2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

SafeFlowMPC: Predictive and Safe Trajectory Planning for Robot Manipulators with Learning-based Policies

Thies Oelerich[1], Gerald Ebmer[1], Christian Hartl-Nesic[1], Andreas Kugi[1,2]

***Abstract*— The emerging integration of robots into everyday life brings several major challenges. Compared to classical industrial applications, more flexibility is needed in combination with real-time reactivity. Learning-based methods can train powerful policies based on demonstrated trajectories, such that the robot generalizes a task to similar situations. However, these black-box models lack interpretability and rigorous safety guarantees. Optimization-based methods provide these guarantees but lack the required flexibility and generalization capabilities. This work proposes SafeFlowMPC, a combination of flow matching and online optimization to combine the strengths of learning and optimization. This method guarantees safety at all times and is designed to meet the demands of real-time execution by using a suboptimal model-predictive control formulation. SafeFlowMPC achieves strong performance in three real-world experiments on a KUKA 7-DoF manipulator, namely two grasping experiment and a dynamic human-robot object handover experiment. A video of the experiments is available at https://www.acin.tuwien.ac.at/en/42d6. The code is available at https://github.com/TU-Wien-ACIN-CDS/SafeFlowMPC.**

I. INTRODUCTION

Trajectory planning for robot manipulators is inherently difficult due to multiple factors. Firstly, the kinematics are non-linear, which leads to multiple solutions for a given task, and have singularities and physical limits that must be considered [1]. Secondly, the environment in which the robot acts in may change during operation, requiring online adaptions of the motion during the operation. This might happen due to a change of the task initiated by an operator [2], moving obstacles in the scene [3], [4], or other actors in the environment [5], [6]. Thirdly, many task objectives for robot motions are hard to encode in numerical objective functions. Hence, existing optimization algorithms are difficult to apply. Examples include human-robot object handovers [5], [7], [8], cleaning of sinks [9], and generalizability to different environments [10]. Lastly, physical interaction with the world requires safety considerations for actors in the scene, other objects, and the robot's hardware. The robot must not collide with obstacles to avoid physical damage to the environment and itself.

Many solutions exist to tackle motion planning in such challenging environments. Notably, global optimization-based and sampling-based planners [11], [12] exist to plan a trajectory for the entire task with constraints. These require a numerical objective and are not capable to adapt the robot motion in real time, which is necessary to react to changes in the environment. To improve the computational efficiency, finite-horizon planning, e.g., using model-predictive control strategies [13], [14] or sampling-based approaches [15], is employed. The improved computational efficiency comes at the cost of degraded optimality of the solution but enables reactive behavior in real-time to changes in the environments. However, these approaches need well-posed problem formulations, i.e., a numerical reward must be designed for the task, which is often complicated. An alternative are learning-based methods [16], [17], [18], which enable fast inference times to plan motions online. The design of numerical rewards can be avoided by learning from a dataset of demonstrations [19], [20]. These demonstrations encode the desired behavior in diverse scenarios and the learning agent learns generalized behavior in similar situations. The downside of these approaches is the lack of systematic safety considerations as the learning agent is often modeled as a black box.

Flow matching [21] and diffusion models [16] have recently shown promising results in robot trajectory planning. Instead of learning the distribution over desired trajectories, these methods learn probability paths from a source distribution to the target distribution. This involves an iterative procedure during inference to create trajectories of the target distribution. During these iterative steps, the intermediate trajectories can be adapted to bias the model towards a desired behavior, e.g., enforcing safety. Current approaches include solving an optimization problem [22], using control barrier functions [23], and using cost guidance [17]. We extend this work by tailoring it further toward robot manipulators. As non-convex optimization [22] compromises real-time capabilitiy as it my exhibit significant variety in terms of computation time. Cost guidance with gradients [17] is unreliable as it cannot guarantee constraint satisfaction and does not scale well to complex environments and models. Control barrier functions [23] are difficult to design and therefore limit the performance of the system.

In real-world applications safety is critical and needs to be systematically enforced. For example, safety filters [24], [25], [26] are employed to project a potentially unsafe input signal into a set of safe inputs. This is generally possible for any kind of learning policy but has the disadvantage that the system behavior changes, which alters the state distribution and, thus, deteriorates performance of the policy. Therefore, it is advantageous to incorporate the safety filtering more deeply with the agent. The work in [27] includes the safety consideration directly in the learning process, but

[1]All authors are with the Automation and Control Institute (ACIN), TU Wien, Vienna, Austria, {oelerich, ebmer, hartl, kugi}@acin.tuwien.ac.at
[2]Andreas Kugi is with the center for Vision, Automation & Control, AIT Austrian Institute of Technology GmbH, Vienna, Austria andreas.kugi@ait.ac.at

this approach does not scale well to predictive planning. In [28], the authors use Hamilton-Jacobi reachability [29] to improve safety. A one-step QP-projection is used to create a safe reinforcement learning policy update in [30]. Constrained reinforcement learning often considers the expected constraint violations [30], which is extended to worst-case safety constraints in [31]. A learning objective in Lagrangian formulation is proposed in [32] to improve safe behavior, which is applied to learning from demonstrations in [33]. The authors in [34] rely on Gaussian processes with a safe backup policy to ensure safety. However, these methods struggle to scale to complicated systems like robot manipulators, or cannot ensure constraint satisfaction at all times, providing only probabilistic bounds. For many constraints like collision avoidance or physical manipulator limits, probabilistic bounds are not sufficient, and the policy cannot be safely employed on the robot.

This work focuses on safe learning from demonstrations for online trajectory planning for robot manipulators. In particular, we use an adapted flow-matching model to enforce safe behavior and present constraint design considerations to enforce safety at all times. The contributions of this work are as follows:

- A novel safe flow-matching procedure, called SafeFlowMPC, is developed, where the flow-matching model improves performance and a suboptimal real-time optimization solver enforces safety. This provides a deep integration of the learning agent and constraint enforcement.
- SafeFlowMPC guarantees safety at all times by enforcing a safe terminal constraint.
- Reactive motion planning is demonstrated on a real-world 7-DoF robot manipulator. Two challenging scenarios are considered: An object grasp with a dynamically changing grasping position, and a human-robot object handover.

By integrating a flow-matching model with a real-time optimization backend, our method combines the adaptability of learning-based planning with the safety guarantees of classical control methods. This addresses the challenges of safety, reactivity, and real-time applicability in dynamic environments.

