VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior

Julius Jankowski* [1,2], Lara Brudermüller* [3], Nick Hawes[3] and Sylvain Calinon[1,2]

Abstract— Achieving reactive robot behavior in complex dynamic environments is still challenging as it relies on being able to solve trajectory optimization problems quickly enough, such that we can replan the future motion at frequencies which are sufficiently high for the task at hand. We argue that current limitations in Model Predictive Control (MPC) for robot manipulators arise from inefficient, high-dimensional trajectory representations and the negligence of time-optimality in the trajectory optimization process. Therefore, we propose a motion optimization framework that optimizes *jointly* over space and time, generating smooth and timing-optimal robot trajectories in joint-space. While being task-agnostic, our formulation can incorporate additional task-specific requirements, such as collision avoidance, and yet maintain real-time control rates, demonstrated in simulation and real-world robot experiments on closed-loop manipulation. For additional material, please visit https://sites.google.com/oxfordrobotics.institute/vp-sto.

I. INTRODUCTION

In this paper we consider the problem of generating continuous, *timing-optimal* and smooth trajectories for robots operating in dynamic environments. Such task settings require the robot to be *reactive* to unforeseen changes in the environment, *e.g.*, due to dynamic obstacles, as well as to be *robust* and *compliant* when operating alongside or together with humans. However, generating this kind of reactive and yet efficient robot behavior within a high-dimensional configuration space is significantly challenging. This is especially the case in robot manipulation scenarios with many degrees of freedom (DoFs) as the resulting high-dimensional and multi-objective optimization problems are difficult to solve on-the-fly. A widespread approach in robotics is to formulate the task of motion generation as an optimization problem. Such *trajectory-optimization* based methods aim at finding a trajectory that minimizes a cost function, *e.g.*, motion smoothness, subject to constraints, *e.g.*, collision avoidance. Solution strategies can either be *gradient-based* or *sampling-based*. Approaches falling in the former category, *e.g.*, CHOMP [1] and TrajOpt [2], typically employ second-order iterative methods to find locally optimal solutions. However, they require the cost function to be once or even twice-differentiable, which constitutes a major limitation for manipulation tasks as they usually involve
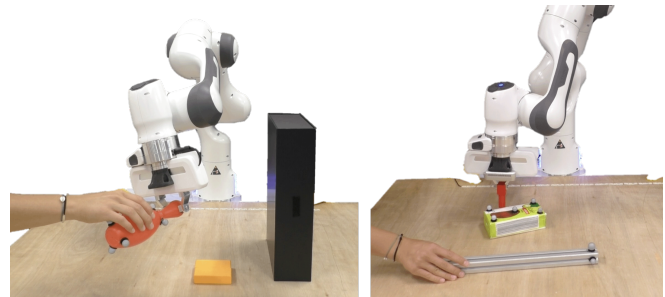
[1]Idiap Research Institute, Martigny, CH; `name.surname@idiap.ch`
[2]Ecole Polytechnique Fédérale de Lausanne (EPFL), CH
[3]Oxford Robotics Institute, University of Oxford, UK; `{larab, nickh}@robots.ox.ac.uk`.

Fig. 1. **Experiment settings.** *Left:* Pick-and-place scenario, where the task is to grasp a bowling pin that is arbitrarily handed over to the robot and to place it upright in the middle of the table. *Right:* Pushing scenario, where the robot has to push the center of the green coffee packet to a moving target location indicated by the tip of the metal stick.

many complex, discontinuous cost terms and constraints. In contrast, sampling-based methods [3], [4] can operate on discontinuous costs by sampling candidate trajectories from a proposal distribution, evaluating them on the objective, and updating the proposal distribution according to their relative performance. Compared to gradient-based optimization, stochastic approaches typically also achieve higher robustness to difficult reward landscapes due to their exploratory properties [5]. Yet, achieving reactive robot behavior is challenging as it requires solving trajectory optimization problems at frequencies which are sufficiently high for the task at hand. This issue can be alleviated in *Model Predictive Control (MPC)* settings by optimizing over a shorter receding time-horizon. Stochastic, gradient-free trajectory optimization, such as Model-Predictive Path Integral (MPPI) control [6] and the Cross-Entropy-Method (CEM) [4], combined with MPC, also known as *sampling-based MPC*, has proven state-of-the-art real-time performance on real robotic systems in challenging and dynamic environments [7], [8], [9]. However, these works still suffer from limited long-term anticipation, *e.g.*, getting stuck in front of obstacles, due to the optimization over a short receding horizon.

Motivated by the above, we propose *Via-Point-based Stochastic Trajectory Optimization (VP-STO)*, a framework that introduces the following contributions

1) A low-dimensional, time-continuous representation of trajectories in joint-space based on via-points that by-design respect kinodynamic constraints of the robot.
2) Stochastic via-point optimization, based on an evolutionary strategy, aiming at minimizing movement duration and task-related cost terms.
3) An MPC algorithm optimizing over the full horizon for real-time application in complex high-dimensional task settings, such as closed-loop object manipulation.

II. Related Work

In the context of closed-loop object manipulation with MPC, successful approaches to producing reactive robot behavior typically optimize in joint-space subject to kinodynamic constraints. While Fishman et al. use gradient-based MPC in order to find trajectories for human-robot handovers [10], a very recent approach named STORM [9] employed sampling-based MPC on robotic manipulation tasks. It is able to generate particularly smooth trajectories via low discrepancy action sampling, smooth interpolation and careful cost function design. Moreover, the parallelizability of sampling-based MPC is exploited by deploying the stochastic tensor optimization framework on a GPU. However, in contrast to our work, the approach relies on optimizing over a short receding horizon.

In the realm of time-parametrization of trajectories, most existing approaches fix the overall motion duration or do not specify it at all. For instance, the majority of MPC-based approaches only handle time implicitly via kinodynamic constraints. While the works of [11], [12] progress the state of the art in time-optimal MPC, their applicability to high-dimensional robotic systems yet is limited. In the context of motion planning, T-CHOMP [13] jointly optimizes a trajectory and the corresponding via-point timings. Yet, the total execution time is still fixed in advance. The way we approach the minimization of the movement duration is most similar to the work of [14]. However, in contrast to our work, their approach optimizes via-points and their timing separately.

