the final steps. Table I summarizes the results obtained for both methods[2]. The results show that RRT*-RM outperforms DM-RM in RM values, as expected, due to its greater flexibility in following gradients since DM-RM is constrained by its training data. On the other hand, DM-RM leverages the trajectory smoothness from the training data.

G. Real robot experiment

We further validate our methods on a mobile manipulator for a reaching task. For simplicity, only the right arm of the robot is modeled and trained. The results are demonstrated in the accompanying video.

Method	RM (Kg, min-max)	Cumulative RM (Kg, min-max)
RRT*	0.098 ± 0.032 $(0.064, 0.195)$	4.72 ± 0.42 $(3.86, 5.80)$
RRT*-RM	0.067 ± 0.006 $(0.060, 0.096)$	3.26 ± 0.05 $(3.16, 3.39)$
Unguided DM	0.160 ± 0.096 $(0.066, 0.875)$	7.70 ± 2.09 $(4.61, 17.51)$
DM-RM	0.100 ± 0.037 $(0.066, 0.278)$	4.79 ± 0.32 $(4.09, 5.42)$

TABLE I. RM and cumulative RM values with ranges.

[2]The table values are based on robot parameters from a publicly available URDF and are for visualization purposes, as they do not reflect the actual robot. RM is evaluated at the EE but can also be assessed at other points using the Jacobians in Eq. 7.

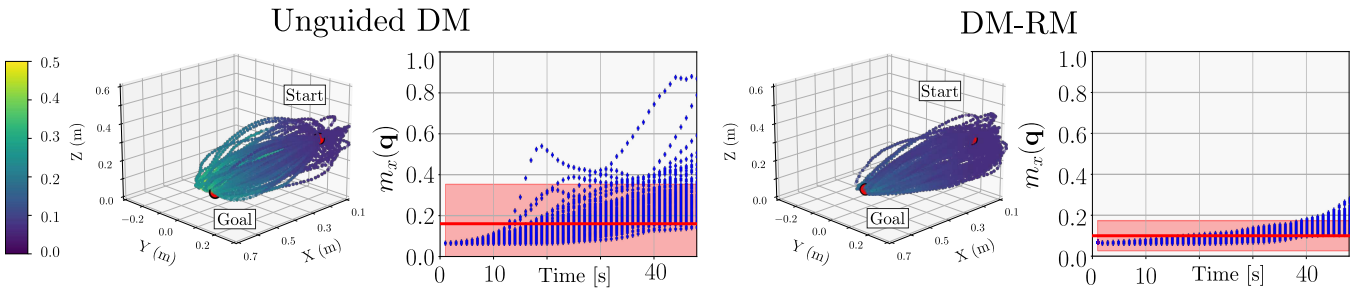

Fig. 4. **Diffusion model trajectory planning**. Trajectories and RM values obtained by the unguided DM (left side) and the DM-RM (right side) along C10-C14 path. DM-RM reduces cumulative RM by 38% compared to the unguided DM. The red line denotes the mean RM value and the shaded area the ± 2 standard deviation.

VI. CONCLUSION

This work addresses the gap in optimizing RM along entire trajectories via motion planning by employing two distinct approaches: a diffusion model and an RRT*-based method. Moreover, the proposed algorithms work on redundant and non-redundant robots. In our study, both methods successfully reduced RM compared to unoptimized baselines. RRT*-RM reduces RM by 31% and DM-RM by 38%. Experiments on a UR5 industrial robot further validated these findings in real-world scenarios.

In future work, we plan to address the DM-RM's tendency to prioritize reaching the goal over the cumulative RM optimization near the end of the trajectories. This limitation could be mitigated by improving the training data or incorporating explicit RM constraints toward the horizon's end during inference. Additionally, while the RRT*-RM method effectively reduces RM, it produces non-smooth paths typical of sampling-based methods. Future work could explore post-processing techniques and hybrid approaches to improve path smoothness. Comparative studies between motion planning and control methods for RM optimization would be valuable, as control methods operate locally, providing additional insights into effectiveness and applicability.

ACKNOWLEDGMENT

The work of David Gutierrez-Moreno was supported by the Lothar-und-Sigrid-Rohde-Stiftung and the Japan Student Services Organization (JASSO).

REFERENCES

[1] P. A. Lasota, T. Fong, and J. A. Shah, "A survey of methods for safe human-robot interaction," Foundations and Trends® in Robotics, vol. 5, no. 2, pp. 261–349, 2017.

[2] S. Haddadin, S. Haddadin, A. Khoury, T. Rokahr, S. Parusel, R. Burgkart, A. Bicchi, and A. Albu-Schaeffer, "A truly safely moving robot has to know what injury it may cause," in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 5406–5413.

[3] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing," Frontiers in Robotics and AI, vol. 7, 2020.

[4] O. Khatib, "Inertial properties in robotic manipulation: An object-level framework," The International Journal of Robotics Research, vol. 14, no. 1, pp. 19–36, 1995.

[5] N. Mansfeld, B. Djellab, J. R. Veuthey, F. Beck, C. Ott, and S. Haddadin, "Improving the performance of biomechanically safe velocity control for redundant robots through reflected mass minimization," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 5390–5397.

[6] H. Khurana and A. Billard, "Motion planning and inertia-based control for impact aware manipulation," IEEE Transactions on Robotics, vol. 40, pp. 2201–2216, 2024.

[7] R. Laha, W. Wu, R. Sun, N. Mansfeld, L. F. Figueredo, and S. Haddadin, "S*: On safe and time efficient robot motion planning," in 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 12 758–12 764.

[8] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164), vol. 1. IEEE, 2001, pp. 473–479.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The international journal of robotics research, vol. 30, no. 7, pp. 846–894, 2011.

[10] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ser. ICML'15. JMLR.org, 2015, p. 2256–2265.

[11] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in Proceedings of the 38th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8162–8171.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in Proceedings of the 34th International Conference on Neural Information Processing Systems, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[13] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in Proceedings of the 35th International Conference on Neural Information Processing Systems, ser. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2024.

[14] T. Pardi, V. Ortenzi, C. Fairbairn, T. Pipe, A. M. G. Esfahani, and R. Stolkin, "Planning maximum-manipulability cutting paths," IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 1999–2006, 2020.

[15] H. Shen, W.-F. Xie, J. Tang, and T. Zhou, "Adaptive manipulability-based path planning strategy for industrial robot manipulators," IEEE/ASME Transactions on Mechatronics, vol. 28, no. 3, pp. 1742–1753, 2023.

[16] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in International Conference on Machine Learning, 2022.

[17] I. 9283, "Manipulating industrial robots - performance criteria and related test methods." 1998.

[18] J. Carvalho, A. Le, M. Baierl, D. Koert, and J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.