II. Formulation

The motion planning problem consists of finding a trajectory $\mathbf{q}^*(t)$ for the configuration state $\mathbf{q}$ of a robot manipulator that satisfies the motion constraints and achieves the desired objective. Formally, this is defined as

$$\mathbf{q}^*(t) = \arg \min_{\mathbf{q}(t) \in \mathcal{M}_{\text{safe}}} \int_{t_0}^{t_0+T} J(\mathbf{q}(t)) \mathrm{d}t \ , \tag{1}$$

where the trajectory $\mathbf{q}^*(t)$ starts at t_0 with a trajectory duration T. The manifold of allowed trajectories $\mathcal{M}_{\text{safe}}$ is defined by enforcing all motion constraints, and the objective function $J(\mathbf{q}(t))$ is minimal when the desired objective is achieved. A visualization of the formulation is shown in Fig. 1. For a simple movement task, the objective function

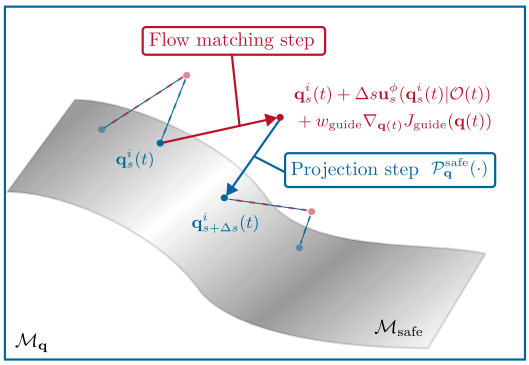

Fig. 1. Visual explanation of the optimization scheme of the proposed method. At time step i the trajectory $\mathbf{q}_s^i(t)$ on the safety manifold $\mathcal{M}_{\text{safe}}$ is improved by a flow matching step (red) and projected back onto $\mathcal{M}_{\text{safe}}$ (blue) to obtain the safe trajectory $\mathbf{q}_{s+\Delta s}^i(t)$. These two steps are executed multiple times for each time step (dashed lines) to traverse the flow from $s = 0$ to $s = 1$.

is defined as the distance to a desired configuration. The constraints limit the allowed trajectories to the manifold $\mathcal{M}_{\text{safe}}$ which contains all safe trajectories. Difficulties arise due to the complicated form of $\mathcal{M}_{\text{safe}}$ and the numerical objective function $J(\mathbf{q}(t))$ that achieves the desired task. For the latter, learning from demonstrations [19] has been successful in planning robot motions without explicitly specifying $J(\mathbf{q}(t))$. Instead, motions are learned that behave similar to a set of demonstrated trajectories. These learning-based methods often lack the ability to adhere to certain constraints and to keep the robot on $\mathcal{M}_{\text{safe}}$ at all times.

In this work, we propose a combination of flow matching [21] for learning behavior from demonstrations and optimizing $J(\mathbf{q}(t))$ with an optimization-based MPC to adhere to motion constraints and keep $\mathbf{q}(t)$ on $\mathcal{M}_{\text{safe}}$.

A. Safety Manifolds

The manifold of all possible trajectories is defined as $\mathcal{M}_{\mathbf{q}}$. The safety manifold

$$\mathcal{M}_{\text{safe}} = \{\mathbf{q}(t) : \mathbf{h}(\mathbf{q}(t)) \leq \mathbf{0} \wedge \mathbf{g}(\mathbf{q}(t)) = \mathbf{0}\} \tag{2}$$

consists of all trajectories that satisfy the inequality constraints $\mathbf{h}(\mathbf{q}(t)) \leq \mathbf{0}$ and the equality constraints $\mathbf{g}(\mathbf{q}(t)) = \mathbf{0}$. Furthermore, the performance manifold $\mathcal{M}_* = \{\mathbf{q}(t) : \mathbf{q}(t) \in \mathcal{M}_{\text{safe}} \wedge J(\mathbf{q}(t)) \text{ is minimized}\}$ is constructed based on the trajectories that are in $\mathcal{M}_{\text{safe}}$ and also minimize the objective function $J(\mathbf{q}(t))$. When optimizing a trajectory $\mathbf{q}_{\text{safe}}(t) \in \mathcal{M}_{\text{safe}}$ to move it onto $\mathcal{M}_*$, it is possible to define the projection operator

$$\mathbf{q}_*(t) = \mathcal{P}^*_{\text{safe}}(\mathbf{q}_{\text{safe}}(t)) \tag{3}$$

that projects $\mathbf{q}_{\text{safe}}(t)$ onto $\mathcal{M}_*$. However, such a projection is generally non-trivial for manipulator motion planning as it poses a non-convex optimization problem. Another approach is to use the local properties of $\mathcal{M}_{\text{safe}}$ at $\mathbf{q}_{\text{safe}}(t)$ to move the trajectory iteratively toward $\mathcal{M}_*$. This iterative procedure is formally defined as

$$\mathbf{q}^+_{\text{safe}}(t) = \mathcal{P}^{\text{safe}}_{\mathbf{q}}(\mathbf{q}_{\text{safe}}(t) + \Delta\mathbf{q}_{\text{safe}}(t)) \ , \tag{4}$$

where $\mathcal{P}_{\mathbf{q}}^{\text{safe}}$ projects the initial trajectory modified by $\Delta\mathbf{q}_{\text{safe}}(t)$ onto $\mathcal{M}_{\text{safe}}$. The modification $\Delta\mathbf{q}_{\text{safe}}(t)$ may be arbitrary, but it is assumed to be small. A possible choice is the gradient of the objective function $J(\mathbf{q}_{\text{safe}}(t))$ projected onto the tangent space of $\mathcal{M}_{\text{safe}}$ at $\mathbf{q}_{\text{safe}}(t)$. This choice results in the projection-gradient method. Generally, any choice is viable as the projection operator $\mathcal{P}_{\mathbf{q}}^{\text{safe}}$ projects $\mathbf{q}_{\text{safe}}(t) + \Delta\mathbf{q}_{\text{safe}}$ back onto $\mathcal{M}_{\text{safe}}$. The projection in (4) is simpler than the projection in (3) due to the assumption that $\Delta\mathbf{q}_{\text{safe}}$ is small. This is valid, if the projection is assumed to be performed by gradient-based non-convex optimization where the problem simplifies if the initial guess $\mathbf{q}_{\text{safe}}(t)$ is close to the optimal solution $\mathbf{q}_{\text{safe}}^{+}(t)$. The components in (4) are described more thoroughly in Sections II-B and II-C.