III. Preliminaries: Trajectory Representation

The way we represent trajectories is based on previous work showing that the closed-form solution to the following optimization problem

$$
\begin{aligned}
\min \quad & \int_0^1 \boldsymbol{q}''(s)^\top \boldsymbol{q}''(s)ds \\
\text{s.t.} \quad & \boldsymbol{q}(s_n) = \boldsymbol{q}_n, \quad n = 1, ..., N \\
& \boldsymbol{q}(0) = \boldsymbol{q}_0, \boldsymbol{q}'(0) = \boldsymbol{q}_0', \ \boldsymbol{q}(1) = \boldsymbol{q}_T, \boldsymbol{q}'(1) = \boldsymbol{q}_T'
\end{aligned}
\tag{1}
$$

is given by cubic splines [15] and that it can be formulated as a weighted superposition of basis functions [16]. Hence, the robot's configuration is defined as $\boldsymbol{q}(s) = \boldsymbol{\Phi}(s)\boldsymbol{w} \in \mathbb{R}^D$, with D being the number of degrees of freedom. The matrix $\boldsymbol{\Phi}(s)$ contains the basis functions which are weighted by the vector $\boldsymbol{w}$[1]. The trajectory is defined on the interval $\mathcal{S} = [0, 1]$, while the time t maps to the phase variable $s = \frac{t}{T} \in \mathcal{S}$ with T being the total duration of the trajectory. Consequently, joint velocities and accelerations along the trajectory are given by $\dot{\boldsymbol{q}}(s) = \frac{1}{T}\boldsymbol{\Phi}'(s)\boldsymbol{w}$ and $\ddot{\boldsymbol{q}}(s) = \frac{1}{T^2}\boldsymbol{\Phi}''(s)\boldsymbol{w}$, respectively[2]. The basis function weights $\boldsymbol{w}$ include the trajectory constraints consisting of the boundary condition parameters $\boldsymbol{w}_{\text{bc}} = [\boldsymbol{q}_0^\top, \boldsymbol{q}_0'^\top, \boldsymbol{q}_T^\top, \boldsymbol{q}_T'^\top]^\top$ and N via-points the

---

[1] A more detailed explanation of the basis functions and their derivation can be found in the appendix of [16].

[2] We use the notation $f'(s)$ for derivatives w.r.t. s and the notation $\dot{f}(s)$ for derivatives w.r.t. t.

---

trajectory has to pass through $\boldsymbol{q}_{\text{via}} = [\boldsymbol{q}_1^\top, \ldots, \boldsymbol{q}_N^\top]^\top \in \mathbb{R}^{DN}$, such that $\boldsymbol{w} = [\boldsymbol{q}_{\text{via}}^\top, \boldsymbol{w}_{\text{bc}}^\top]^\top$. Throughout this paper, the via-point timings s_n are assumed to be uniformly distributed in $\mathcal{S}$. Note that boundary velocities map to boundary derivatives w.r.t. s by multiplying them with the total duration T, i.e., $\boldsymbol{q}_0' = T\dot{\boldsymbol{q}}_0$ and $\boldsymbol{q}_T' = T\dot{\boldsymbol{q}}_T$. Furthermore, the optimization problem in Eq. (1) minimizes not only the objective $\boldsymbol{q}''(s)$, but also the integral over accelerations, since $\boldsymbol{q}''(s) = T^2\ddot{\boldsymbol{q}}(s)$ and thus the objective $\int_0^1 \ddot{\boldsymbol{q}}(s)^\top\ddot{\boldsymbol{q}}(s)ds$ directly maps to $\frac{1}{T^4}\int_0^1 \boldsymbol{q}''(s)^\top\boldsymbol{q}''(s)ds$, corresponding to the control effort. It is minimal *iff* the objective in Eq. (1) is minimal. As a result, this trajectory representation provides a linear mapping from via points, boundary conditions and the movement duration to a time-continuous and smooth trajectory.

In the remainder of the paper, we exploit this explicit parameterization with via-points and boundary conditions by optimizing only the via-points while keeping the predefined boundary condition parameters fixed. Thus, we write the computation of the trajectory as a superposition of a via-point term and a boundary constraints term, i.e., $\boldsymbol{q}(s) = \boldsymbol{\Phi}_{\text{via}}(s)\boldsymbol{q}_{\text{via}} + \boldsymbol{\Phi}_{\text{bc}}(s)\boldsymbol{w}_{\text{bc}}$. The matrices $\boldsymbol{\Phi}_{\text{via}}(s)$ and $\boldsymbol{\Phi}_{\text{bc}}(s)$ are extracted from the basis function matrix $\boldsymbol{\Phi}(s)$.

IV. VP-STO: Via-Point-based Stochastic Trajectory Optimization

In the following, we introduce our stochastic trajectory optimization framework. The core idea is to find via-points $\boldsymbol{q}_{\text{via}}$ such that the synthesized trajectory minimizes a task-related objective, i.e.,

$$
\min_{\boldsymbol{q}_{\text{via}}} \quad c\left[\boldsymbol{q}(s), \dot{\boldsymbol{q}}(s), \ddot{\boldsymbol{q}}(s), T\right].
\tag{2}
$$