B. Flow Matching on Manifolds

The trajectory modification $\Delta\mathbf{q}_{\text{safe}}(t)$ in (4) is chosen to be computed by a flow matching model. Flow matching models are used in motion planning to learn a motion policy from a set of demonstrations $\mathcal{D}$. This is achieved by learning a flow model that transforms samples from an initial distribution over trajectories $p_0(\mathbf{q}(t)|\mathcal{O}(t))$ into a target distribution $p_1(\mathbf{q}(t)|\mathcal{O}(t))$. It is assumed that the demonstrations in $\mathcal{D}$ are samples of $p_1(\mathbf{q}(t)|\mathcal{O}(t))$. The distributions are conditioned on the observation $\mathcal{O}(t)$, which differs from formulations used in similar work. Further, the source distribution $p_0(\mathbf{q}(t)|\mathcal{O}(t))$ may be an arbitrary distribution and is not limited to normal distributions. Sampling from a normal distribution over joint configurations will generally not output trajectories that are on the manifold $\mathcal{M}_{\text{safe}}$. This work focuses on having safe trajectories during the transfer from the source distribution $p_0(\mathbf{q}(t)|\mathcal{O}(t))$ to the target distribution $p_1(\mathbf{q}(t)|\mathcal{O}(t))$. The source distribution is the distribution over safe trajectories on $\mathcal{M}_{\text{safe}}$ given the current observation $\mathcal{O}(t)$, and the target distribution is the distribution over safe trajectories that also optimize the objective function $J(\mathbf{q}(t))$ and, thus, lie on $\mathcal{M}_*$.

1) Mathematical Formulation: Transforming a sample $\mathbf{q}_0(t)$ from the initial distribution $p_0(\mathbf{q}(t)|\mathcal{O}(t))$ to a sample $\mathbf{q}_1(t)$ of $p_1(\mathbf{q}(t)|\mathcal{O}(t))$ is done using a conditional flow $\mathbf{u}_s^{\phi}(\mathbf{q}(t)|\mathcal{O}(t))$ with $0 \leq s \leq 1$ such that

$$\mathbf{q}_1(t) = \mathbf{q}_0(t) + \int_0^1 \mathbf{u}_s^{\phi}(\mathbf{q}(t)|\mathcal{O}(t))\mathrm{d}s \ . \tag{5}$$

The flow $\mathbf{u}_s^{\phi}$ is parametrized by the parameters ϕ, which are learned using the loss function

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{\substack{s,\mathbf{q}_0\sim p_0,\\ \mathbf{q}_1\sim p_1}} \|\mathbf{u}_s^{\phi}(\mathbf{q}(t)|\mathcal{O}(t)) - \mathbf{u}_s(\mathbf{q}(t)|\mathbf{q}_0(t), \mathbf{q}_1(t))\|_2^2 \ , \tag{6}$$

where the target flow

$$\mathbf{u}_s(\mathbf{q}(t)|\mathbf{q}_0(t), \mathbf{q}_1(t)) = \frac{\partial}{\partial s}\mathcal{P}_{\mathbf{q}}^{\text{safe}}\left(\mathbf{q}_0(t) + s(\mathbf{q}_1(t) - \mathbf{q}_0(t))\right) \tag{7}$$

is computed using the projection operator from (4) and conditioned on $\mathbf{q}_0(t)$ and $\mathbf{q}_1(t)$ for its tractability. The projection operator ensures that the path (5) always remains on $\mathcal{M}_{\text{safe}}$ as long as $\mathbf{q}_0(t) \in \mathcal{M}_{\text{safe}}$. It further ensures that the path ends on $\mathcal{M}_*$ as $s \to 1$ because $\mathbf{q}_1(t) = \mathcal{P}_{\mathbf{q}}^{\text{safe}}(\mathbf{q}_1(t))$. For more information on flow matching, the reader is referred to [21].

2) Training Procedure: Training our model is a two-step procedure. First a model is trained without any safety considerations by trying to match the demonstrations as done in standard flow matching [21]. Secondly, this model is finetuned on a safety dataset, which is the original dataset, but each demonstration is adapted to be on $\mathcal{M}_{\text{safe}}$. Specifically, the output distribution $p_1(\mathbf{q}(t)|\mathcal{O}(t))$ is computed by projecting each sample in $\mathcal{D}$ onto $\mathcal{M}_{\text{safe}}$. Furthermore, the safe input distribution $p_0(\mathbf{q}(t)|\mathcal{O}(t))$, needed in (7), is approximated by sampling and projection. The derivative of the projection operator (4) with respect to s in (7) is computed using finite differences. This way the safety dataset consists of samples of (7) for different values of s and samples of $\mathbf{q}_0(t)$ that move toward $\mathbf{q}_1(t)$.

C. Trajectory projection

Projecting joint-space trajectories $\mathbf{q}(t)$ onto the safety manifold $\mathcal{M}_{\text{safe}}$ is described as a non-convex optimization problem to enforce the constraints $\mathbf{h}(\mathbf{q}(t))$ and $\mathbf{g}(\mathbf{q}(t))$. This assumes that $\mathbf{h}(\mathbf{q}(t))$ and $\mathbf{g}(\mathbf{q}(t))$ are continuously differentiable to make gradient-based optimization feasible. The projection operator $\mathcal{P}_{\mathbf{q}}^{\text{safe}}$ in (4) is defined as

$$\mathcal{P}_{\mathbf{q}}^{\text{safe}}(\mathbf{q}_{\text{init}}(t)) = \arg\min_{\mathbf{q}(t)} D(\mathbf{q}(t), \mathbf{q}_{\text{init}}(t)) \tag{8a}$$

$$\text{s.t.} \quad \mathbf{h}(\mathbf{q}(t)) = \begin{bmatrix} \mathbf{h}_t(\mathbf{q}(t)) \\ \mathbf{h}_T(\mathbf{q}(t)) \end{bmatrix} \leq \mathbf{0} \tag{8b}$$

$$\mathbf{g}(\mathbf{q}(t)) = \begin{bmatrix} \mathbf{g}_t(\mathbf{q}(t)) \\ \mathbf{g}_T(\mathbf{q}(t)) \end{bmatrix} = \mathbf{0} \tag{8c}$$

where $D(\cdot)$ is an appropriate distance measure for the trajectories. The constraints (8b) and (8c) comprise the constraints during the trajectory $\mathbf{h}_t(\mathbf{q}(t))$ and $\mathbf{g}_t(\mathbf{q}(t))$, and terminal constraints at the end of the trajectory $\mathbf{h}_T(\mathbf{q}(t))$ and $\mathbf{g}_T(\mathbf{q}(t))$ at time $t = t_0 + T$. These terminal constraints define the safety set $\mathcal{S}_T$.

Assumption 1. *A controller exists that has a control-invariant safety set that encompasses the terminal safety set $\mathcal{S}_T$ defined by $\mathbf{h}_T(\mathbf{q}(t))$ and $\mathbf{g}_T(\mathbf{q}(t))$.*

Theorem 1. *The projected trajectory $\mathbf{q}_{\text{proj}}(t) = \mathcal{P}_{\mathbf{q}}^{\text{safe}}(\mathbf{q}_{\text{init}}(t))$ extending from the initial time $t = t_0$ to the end time $t = t_0 + T$ is safe for all times $t > t_0 + T$. Safety at time t is defined by satisfying the constraints $\mathbf{h}(\mathbf{q}(t))$ and $\mathbf{g}(\mathbf{q}(t))$.*

Proof. The trajectory $\mathbf{q}_{\text{proj}}(t)$ will end in the terminal safety set $\mathcal{S}_T$ defined by $\mathbf{h}_T(\mathbf{q}(t))$ and $\mathbf{g}_T(\mathbf{q}(t))$ at time $t = t_0 + T$. At time $t = t_0 + T$ the controller from Assumption 1 is employed to keep the trajectory in the terminal safety set $\mathcal{S}_T$ for all times $t > t_0 + T$. □

D. Inference

After training the model on the dataset $\mathcal{D}$ with the loss (6), the model is used for inference. Similar to the work [10], [22], we use the model in closed-loop, where the model

predicts the trajectory $\mathbf{q}^i(t)$ at time step i for the time span $T = NT_\mathrm{s}$, with the prediction horizon N and the sampling time T_s. The trajectory $\mathbf{q}^i(t)$ is executed for the time T_s only, and then a new trajectory $\mathbf{q}^{i+1}(t)$ is computed at time step $i+1$. A step-by-step explanation is given in Algorithm 1. The prediction step is executed with

$$\begin{aligned}\mathbf{q}^i_{s+\Delta s}(t) = \mathcal{P}^{\mathrm{safe}}_{\mathbf{q}}\Big(&\mathbf{q}^i_s(t) + \Delta s \mathbf{u}^\phi_s(\mathbf{q}^i_s(t)|\mathcal{O}(t)) \\ &+ w_{\mathrm{guide}} \nabla_{\mathbf{q}(t)} J_{\mathrm{guide}}(\mathbf{q}(t))\Big)\end{aligned} \tag{9}$$

for a fixed flow step Δs to get from $\mathbf{q}^i_0(t)$ to $\mathbf{q}^i_1(t)$, where the subscript denotes the probability flow time with bounds $0 \leq s \leq 1$. This is visualized in Fig. 1. The derivative of the guidance cost term $J_{\mathrm{guide}}(\mathbf{q}(t))$ is added with the weight $w_{\mathrm{guide}} > 0$, similar to diffusion guidance [17]. Its particular design is application dependent and will be explained in the experiments section. The trajectory for the next step $i+1$ is then defined by

$$\mathbf{q}^{i+1}_0(t) = \begin{cases} \mathbf{q}^i_1(t) & t_0 + T_\mathrm{s} < t < t_0 + T \\ \mathbf{q}^i_T(t) & t_0 + T \leq t \leq t_0 + T + T_\mathrm{s} \end{cases}, \tag{10}$$

where t_0 is the starting time of $\mathbf{q}^i_1(t)$. This sets the initial trajectory for (9) at step $i+1$ to the optimized trajectory at step i until it ends at t_0+T and afterward applies the terminal trajectory $\mathbf{q}^i_T(t)$ according to Assumption 1, which keeps the robot in the terminal safety set $\mathcal{S}_T$. A visual explanation of the iterative planning is provided in Fig. 2.

Assumption 2. *There exists a safe trajectory $\mathbf{q}^0(t)$ at the start of the robot movement such that the robot stays on the safety manifold $\mathcal{M}_{\mathrm{safe}}$.*

This assumption is not very restrictive as it is often possible to stay in the safe initial state indefinitely.

Theorem 2. *The iterative trajectory generation (10) will keep the robot safe at all times.*

Proof. At the start of the movement, the robot is in a safe state according to Assumption 2. After moving from step i to step $i+1$, the robot is safe because the projection in (9) keeps the new trajectory $\mathbf{q}^i_s(t)$ on the safety manifold $\mathcal{M}_{\mathrm{safe}}$. This proof by induction works as long as the projection in (9) converges. However, the projection operator $\mathcal{P}^{\mathrm{safe}}_{\mathbf{q}}(\cdot)$ introduced in (8) is a non-convex optimization problem where convergence cannot be ensured. In case of a failure in (9) at $s = s_{\mathrm{fail}}$, the current trajectory $\mathbf{q}^i_{s_{\mathrm{fail}}}(t)$ is executed, which keeps the system safe according to Theorem 1 at all times due to enforcing the terminal safety set $\mathcal{S}_T$. □

III. Practical Implementation: Robot Manipulator

This section discusses the implementation details of using the safe trajectory generation described in Section II for the motion planning of a robot manipulator, for which $\mathbf{q}(t)$ denotes the joint-position trajectory. For shorter notation, we use the state $\mathbf{x}^{\mathrm{T}}(t) = [\mathbf{q}^{\mathrm{T}}(t), \dot{\mathbf{q}}^{\mathrm{T}}(t), \ddot{\mathbf{q}}^{\mathrm{T}}(t), \dddot{\mathbf{q}}^{\mathrm{T}}(t)]$. The

Algorithm 1 SafeFlowMPC Inference

Require: Initial safe trajectory $\mathbf{q}^0_0(t)$, flow model $\mathbf{u}^\phi_s$, guidance cost J_{guide}, projection operator $\mathcal{P}^{\mathrm{safe}}_{\mathbf{q}}$, weight for cost guidance w_{guide}
Require: Number of flow steps N_s, prediction horizon N, sampling time T_s

```
Initialize i ← 0, t_0 ← 0
while task not completed do
    q^i_s(t) ← q^i_0(t)                          ▷ Initialize flow trajectory
    Δs ← 1/N_s                                   ▷ Flow step size
    Retrieve observation O(t)
    for k = 1 to N_s do                          ▷ Safe flow matching steps
        Compute q^i_{s+Δs}(t) from q^i_s(t) using (9)
        s ← s + Δs
    end for
    q^i_1(t) ← q^i_s(t)                          ▷ Final flow trajectory
    Execute the trajectory q^i_1(t) for one time step T_s
    Update the initial trajectory using (10)
    i ← i + 1
    t_0 ← t_0 + T_s
end while
```

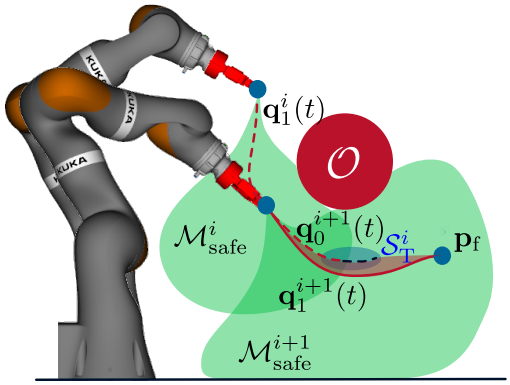

Fig. 2. Planning scheme of SafeFlowMPC for the trajectory planning of a robot manipulator around an obstacle $\mathcal{O}$. At step i, the robot plans the trajectory $\mathbf{q}^i_1(t)$ on the safety manifold $\mathcal{M}^i_{\mathrm{safe}}$. This trajectory ends in the terminal safety set $\mathcal{S}^i_T$ indicated by the blue shaded area. At step $i+1$ the planner reuses the previous trajectory according to (10) and transfers it (red shaded area) to $\mathbf{q}^{i+1}_1(t)$ on $\mathcal{M}^{i+1}_{\mathrm{safe}}$.