Based on these via-points, we efficiently synthesize high-quality trajectories, i.e., $\boldsymbol{q}_{\text{via}} \rightarrow \boldsymbol{\xi}$ with $\boldsymbol{\xi} = \{\boldsymbol{q}(s), \dot{\boldsymbol{q}}(s), \ddot{\boldsymbol{q}}(s), T\}$. We aim at synthesizing trajectories that *by-design* minimize task-agnostic objectives, i.e., *minimum time* and *smoothness*, and satisfy task-agnostic constraints, i.e., equality constraints on the *initial* and *final state* and inequality constraints on *joint-space velocities* and *accelerations*. We employ stochastic black-box optimization, namely *Covariance Matrix Adaptation (CMA-ES)* [5] to optimize for the via-points. As each trajectory constructed from the sampled via-points already provides the optimal solution to the optimization problem given in Eq. 1, the CMA-ES optimization in the low-dimensional via-point space is particularly fast, evaluating only high-quality trajectories. Moreover, with CMA-ES we are not only able to quickly converge to a local minimum, but to also leverage the exploration aspect of the evolutionary strategy (ES). In more detail, this nested optimization process, which is also illustrated in Fig. 2, comprises the following steps. First, a new population of M via-points $\boldsymbol{q}_{\text{via}}$ is sampled from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{\text{via}}, \boldsymbol{\Sigma}_{\text{via}})$. As $\boldsymbol{q}_{\text{via}}$ is a vector of the stacked via-points, note that $\boldsymbol{\mu}_{\text{via}} \in \mathbb{R}^{DN}$ and $\boldsymbol{\Sigma}_{\text{via}} \in \mathbb{R}^{DN \times DN}$. By taking M samples in this higher-dimensional space, instead of $M \cdot N$ samples for all via points separately
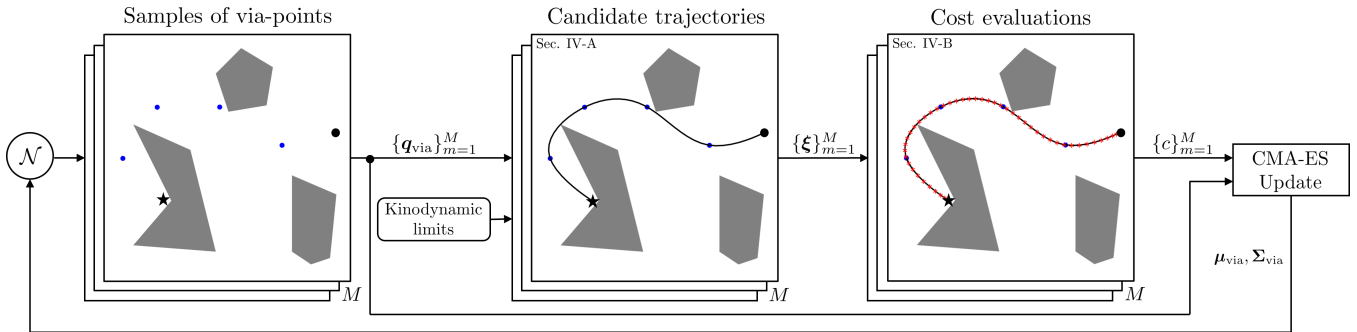
Fig. 2. An illustration of the via-point-based stochastic trajectory optimization loop. First, a new population of M via-points $\boldsymbol{q}_{\text{via}}$ is sampled from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{\text{via}}, \boldsymbol{\Sigma}_{\text{via}})$. Then, the sampled via-points are transformed into a population of candidate trajectories subject to kinodynamic limits. Next, the resulting trajectories are ranked according to their cost evaluations. Last, the parameters of the Gaussian sampling distribution are updated via CMA-ES using the cost rankings and the via-point sets themselves.

in the configuration space, we are able to sample M sets of correlated via-points. Then, as described in detail in Sec. IV-A, the sampled via-points are transformed into a population of candidate trajectories that are evaluated according to cost terms as outlined in Sec. IV-B. Finally, we use CMA-ES in order to update the parameters $\boldsymbol{\mu}_{\text{via}}, \boldsymbol{\Sigma}_{\text{via}}$ of the Gaussian distribution of via-points. This optimization setup enables us to find a valid local minimum or even the global minimum at rates sufficient for reactive robot behavior in closed-loop manipulation tasks, as we demonstrate in our experiments outlined in Section VI.

A. Synthesis of Kinodynamically Admissible Trajectories

In this section, we show how we translate the sampled via-points $\boldsymbol{q}_{\text{via}}$ into kinodynamically admissible trajectories. So far, the trajectory is implicitly given in phase space as described in Sec. III. Given the via-points and the boundary conditions $\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0, \boldsymbol{q}_T, \dot{\boldsymbol{q}}_T$, the computation of the explicit continuous trajectory only depends on the total movement duration T. It is determined by the dynamic limits on velocity $\dot{\boldsymbol{q}}_{\min}, \dot{\boldsymbol{q}}_{\max}$ and acceleration $\ddot{\boldsymbol{q}}_{\min}, \ddot{\boldsymbol{q}}_{\max}$ and is thus given as the minimum positive duration such that the resulting velocity and acceleration profiles satisfy the limits. We find a sufficient approximation for T by solving the above problem over a discrete set of K evaluation points $\{s_k\}_{k=0}^{K}$, uniformly distributed in the continuous phase space $\mathcal{S}$. For each evaluation point there exists a closed-form solution $T_k(\boldsymbol{q}_{\text{via}})$ that is computationally cheap to evaluate. We then pick the most conservative duration among the K solutions for T, i.e., $T(\boldsymbol{q}_{\text{via}}) = \max_k T_k(\boldsymbol{q}_{\text{via}})$, in order to make sure that the velocity and acceleration constraints are satisfied at all evaluation points. This procedure will result in trajectories where either the velocity or the acceleration profile reaches the limit in at least one evaluation point. Finally, having determined T, we are able to explicitly compute the kinodynamically admissible trajectory $\boldsymbol{\xi}$.

B. Cost Evaluation

Given the sampled and synthesized population of trajectories, we evaluate the performance, i.e., the cost c of each trajectory, independently. The gradient-free optimizer allows for sharp cost function profiles, e.g., trajectory constraints expressed through discontinuous barrier functions (cf. Sec. VI

for examples). We approximate $c(\boldsymbol{\xi})$ by sampling the given trajectory with a predefined resolution Δs in the phase space $\mathcal{S}$ and accumulating the costs at these K evaluation points. In the time domain this can still map to varying resolutions of individual trajectories, as $\Delta t = T\Delta s$. Note that the evaluation points at s_k are not equivalent to the via-points at s_n, as depicted in Fig. 2. The resolution of s_k can be higher than that of s_n in order to have a better approximation of the trajectory cost while keeping the actual optimization variable $\boldsymbol{q}_{\text{via}}$ low-dimensional.

V. ONLINE VP-STO (MPC)

In order to perform closed-loop control via continuous online re-optimization, we embed the *VP-STO* framework into an MPC algorithm. In this online setting, the main focus lies on rapidly finding valid movements connecting the current robot state $\boldsymbol{q}, \dot{\boldsymbol{q}}$ with a goal state $\boldsymbol{q}_T, \dot{\boldsymbol{q}}_T$ and re-optimizing them at a sufficient rate $f_{\text{mpc}} = \frac{1}{\Delta t_{\text{mpc}}}$. Algorithm 1 outlines a single MPC step that, given the current robot state, attempts to find an optimal *full-horizon* trajectory and to extract a short-horizon reference to be tracked by a lower-level impedance controller. The details of the algorithm will be outlined in the remainder of this section.

In the online setting the number of via-points N used to parameterize the trajectory plays an important role. A large number of via-points can capture highly complex movements and may find more optimal solutions. However, it also implies a higher-dimensional decision space which increases the computational complexity of the optimization loop. Consequently, a particular focus within the MPC algorithm lies on the selection of N.

A. No-Via-Point Trajectory for Stopping Behavior

VP-STO is based on optimizing the locations of a given number of via-points. However, the trajectory synthesis, described in Sec. IV-A, also works without any via-points, i.e., $N=0$. The resulting trajectory connects the current robot state and the desired state by a third-order polynomial that minimizes the smoothness objective in Eq. (1) and satisfies the kinodynamic limits. As this no-via-point trajectory is a unique solution, it can not account for any other movement objectives, e.g., to avoid collisions. Yet, the advantage is a cheap-to-construct trajectory that has no stochasticity, which

Algorithm 1: Online VP-STO: i-th MPC Step

Input: $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{q}_T, \dot{\boldsymbol{q}}_T, \dot{\boldsymbol{q}}_{\min}, \dot{\boldsymbol{q}}_{\max}, \ddot{\boldsymbol{q}}_{\min}, \ddot{\boldsymbol{q}}_{\max}, \Delta t_{\text{mpc}}, T_{\text{stop}}$
Output: Short-horizon reference $\boldsymbol{q}_{\text{d}}(t), \dot{\boldsymbol{q}}_{\text{d}}(t), \ddot{\boldsymbol{q}}_{\text{d}}(t)$

$t_{\text{optimize}} \leftarrow 0$
$\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0 \leftarrow \boldsymbol{q}, \dot{\boldsymbol{q}}$
$\boldsymbol{\xi}_{\text{direct}} \leftarrow \text{synthesize}()$ // V-A
if $\boldsymbol{\xi}_{\text{direct}}$ *is valid* **and** $\boldsymbol{\xi}_{\text{direct}}$ *is shorter than* T_{stop} **then**
    | $\boldsymbol{\xi}_i^* \leftarrow \boldsymbol{\xi}_{\text{direct}}$
else
    | **if** $\boldsymbol{\xi}_{i-1}^*$ *is valid* **then**
    | | ${}^{0}\boldsymbol{\mu}_{\text{via}}, {}^{0}\boldsymbol{\Sigma}_{\text{via}}, N \leftarrow \text{warmStart}(\boldsymbol{\xi}_{i-1}^*)$ // V-B
    | **else**
    | | ${}^{0}\boldsymbol{\mu}_{\text{via}}, {}^{0}\boldsymbol{\Sigma}_{\text{via}}, N \leftarrow \text{exploreInit}()$ // V-B
    | **end**
    | $j \leftarrow 0$
    | **while** $t_{\text{optimize}} < \Delta t_{\text{mpc}}$ **do**
    | | $\{\boldsymbol{q}_{\text{via}}\}_{m=1}^{M} \leftarrow \text{sample}\left({}^{j}\boldsymbol{\mu}_{\text{via}}, {}^{j}\boldsymbol{\Sigma}_{\text{via}}\right)$ // V-C
    | | $\{\boldsymbol{\xi}\}_{m=1}^{M} \leftarrow \text{synthesize}\left(\{\boldsymbol{q}_{\text{via}}\}_{m=1}^{M}\right)$ // IV-A
    | | $\{c\}_{m=1}^{M} \leftarrow \text{evaluate}\left(\{\boldsymbol{\xi}\}_{m=1}^{M}\right)$ // IV-B
    | | $\boldsymbol{\mu}_{\text{via}}^{j+1}, \boldsymbol{\Sigma}_{\text{via}}^{j+1} \leftarrow \text{sep-CMA-ES}\left(\{\boldsymbol{q}_{\text{via}}, c\}_{m=1}^{M}\right)$
    | | $j \leftarrow j+1$
    | **end**
    | $\boldsymbol{\xi}_i^* \leftarrow \text{synthesize}\left(\boldsymbol{\mu}_{\text{via}}^{j}\right)$
end
$\boldsymbol{q}_{\text{d}}(t), \dot{\boldsymbol{q}}_{\text{d}}(t), \ddot{\boldsymbol{q}}_{\text{d}}(t) \leftarrow \text{shortHorizon}(\boldsymbol{\xi}_i^*)$ // V-D
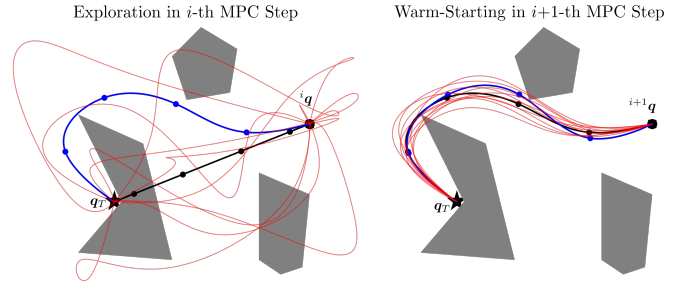

Fig. 3. An illustration of the stochastic optimization process within the proposed MPC algorithm. ***Left:*** In the *exploration* mode, trajectories are sampled and synthesized with a large initial variance in order to discover valid solutions. ***Right:*** If a valid solution is available from the previous MPC step, we *warm-start* the optimization by shifting the solution and sampling from a lower-variance initial distribution. All sampled trajectories are shown in red. The initial guesses ${}^{0}\boldsymbol{\mu}_{\text{via}}$ of an MPC step are depicted by the black solid lines, while the blue trajectories illustrate the mean solution ${}^{20}\boldsymbol{\mu}_{\text{via}}$ after 20 optimization iterations.

is useful for driving the robot to the target configuration and stopping with zero velocity. Therefore, at the beginning of each optimization cycle, we first check if this simple direct trajectory is valid, *e.g.*, collision-free, and if the corresponding duration of the movement is below the user-defined threshold T_{stop}. By setting the threshold rather small, we let the mechanism take over towards the final part of the total trajectory to achieve robust stopping behavior for reaching the goal. If the direct solution is not used, we perform a *VP-STO* optimization cycle.