constraints of the safety manifold $\mathcal{M}_{\mathrm{safe}}$ in (2) are given by

$$\mathbf{h}(\mathbf{q}(t)) = \begin{cases} \underline{\mathbf{x}} \leq \mathbf{x}(t) \leq \overline{\mathbf{x}} & \forall \quad t_0 \leq t \leq t_0 + T \\ \mathbf{f}_{\mathrm{fk},i}(\mathbf{q}(t)) \in \mathcal{S}_{\mathrm{free}} & \forall \quad t_0 \leq t \leq t_0 + T \\ & \quad i = 1, \ldots, N_{\mathrm{fk}} \end{cases} \tag{11}$$

and

$$\mathbf{g}(\mathbf{q}(t)) = \begin{cases} \mathbf{x}(t_0) = \mathbf{x}_0 \\ [\dot{\mathbf{q}}^{\mathrm{T}}(t_0+T), \ddot{\mathbf{q}}^{\mathrm{T}}(t_0+T), \dddot{\mathbf{q}}^{\mathrm{T}}(t_0+T)] = \mathbf{0} \end{cases}, \tag{12}$$

where the initial state $\mathbf{x}_0$ is taken from the previous trajectory according to (10), which ensures continuity across time steps. The state $\mathbf{x}$ is kinematically bounded by the lower bound $\underline{\mathbf{x}}$ and the upper bound $\overline{\mathbf{x}}$. The second constraint in (12) ensures containment in the terminal set $\mathcal{S}_T$ and, thus, safety of the robot for $t \geq t_0 + T$. The constraint $\mathbf{f}_{\mathrm{fk},i}(\mathbf{q}(t)) \in \mathcal{S}_{\mathrm{free}}$ in (11) uses the forward kinematics $\mathbf{f}_{\mathrm{fk},i}(\mathbf{q}(t))$ of the manipulator to ensure that the kinematic chain is collision

free. In this work, the collision formulation from [2] is utilized, which uses collision-free convex sets around N_{fk} key points of the manipulator. This approach scales well with the number of obstacles and is real-time capable. For more information on this approach, the reader is referred to [2]. The forward kinematics $\mathbf{f}_{\text{fk},i}(\mathbf{q}(t))$ are nonlinear, rendering the problem (8) non-convex. This implies that (8) has multiple (local) minima, making it difficult to reliably solve. Employing a non-convex solver for this projection and solving it optimally is computationally infeasible as the projection is performed multiple times to get from $\mathbf{q}_0^i(t)$ to $\mathbf{q}_1^i(t)$ using (9). Therefore, this work adapts a suboptimal approach using the real-time-iteration (RTI) scheme [35] in the *acados* framework [36], which solves exactly one QP problem based on (8) for each step (9). In practice, a suboptimal solution is often sufficient and will converge to the optimal solution over multiple steps using (9). In this work, $N_s = 7$ steps of (9) are used in each time step.

IV. Experiments

Three experiments are presented in this section to evaluate performance, versatility, and adherence to safety constraints of the developed method. The first experiment utilizes our method to learn from a global trajectory planner to achieve fast local planning in a global context. The second experiment focuses on reactive online replanning in an obstructed environment. The third experiment is a dynamic human-robot object handover where the robot learns behavior from a human-human handover dataset [7].

We compare our method to the following formulations:

1) *VP-STO*: VP-STO [37] is a global trajectory planner based on stochastic sampling.
2) *BoundMPC*: BoundMPC [13] is a model-predictive trajectory planner with a path-following-based formulation. The path is computed by BoundPlanner [2].
3) *BC*: Behavior cloning [38] using a multi-layer perceptron, which takes the last trajectory and the current state as input and outputs the next trajectory. The training only includes safe trajectories to incentivize safe behavior.
4) *FM*: The baseline flow-matching training procedure without any safety considerations [21].
5) *Ours with NL-Opt*: Our proposed method, but the projection (8) is solved to optimality instead of using the real-time iteration approach described in Section III. The IPOPT [39] solver with the MA57 linear solver [40] is used in this work. To increase the speed, only 4 steps of (9) are used in each time step.
6) *Ours w/o finetuning*: Our proposed method without the finetuning on the safety dataset described in Section II-B.2. The training simplifies to the standard flow matching training procedure.

The flow network architecture for all flow matching models is a temporal U-Net network as in [16]. All online planners are executed at 10 Hz. A video of the experiments is available at https://www.acin.tuwien.ac.at/en/42d6.

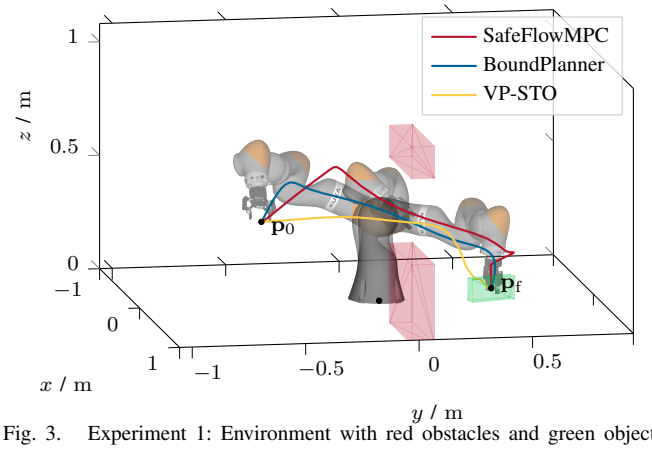

Fig. 3. Experiment 1: Environment with red obstacles and green object to grasp. An example trajectory for three planners is shown. The robot is visualized for the start and end configuration of the SafeFlowMPC trajectory.

A. Experiment 1: Global trajectory planning made local

By learning from global trajectory planners, SafeFlowMPC is able to exhibit close-to-optimal behavior while being real-time executable, thus, combining the advantages of global and local planners. To learn this behavior, a dataset $\mathcal{D}$ was created using the global planner VP-STO [37] in the environment depicted in Fig. 3 with two obstacles forming a narrow passage. The goal is to plan from an initial joint configuration $\mathbf{q}_{\text{init}}(0)$, where the end-effector is at pose $\mathbf{p}_0$, to a final end-effector pose $\mathbf{p}_\text{f}$, where an object is picked up. The dataset consists of 4100 trajectories, i.e., 4000 for training and 100 for evaluation. The guidance function

$$J_{\text{guide}}(\mathbf{q}(t)) = \sum_{j=1}^{N_s} D_{\text{pose}}\left(\mathbf{f}_{\text{fk,ee}}(\mathbf{q}(t_0 + jT_\text{s})), \mathbf{p}_\text{f}\right) \quad (13)$$