B. Initialization: Exploration vs. Warm-Starting

The use of an evolutionary optimization strategy, such as CMA-ES, allows us to initialize the optimization not only with an initial guess of the via-points $\boldsymbol{\mu}_{\text{via}}$, but also to set the corresponding initial variance $\boldsymbol{\Sigma}_{\text{via}}$ as an estimate of how certain we are about the initial solution. The initial variance can thus be interpreted as an exploration parameter influencing how the very first population of candidate trajectories will be sampled. Therefore, in each MPC step we use two possible modes on how to initialize these parameters. The effects of each mode on the resulting candidate trajectories are shown in Fig. 3.

Exploration. If a MPC step was not successful in finding a valid trajectory, the successive MPC step will be used to *explore* a larger area of the trajectory space to ideally discover a valid solution, as can be seen from the sampled trajectories in the left of Fig. 3. We initialize the mean solution $\boldsymbol{\mu}_{\text{via}}$ with a naive straight-line guess with high

uncertainty, *i.e.*, large diagonal values of $\boldsymbol{\Sigma}_{\text{via}}$. The number of via-points used to parameterize the trajectory is set to $N = N_{\max}$. $N_{\max}$ needs to be specified by the user and depends on the complexity of the task, as well as on the available computational resources.

Warm-Starting. If a valid solution was found in a MPC step, we shift the solution forward in time and use it to *warm-start* the mean $\boldsymbol{\mu}_{\text{via}}$ in the successive MPC step, potentially further improving the current solution. In this case, we initialize the covariance matrix $\boldsymbol{\Sigma}_{\text{via}}$ with low values on the diagonal as we are more certain about the proximity of the current solution to a valid local minimum, as can be seen on the right of Fig. 3. In order to determine the number of via-points N for the successive MPC step, we use the movement duration of the current solution as a proxy for how complex the remainder of the movement will be. We therefore set $N = \max(1, \min(\lceil \alpha T \rceil, N_{\max}))$, where T is the total duration of the current solution and α a user-defined scaling parameter.

C. Efficient Gaussian Sampling of Smooth Trajectories through Covariance Matrix Decomposition

For the sake of computational efficiency and linear scalability to high-DoF systems in our MPC solver, we use a variant of CMA-ES that iterates on diagonal covariance matrices instead of full covariance matrices, namely *sep-CMA-ES* [17]. However, a diagonal covariance matrix does not capture the correlations between the sampled via-points that are important for sampling smooth trajectories. We counteract this disadvantage by using a Cholesky factorization of the covariance matrix, such that $\boldsymbol{\Sigma}_{\text{via}} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\top}$, where the diagonal matrix $\boldsymbol{D} = \text{diag}(\boldsymbol{\sigma}_{\text{via}})$ is subject to iterative optimization through sep-CMA-ES. This renders our algorithm to a computational complexity of $\mathcal{O}(ND)$, with N being the number of via-points and D the DoF of the robot, instead of $\mathcal{O}((ND)^2)$ in the case of the full covariance matrix. The lower triangular matrix $\boldsymbol{L}$ is computed offline as the Cholesky decomposition of a constant covariance matrix

$$\boldsymbol{\Sigma}_{\text{smooth}} = \boldsymbol{L}\boldsymbol{L}^{\top} = \left(\int_0^1 \boldsymbol{\Phi}_{\text{via}}''(s)^{\top} \boldsymbol{\Phi}_{\text{via}}''(s) ds\right)^{-1}, \quad (3)$$

that is derived from a probability distribution of smooth trajectories, *i.e.*, $p_{\text{smooth}}(\boldsymbol{q}_{\text{via}}, \boldsymbol{w}_{\text{bc}}) \propto \exp(-c_{\text{effort}})$, with $c_{\text{effort}} = \frac{1}{2} \int_0^1 \boldsymbol{q}''(s)^\top \boldsymbol{q}''(s) ds$.

D. Impedance Control

At a lower control level, we deploy an impedance controller that runs at a control rate of 1 kHz, which requires a finely sampled reference trajectory. Due to our time-continuous representation of the optimized trajectory, we can sample configurations from it with arbitrarily small temporal resolution. Each MPC step yields an optimized trajectory $\boldsymbol{\xi}_i^*$, from which we extract a position-, velocity- and acceleration-reference enabling the robot to track the current movement plan.

VI. EXPERIMENTS

We evaluate the effectiveness and performance of the *VP-STO* framework in simulation, as well as in real-world experiments with a Franka Emika robot arm.

A. Simulation

We begin by evaluating our framework in an offline planning setting for a 2D point mass in a cluttered toy environment adopted from [9]. In this experiment, we run *VP-STO* (cf. Sec. IV) for 100 times with a straight-line initialization. The left plot in Fig. 4 shows the resulting 100 trajectories after convergence. The majority of the found solutions converged to 3 valid local optima, *i.e.*, 28 solutions to the red, 69 to the blue and one to the green trajectory. Only 2 runs produce a non-valid solution, shown in yellow. We note here that gradient-based trajectory optimization methods given the straight-line initial guess in such a challenging environment would only converge to this non-valid local optimum. Moreover, this also shows that the choice of CMA-ES as a solver for our framework helps to converge to the present local optima with negligible error, despite the stochasticity in the sampling of the via-points. Last, the corresponding velocity and acceleration profiles (only shown for the valid solutions), depicted on the right of Fig. 4, reflect the timing-optimal property of the generated trajectories. After applying maximum acceleration at the start of the movement, the robot moves at maximum speed within the limits before it again applies the maximum acceleration to stop at the goal. This implies that our framework generates trajectories that not only respect the given dynamic limits, but also exploits them in the spirit of timing-optimality.