used in (9) is defined by the distance between the current end-effector pose $\mathbf{f}_{\text{fk,ee}}(\mathbf{q}(t))$ and the final pose $\mathbf{p}_\text{f}$, where the function D_{pose} measures the distance between two poses. It is the sum of the Cartesian distance for the position and the rotation vector angle difference between two orientations. Equation (13) ensures convergence to the final pose $\mathbf{p}_\text{f}$. The weight

$$w_{\text{guide}}(t) = \exp\left(-\alpha\left(\frac{\|\mathbf{f}_{\text{fk,ee}}(\mathbf{q}(t)) - \mathbf{p}_0\|_2}{\|\mathbf{p}_0 - \mathbf{p}_\text{f}\|_2} - \beta\right)\right) \quad (14)$$

is chosen such that it only influences the behavior close to $\mathbf{p}_\text{f}$ with the parameters $\alpha > 0$ and $\beta > 0$. The observation $\mathcal{O}(t)$ consists of the joint positions of the previous 10 time steps, the Cartesian positions of the collision-checked points on the robot, the current pose of the end effector, and the desired pose of the end effector $\mathbf{p}_\text{f}$.

The results of all planners described above are in Table I for the 100 evaluation trajectories in terms of average trajectory duration T_{traj}, planning time T_{plan}, success rate r_{success} for reaching the final pose without collisions, and the maximum obstacle collision c_{obs}. The latter is the maximum obstacle penetration depth in case of a collision and zero otherwise. SafeFlowMPC shows low trajectory times T_{traj} comparable to the global planner VP-STO it

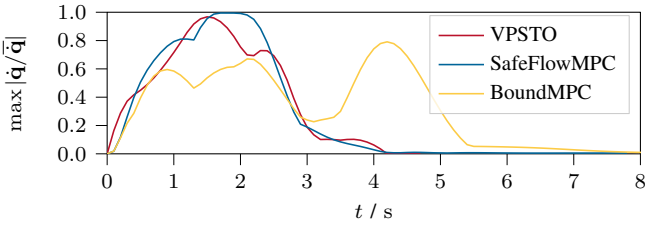

Fig. 4. Experiment 1: Maximum values of the normalized joint velocity for the example trajectories in Fig. 3.

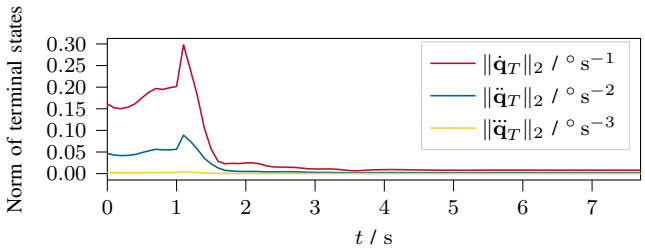

Fig. 5. Experiment 1: Norm of joint velocity $\dot{\mathbf{q}}_T$, acceleration $\ddot{\mathbf{q}}_T$, and jerk $\dddot{\mathbf{q}}_T$ at the end of each planning horizon using SafeFlowMPC for the example trajectory in Fig. 3.

learned from. Additionally, the robot moves through the environment without violating the constraints. The success rate r_{success} is higher than for the MPC-based planner BoundMPC. BoundMPC is also safe but has considerably longer trajectory times T_{traj} since it only plans locally. Behavior cloning (BC) has fast planning time T_{plan} since it is only executed once every time step. However, this model is unable to solve the task successfully most of the time. The baseline flow-matching method (FM) is fast in comparison to SafeFlowMPC in terms of T_{traj}, but has much a lower success rate r_{success} due to collisions. The average planning time T_{plan} is fast for BoundMPC, but the time to solve its nonlinear optimization problem varies significantly. Therefore, its planning horizon needs to be chosen such that the average time is low to account for potential longer solves. SafeFlowMPC is much better at exploiting the maximum planning time of $100\,\text{ms}$ because the iterations (9) are designed such that staying on the safety manifold is incentivized and a safe trajectory always exists according to Theorem 2. Solving the projection problem (8) to optimality (Ours w/ NL-Opt) leads to similar success rates as our method but incurs much higher computational cost as indicated by the average planning time T_{plan}. The large standard deviation of T_{plan} is especially problematic as it frequently violates the real-time property. This is more pronounced than for BoundMPC, which also solves a nonlinear optimization problem, because the flow network presents an unknown input to the solver, which complicates the problem. The finetuning for SafeFlowMPC, described in Section II-B.2, is very important as the model without the finetuning (Ours w/o finetuning) has a considerably lower task success r_{success}. Note that even without the finetuning, the model remains safe, i.e., $c_{\text{obs}} = 0$. Furthermore, the maximum of the normalized joint velocities in Fig. 4 shows that SafeFlowMPC is able to exploit the full range of the joint velocities without violating the limits. This leads to a shorter trajectory compared to BoundMPC. Additionally, the joint states at the end of each planning horizon, depicted in Fig. 5, are close to zero such that the horizon always ends in the safe terminal set $\mathcal{S}_T$ according to (12). The states are not exactly zero as this is difficult to enforce but sufficiently low for the low-level joint controller to stabilize around the final point as demonstrated in the supplementary video.

B. Experiment 2: Online replanning for object grasps

The reactivity of SafeFlowMPC is compared in the environment shown in Fig. 3 by changing the pose of the object during the motion three times randomly. The results for the real-time capable methods are reported in Table II. A motion is counted as successful if it reaches the last replanned object pose and has no collisions. Generally, the drop in success rate for all methods indicates that this task is more challenging than Experiment 1 in Section IV-A. SafeFlowMPC is the most successful method and is able to replan motions in real time without any collisions in $82\,\%$ of the trails.

Thus, the SafeFlowMPC formulation enables fast and safe trajectory planning with similar quality as global trajectory planners. At the same time, it allows for more reactivity due to the online replanning.

TABLE I

EXPERIMENT 1: RESULTS OF DIFFERENT PLANNERS IN TERMS OF AVERAGE TRAJECTORY DURATION T_{traj}, AVERAGE PLANNING TIME T_{plan}, OBSTACLE COLLISIONS $c_{\text{obs}} > 0$, AND SUCCESS RATE r_{success}

Method	T_{traj} / s	T_{plan} / ms	c_{obs} / m	r_{success}
VP-STO	4.46	11400 (offline)	0	100 %
BoundMPC	6.16	35 ± 10	0	76 %
BC	7.6	1 ± 1	0.08	2 %
FM	5.77	33 ± 2	0.08	22 %
Ours w/ NL-Opt	5.69	93 ± 178	0	87 %
Ours w/o finetuning	6.40	64 ± 5	0	58 %
SafeFlowMPC (Ours)	5.12	62 ± 5	0	86 %

C. Experiment 3: Dynamic human-robot object handover

While the objective for the point-to-point motions in Sections IV-A and IV-B is easily encoded as a numerical function, i.e., minimizing the distance to the goal pose, this is not the case for many other objectives. Especially in human-robot interactions, it is difficult to find such functions. SafeFlowMPC allows using models learned from demonstrated

TABLE II

EXPERIMENT 2: RESULTS OF DIFFERENT PLANNERS IN TERMS OF AVERAGE PLANNING TIME T_{plan}, OBSTACLE COLLISIONS $c_{\text{obs}} > 0$, AND SUCCESS RATE r_{success}

Method	T_{plan} / ms	c_{obs} / m	r_{success}
BoundMPC	36 ± 11	0	74 %
BC	1 ± 1	0.08	0 %
FM	33 ± 2	0.08	15 %
Ours w/o finetuning	62 ± 5	0	52 %
SafeFlowMPC (Ours)	63 ± 5	0	82 %

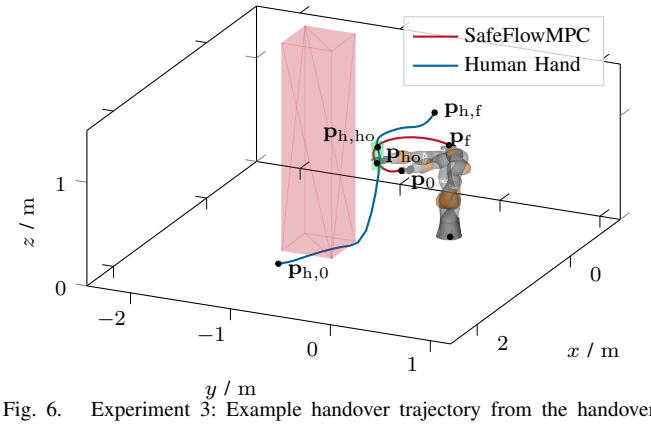

Fig. 6. Experiment 3: Example handover trajectory from the handover dataset. The robot is visualized for the start, handover, and end configuration of the SafeFlowMPC trajectory. The human is represented by the red box at the handover location grabbing the green object from the robot's end-effector.

trajectories instead of optimizing a scenario-specific reward function. We demonstrate this with a dynamic human-robot object handover, where a robot hands an object to a human as the human passes the robot. The human-human handover dataset from [7] is used for this task. The human hand trajectories of the giver are converted to the joint space of the robot using a trajectory optimization problem. The dataset contains 900 trajectories for training and 100 for testing of which 67 where successfully translated to the joint space. An example trajectory is depicted in the environment in Fig. 6. In this example, the human hand moves from the initial position $\mathbf{p}_{\mathrm{h},0}$ to the handover location $\mathbf{p}_{\mathrm{h,ho}}$ to grab the object and then continues to the final position $\mathbf{p}_{\mathrm{h,f}}$. The corresponding robot end-effector poses are $\mathbf{p}_0$, $\mathbf{p}_{\mathrm{ho}}$, and $\mathbf{p}_{\mathrm{f}}$. At the handover location, the human hand is above the robot's end-effector, as the object extends in the z-direction. Safety is introduced by modeling the human body as an obstacle. This does not include the arm of the human as it needs to grab the object and come in close proximity to the end effector. As (11) limits the joint jerk of the robot, the resulting trajectory is smooth, which is important for the perceived safety for the human [8], [41]. Furthermore, the constraint

$$\mathbf{f}_{\mathrm{fk}}(\mathbf{q}(t_0 + T)) \in \mathcal{S}_{T,\mathrm{ho}} \tag{15}$$

is added to (11) as a terminal constraint to ensure that the robot returns the end-effector to a safe set when planning fails. As the human always passes the robot on the negative y-direction (see Fig. 6), the safe terminal set $\mathcal{S}_{T,\mathrm{ho}}$ constraints the end-effector to limit the y-position to remain close to the robot such that it cannot interfere with the human. SafeFlowMPC computes the robot joint trajectory and the binary gripper state to learn the release timing of the object. No guiding function $J_{\mathrm{guide}}(\mathbf{q}(t))$ is used here.

The comparison in Table III on the test dataset shows that SafeFlowMPC computes trajectories that are closest to the demonstrations while remaining safe according to (15) and real-time capable. Note that the average distance to the demonstrations d_{demo} assumes no reaction of the human to the differing robot motions, which is not applicable to real-world object handovers. Therefore, several real-world handovers are shown in the supplementary video, where the human hand is tracked with an OptiTrack system[1]. They show that SafeFlowMPC succeeds to hand over an object in the real world by learning from human demonstrations.

TABLE III

EXPERIMENT 3: RESULTS OF DIFFERENT PLANNERS IN TERMS OF AVERAGE PLANNING TIME T_{plan}, TERMINAL CONSTRAINT VIOLATIONS $c_T > 0$, AND AVERAGE POSITION DISTANCE TO THE DEMONSTRATED TRAJECTORIES d_{demo}

Method	T_{plan} / ms	c_T / m	d_{demo} / m
BC	1 ± 1	0.0	0.28
FM	31 ± 2	0.05	0.21
SafeFlowMPC (Ours)	63 ± 6	0.0	0.12

V. LIMITATIONS

Despite its strengths, SafeFlowMPC has notable limitations. It assumes the availability of a suitable dataset and requires a finetuning step to achieve optimal. Additionally, the method is restricted to constraints expressible as differentiable functions over the planning horizon. The design of safe terminal constraints also remains non-trivial, particularly for complex kinematics. Furthermore, ensuring safety in interactive scenarios, such as the presented dynamic object handover in Section IV-C, is generally difficult. In this work, we simplified the safety definition by considering the collisions with the current position of the human. However, the human is moving around, which is difficult to predict, requiring a trade-off between safety and performance [8]. A possible approach for future work is the prediction of the trajectory distribution of other actors [42] in combination with safety constraints [8]. This is out of the scope of this work as the dataset [7] contains very noisy data of the human movements, which is unsuitable for such models. The object handover further simplifies the interactions within human-robot handovers by only considering the human hand trajectory as input for SafeFlowMPC. Improvements in performance can be gained by considering other factors such as trust, gaze, or other actors in the scene [41], which requires more expressive datasets.

VI. CONCLUSIONS AND FUTURE WORK

This work introduced SafeFlowMPC, a novel method for online trajectory planning that integrates safety manifolds with conditional flow-matching. The approach optimizes receding-horizon trajectories through an iterative process: a flow matching network generates desired motions, while an online optimizer enforces safety constraints. By carefully designing terminal constraints, SafeFlowMPC guarantees safety throughout operation. We demonstrated the effectiveness of SafeFlowMPC in three challenging scenarios—static and dynamic object grasping with obstacles, and human-robot object handover—where it outperformed the baseline

[1]OptiTrack https://www.optitrack.com/

methods. Successful deployment on a real-world KUKA 7-DoF manipulator further validated the method's real-time feasibility and practical applicability. Future research will focus on extending SafeFlowMPC with learning-based safety filters [25] and terminal constraints, aiming to broaden its applicability and robustness in dynamic, real-world settings.