For the online setting, as described in Sec. V, we compare *VP-STO* to *STORM* [9], which we consider as state-of-the-art in sampling-based MPC for producing reactive robot behavior. Again using the scenario from above, we run 5 experiments in which we deploy *VP-STO* within the MPC-algorithm (cf. Alg. 1). The resulting trajectories are shown in blue in Fig. 5 alongside the 5 solutions in red generated by *STORM*. It can be seen that *STORM* is not able to reach the goal. Especially, due to the short-horizon optimization scheme, the robot first follows the path with the shortest distance towards the goal while not being able to anticipate
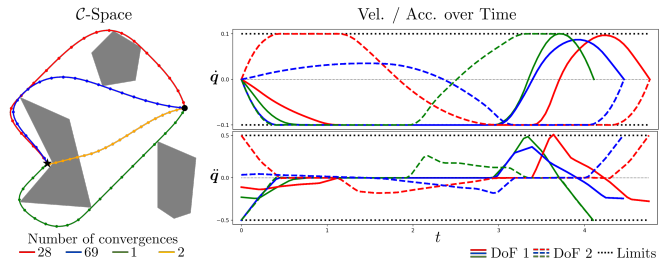

Fig. 4. **Offline VP-STO.** *Left:* The resulting trajectories from 100 experiment runs when initializing with a straight-line guess between the start position (black circle) and the target position (black asterisk). The number of convergence indicates how often *VP-STO* converged to the corresponding color-coded solution. *Right:* The velocity and acceleration profiles for each degree of freedom corresponding to the valid solutions on the left.
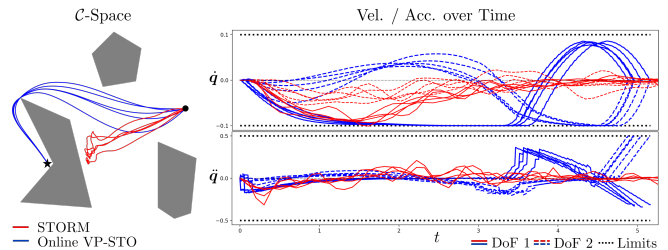

Fig. 5. **Online VP-STO (MPC).** *Left:* The trajectories taken by the robot when deploying *VP-STO* in an MPC setting (blue), as opposed to using *STORM* [9] (red). *Right:* The velocity and acceleration profiles for each degree of freedom corresponding to the found solutions on the left.

moving around the obstacle early enough. Therefore, it gets stuck in front of the obstacle. In contrast, *Online VP-STO* produces solutions which allow the robot to smoothly navigate to the goal, while exploiting its velocity and acceleration limits. The given setting and experiment emphasizes the advantage of our efficient formulation which allows us to always optimize over the full horizon.

B. Real-World Experiments

We demonstrate *VP-STO* on a real robot using the manipulation scenarios in Fig. 1: a *pick-and-place* and a *box pushing* task. We increase the complexity of both scenarios by disturbing the robot and the target objects. This requires a fast feedback loop provided by *Online VP-STO*.

Setup. Both experiments are performed on a Franka Emika robot arm. The framework was run on Ubuntu 20.04 with an Intel Core i7-8700 CPU@3.2GHz and 16GB of RAM. The poses of the objects were tracked with a Vicon motion capture system and post-processed with an extended Kalman filter. The MPC steps are executed at a fixed control rate (specified below). In a single MPC step, we run optimization iterations until the next MPC step starts.

a) Pick-and-Place: First, we consider a *pick-and-place* scenario under human intervention. The robot's task is to grasp a pin, *i.e.*, the picking phase, and to place it in an upright position in a given target location in the workspace, *i.e.*, the placing phase. In the picking phase, the pin can be either handed over to the robot in arbitrary poses or the robot needs to pick it up from the table. This phase requires real-time collision avoidance in narrow configuration passages, *i.e.*, the robot has to avoid collisions between its hand, including the

fingers, and the pin while reaching a configuration where the hand encloses the pin. For the grasp pose, we run a separate pose optimization process in parallel to *VP-STO*, providing the final robot configuration $\boldsymbol{q}_T$. After a successful grasp, the robot continues with the placing phase. The challenge here is that the pin might still move within the gripper due to its own weight or due to interference from a user. Consequently, feedback of the current pin pose is needed to avoid collisions between the pin and the environment and to correctly place the pin. We parameterize the sampled trajectories with a maximum number of via-points $N_{\max}=4$ and $\alpha=2$. *VP-STO* replans with a rate of 12.5 Hz.

b) Box Pushing: In the second scenario, we address the task of planning and control through physical contacts, *i.e.*, the robot is supposed to push a box towards a moving target position. Such a task requires the robot to deliberately make and break contacts, which is subject to discontinuous cost-landscapes. Here, we exploit the presented trajectory parameterization by setting the final robot configuration $\boldsymbol{q}_T$ of each MPC step such that the end-effector moves towards the center of the box. This enforces all sampled candidate trajectories to make contact with the box. The point of contact and the resulting dynamics of the box depends on the location of the via-points which are subject to minimizing the distance between the box position and the target.For the sake of fast simulations of the contact dynamics, we use a quasi-dynamic model for the box dynamics parallel to the table surface. *VP-STO* is executed with a constant number of via-points $N=3$ at a control rate of 20 Hz.

Cost Terms. We begin with the task-agnostic terms and conclude with more task-specific terms.

Movement Duration: The movement duration is used explicitly as part of the cost function in order to minimize the time needed for the remaining robot movement.

Smoothness: In order to optimize not only for fast, but also efficient movements, we use the same metric as in Eq. (1) as the smoothness cost term.

Joint Limit Avoidance: For keeping the robot configuration inside the joint angle limits, we deploy a discontinuous metric that accounts for joint limit violations, *i.e.*,

$$c_{\mathrm{jla}}(q) = \begin{cases} 1 + q - q_{\max}, & \text{if } q \geq q_{\max} \\ 1 + q_{\min} - q, & \text{if } q \leq q_{\min} \\ 0, & \text{otherwise} \end{cases} \qquad (4)$$

We consider a trajectory to be invalid if it results in a joint limit violation, *i.e.*, $q \geq q_{\max}$ or $q \leq q_{\min}$.

Collision Avoidance: In order to efficiently evaluate the validity of a trajectory regarding collisions between the robot and the environment, we perform binary collision checks for each configuration evaluated along the trajectory, instead of computing a distance between two geometries. Thus, the collision cost for a single trajectory is equal to the number of evaluation points that are in collision. Similarly to the joint limit avoidance cost, we consider a trajectory to be invalid if it results in a collision.

Pushing Progress: In the case of a pushing task, we further require a cost term that rewards trajectories which let the robot move the box closer to the current desired target $\boldsymbol{x}_{\mathrm{box}}^{\mathrm{des}}$. We evaluate the pushing progress of a single trajectory by first simulating the contact dynamics that result in a trajectory of the box $\boldsymbol{x}_{\mathrm{box}}(t)$; and then computing the box position error at the beginning $e_{\mathrm{box},0} = ||\boldsymbol{x}_{\mathrm{box}}(0) - \boldsymbol{x}_{\mathrm{box}}^{\mathrm{des}}||_2^2$ and at the end $e_{\mathrm{box},T} = ||\boldsymbol{x}_{\mathrm{box}}(T) - \boldsymbol{x}_{\mathrm{box}}^{\mathrm{des}}||_2^2$ of the robot movement. The final pushing progress cost is given by $c_{\mathrm{push}}(\boldsymbol{\xi}) = \exp(e_{\mathrm{box},T} - e_{\mathrm{box},0})$. Additionally, we consider trajectories that move the box away from the target, *i.e.*, $e_{\mathrm{box},T} \geq e_{\mathrm{box},0}$, to be invalid. In that case, the *exploration* mode in the next MPC step (cf. Sec. V-B) is triggered.

Results. First, we note that throughout the experiments, the robot did not collide with any objects in the workspace and did not violate the joint limits. When the experimenter perturbs the robot, *i.e.*, disturbing it through physical interaction or pulling the pin out of the gripper, the robot is compliant and adapts its motion. In the pick-and-place scenario, it robustly picked up the pin from various locations in the workspace, including handovers by the experimenter; and placed it at the desired target location in all runs. In the box pushing scenario, the robot manages to find pushing motions from arbitrary configurations and box locations and to eventually push the box into the target. We note, however, that some changes of the target location resulted in the robot not finding a valid pushing motion quickly enough, which in turn made the robot push the box out of the workspace. This could only be recovered by the experimenter. Recordings of the experiments and additional material can be found in the accompanying video to this paper and on the dedicated website https://sites.google.com/oxfordrobotics.institute/vp-sto.

VII. Conclusion

We presented a motion optimization framework that is able to generate reactive and yet smooth and efficient robot behavior for complex high-dimensional robot tasks. In contrast to standard trajectory optimization techniques, sampling-based and gradient-based, our framework outputs trajectories which not only optimize over space but also time. Moreover, due to the full-horizon optimization in an MPC-setting, it is particularly suitable for closed-loop manipulation tasks that demand for continuous re-planning and feedback. We successfully demonstrate this in two real-world experiments on a Franka Emika robot arm, *i.e.*, a pick-and-place and a box-pushing scenario.

We wish to extend and improve our work by considering the following points. First, the number via-points to sample yet is subject to heuristic tuning. In general, with increasing movement complexity more via-points are needed at the cost of higher computational complexity. Future work should make the selection of this hyper-parameter more intuitive. And second, we would like to further increase the robustness of *VP-STO* by considering uncertainties in the interaction between the robot and its environment. This includes to explore stochastic roll-outs in the cost evaluation.

REFERENCES

[1] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[2] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

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

[4] R. Y. Rubinstein, "The Cross-Entropy Method for Combinatorial and Continuous Optimization," *Methodology and Computing in Applied Probability*, vol. 1, no. 2, pp. 127–190, 1999.

[5] N. Hansen, "The CMA evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[6] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[7] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1433–1440.

[8] M. Bangura and R. Mahony, "Real-time Model Predictive Control for Quadrotors," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 773–11 780, 2014.

[9] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. Ratliff, D. Fox, F. Ramos, and B. Boots, "STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation," in *Conference on Robot Learning (CoRL)*, 2022, pp. 750–759.

[10] A. Fishman, C. Paxton, W. Yang, D. Fox, B. Boots, and N. Ratliff, "Collaborative interaction models for optimized human-robot teamwork," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 221–11 228.

[11] L. Van den Broeck, M. Diehl, and J. Swevers, "Model predictive control for time-optimal point-to-point motion control," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 2458–2463, 2011.

[12] C. Rosmann, A. Makarow, F. Hoffmann, and T. Bertram, "Time-Optimal nonlinear model predictive control with minimal control interventions," in *Proc. IEEE Conference on Control Technology and Applications (CCTA)*, 2017, pp. 19–24.

[13] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, "Space-time functional gradient optimization for motion planning," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6499–6506.

[14] M. Toussaint, J. Harris, J.-S. Ha, D. Driess, and W. Hönig, "Sequence-of-Constraints MPC: Reactive Timing-Optimal Control of Sequential Manipulation," *arXiv preprint arXiv:2203.05390*, 2022.

[15] Z. Zhang, J. Tomlinson, and C. Martin, "Splines and linear control theory," *Acta Math. Appl*, vol. 49, pp. 1–34, 1997.

[16] J. Jankowski, M. Racca, and S. Calinon, "From Key Positions to Optimal Basis Functions for Probabilistic Adaptive Control," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3242–3249, 2022.

[17] R. Ros and N. Hansen, "A simple modification in cma-es achieving linear time and space complexity," in *International conference on parallel problem solving from nature*. Springer, 2008, pp. 296–305.