REFERENCES

[1] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[2] T. Oelerich, C. Hartl-Nesic, F. Beck, and A. Kugi, "BoundPlanner: A Convex-Set-Based Approach to Bounded Manipulator Trajectory Planning," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 5393–5400, 2025.

[3] S. M. Khansari-Zadeh and A. Billard, "Realtime Avoidance of Fast Moving Objects: A Dynamical System-based Approach," in *Workshop on Robot Motion Planning: Online, Reactive, and in Real-time. International Conference on Intelligent Robots and Systems*, 2012.

[4] J. Kiemel, L. Righetti, T. Kröger, and T. Asfour, "Safe Reinforcement Learning of Robot Trajectories in the Presence of Moving Obstacles," *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11 353–11 360, 2024.

[5] T. Oelerich, C. Hartl-Nesic, and A. Kugi, "Model Predictive Trajectory Planning for Human-Robot Handovers," in *Proceedings of the VDI Mechatroniktagung*, 2024.

[6] M. Khoramshahi and A. Billard, "A dynamical system approach to task-adaptation in physical human–robot interaction," *Autonomous Robots*, vol. 43, no. 4, pp. 927–946, 2019.

[7] H. Kim, C. Kim, M. Pan, K. Lee, and S. Choi, "Learning-based Dynamic Robot-to-Human Handover," *Preprint (arXiv.2502.12602)*, 2025.

[8] V. Ortenzi, A. Cosgun, T. Pardi, W. P. Chan, E. Croft, and D. Kulic, "Object Handovers: A Review for Robotics," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1–19, 2021.

[9] C. Unger, C. Hartl-Nesic, M. N. Vu, and A. Kugi, "ProSIP: Probabilistic Surface Interaction Primitives for Learning of Robotic Cleaning of Edges," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2024, pp. 5956–5963.

[10] M. J. Kim *et al.*, "OpenVLA: An Open-Source Vision-Language-Action Model," in *Proceedings of the Conference on Robot Learning*, 2025, pp. 2679–2713.

[11] M. Elbanhawi and M. Simic, "Sampling-Based Robot Motion Planning: A Review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[12] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[13] T. Oelerich, F. Beck, C. Hartl-Nesic, and A. Kugi, "BoundMPC: Cartesian path following with error bounds based on model predictive control in the joint space," *The International Journal of Robotics Research*, vol. 44, no. 8, pp. 1287–1316, 2025.

[14] F. Beck, M. N. Vu, C. Hartl-Nesic, and A. Kugi, "Model Predictive Trajectory Optimization With Dynamically Changing Waypoints for Serial Manipulators," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6488–6495, 2024.

[15] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[16] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with Diffusion for Flexible Behavior Synthesis," in *Proceedings of the International Conference on Machine Learning*, 2022, pp. 9902–9915.

[17] K. Saha *et al.*, "EDMP: Ensemble-of-costs-guided Diffusion for Motion Planning," in *Proceedings of the International Conference on Robotics and Automation*, 2024, pp. 10 351–10 358.

[18] R. Figueiredo Prudencio, M. R. O. A. Maximo, and E. L. Colombini, "A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 8, pp. 10 237–10 257, 2024.

[19] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020.

[20] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 15, 2011, pp. 627–635.

[21] Y. Lipman *et al.*, "Flow Matching Guide and Code," *Preprint (arXiv:2412.06264)*, 2024.

[22] R. Römer, A. von Rohr, and A. P. Schoellig, "Diffusion Predictive Control with Constraints," *Preprint (arXiv:2412.09342)*, 2025.

[23] X. Dai *et al.*, "Safe Flow Matching: Robot Motion Planning with Control Barrier Functions," *Preprint (arXiv:2504.08661)*, 2025.

[24] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.

[25] K. P. Wabersich *et al.*, "Data-Driven Safety Filters: Hamilton-Jacobi Reachability, Control Barrier Functions, and Predictive Methods for Uncertain Systems," *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.

[26] K. Garg, S. Zhang, O. So, C. Dawson, and C. Fan, "Learning safe control for multi-robot systems: Methods, verification, and open challenges," *Annual Reviews in Control*, vol. 57, p. 100948, 2024.

[27] P. Liu, H. Bou-Ammar, J. Peters, and D. Tateo, "Safe Reinforcement Learning on the Constraint Manifold: Theory and Applications," *IEEE Transactions on Robotics*, vol. 41, pp. 3442–3461, 2025.

[28] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging Hamilton-Jacobi Safety Analysis and Reinforcement Learning," in *Proceedings of the International Conference on Robotics and Automation*, 2019, pp. 8550–8556.

[29] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *Proceedings of the Annual Conference on Decision and Control*, 2017, pp. 2242–2253.

[30] C. Sun, D.-K. Kim, and J. P. How, "FISAR: Forward Invariant Safe Reinforcement Learning with a Deep Neural Network-Based Optimizer," in *Proceedings of the International Conference on Robotics and Automation*, 2021, pp. 10 617–10 624.

[31] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. J. Spaan, "WCSAC: Worst-Case Soft Actor Critic for Safety-Constrained Reinforcement Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 10 639–10 646, 2021.

[32] A. Stooke, J. Achiam, and P. Abbeel, "Responsive Safety in Reinforcement Learning by PID Lagrangian Methods," in *Proceedings of the International Conference on Machine Learning*, 2020, pp. 9133–9143.

[33] Z. Liu *et al.*, "Datasets and benchmarks for offline safe reinforcement learning," *Journal of Data-centric Machine Learning Research*, 2024.

[34] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proceedings of the International Conference on Neural Information Processing Systems*, 2017, pp. 908–919.

[35] M. Diehl, H. G. Bock, and J. P. Schlöder, "A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[36] R. Verschueren *et al.*, "Acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.

[37] J. Jankowski, L. Brudermüller, N. Hawes, and S. Calinon, "VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior," in *Proceedings of the International Conference on Robotics and Automation*, 2023, pp. 10 125–10 131.

[38] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, 1988.

[39] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[40] I. S. Duff, "MA57—a code for the solution of sparse symmetric definite and indefinite systems," *ACM Transactions on Mathematical Software*, vol. 30, no. 2, pp. 118–144, 2004.

[41] A. Zacharaki, I. Kostavelis, A. Gasteratos, and I. Dokas, "Safety bounds in human robot interaction: A survey," *Safety Science*, vol. 127, p. 104667, 2020.

[42] L. Hewing, E. Arcari, L. P. Fröhlich, and M. N. Zeilinger, "On Simulation and Trajectory Prediction with Gaussian Process Dynamics," in *Proceedings of the Conference on Learning for Dynamics and Control*, 2020, pp. 424–434.