A. Efficient Gaussian Sampling of Smooth Trajectories through Covariance Matrix Decomposition (Extended)

In the proposed MPC algorithm, the number of optimization iterations ran in a single step is limited by the desired control rate and the computational resources. We have identified two modifications of the algorithm that drastically reduce the cost of the mean trajectory after a given optimization time budget.

First, we replace the standard CMA-ES optimization by sep-CMA-ES, a variant that iterates only on diagonal covariance matrices. This improves the computational complexity from $\mathbb{O}(N^2 D^2)$ (CMA-ES) to $\mathbb{O}(ND)$ (sep-CMA-ES), meaning that the computational load of sampling from and updating the covariance matrix scale linearly with the number of via-points N and the DoF D of the robot.

Second, instead of initializing the covariance matrix $\boldsymbol{\Sigma}_{\text{via}}$ with an identity matrix scaled by a single scalar, we start the optimization with a covariance matrix that captures smoothness correlations between via-points. This modification can be justified by a probabilistic view on stochastic optimization problems, *i.e.*, rather than minimizing the expected cost $c(\boldsymbol{q}_{\text{via}})$ as in (2), we aim at maximizing a probability $p(\boldsymbol{q}_{\text{via}}) \propto e^{-c(\boldsymbol{q}_{\text{via}})}$. It is easy to show that both optimization problems have equivalent optima. In fact, CMA-ES attempts to locally approximate the generally intractable probability distribution $p(\boldsymbol{q}_{\text{via}})$ by a Gaussian distribution in each iteration. If the trajectory cost is given as a sum of multiple objectives, *i.e.*, $c(\boldsymbol{q}_{\text{via}}) = \sum_i c_i(\boldsymbol{q}_{\text{via}})$, the corresponding probability distribution can be written as a product of multiple probability distributions, *i.e.*, $p(\boldsymbol{q}_{\text{via}}) \propto \prod_i e^{-c_i(\boldsymbol{q}_{\text{via}})}$. A smoothness metric is typically part of the cost function, in our case we use

$$
\begin{aligned}
c_{\text{smooth}}(\boldsymbol{q}_{\text{via}}) &= \frac{1}{2} \int_0^1 \boldsymbol{q}''(s)^\top \boldsymbol{q}''(s) ds \\
&= \frac{1}{2} \boldsymbol{w}^\top \int_0^1 \boldsymbol{\Phi}''(s)^\top \boldsymbol{\Phi}''(s) ds \boldsymbol{w}.
\end{aligned} \tag{5}
$$

Since $\boldsymbol{w} = [\boldsymbol{q}_{\text{via}}^\top, \boldsymbol{w}_{\text{bc}}^\top]^\top$, the smoothness cost term can be exactly represented by a Gaussian distribution, *i.e.*,

$$
p_{\text{smooth}}(\boldsymbol{q}_{\text{via}}, \boldsymbol{w}_{\text{bc}}) = \mathcal{N}\left(\boldsymbol{0}, \int_0^1 \boldsymbol{\Phi}''(s)^\top \boldsymbol{\Phi}''(s) ds\right). \tag{6}
$$

We condition the joint distribution on the given boundary constraints to obtain the corresponding distribution of the via-points $p_{\text{smooth}}(\boldsymbol{q}_{\text{via}}|\boldsymbol{w}_{\text{bc}}) = \mathcal{N}(\boldsymbol{\mu}_{\text{via, smooth}}, \boldsymbol{\Sigma}_{\text{via, smooth}})$ with

$$
\begin{aligned}
\boldsymbol{\mu}_{\text{via, smooth}} &= \boldsymbol{\Sigma}_{\text{via, smooth}} \int_0^1 \boldsymbol{\Phi}''_{\text{via}}(s)^\top \boldsymbol{\Phi}''_{\text{bc}}(s) ds \, \boldsymbol{w}_{\text{bc}} \\
\boldsymbol{\Sigma}_{\text{via, smooth}} &= \left(\int_0^1 \boldsymbol{\Phi}''_{\text{via}}(s)^\top \boldsymbol{\Phi}''_{\text{via}}(s) ds\right)^{-1}.
\end{aligned} \tag{7}
$$

By initializing the covariance matrix with $\boldsymbol{\Sigma}_{\text{via, smooth}}$, the very first population of via-points that is evaluated in an optimization loop is consequently sampled from p_{smooth}. This

can be interpreted as an informed warm-starting of the covariance matrix in a CMA-ES loop.

In order to integrate the off-diagonal structure of $\boldsymbol{\Sigma}_{\text{via, smooth}}$ with the diagonal covariance matrix $\text{diag}(\boldsymbol{\sigma}_{\text{via}})$ that is updated by sep-CMA-ES, we assemble the final covariance matrix by a Cholesky factorization, *i.e.*, $\boldsymbol{\Sigma}_{\text{via}} = \boldsymbol{L}\text{diag}(\boldsymbol{\sigma}_{\text{via}})\boldsymbol{L}^\top$. The off-diagonal structure is imposed by the lower triangular matrix $\boldsymbol{L}$ that is given by the Cholesky decomposition of the smoothness covariance, such that $\boldsymbol{\Sigma}_{\text{via, smooth}} = \boldsymbol{L}\boldsymbol{L}^\top$.

B. Ablation Studies

In the paper, we present design choices that we want to further justify via ablation studies.

1) Impact of the Number of Via-points: In this ablation study, we investigate the impact of the number of via-points used to represent the robot movement. This hyper-parameter has a high impact on the overall framework performance. On one hand, it directly sets the dimensionality of the optimization problem to solve; on the other hand, it directly spans the space of movements that can be synthesized. From an optimization perspective, tuning the number of via-points gives us an intuitive way of increasing/decreasing resources on an optimization result with a decreasing/increasing cost. We illustrate this relationship in Fig. 6, where we let a 1D double-integrator move from $q_0 = 0.0$, $\dot{q}_0 = 0.0$ to $q_T = 1.0$, $\dot{q}_T = 0.0$ in minimal time, subject to a maximum velocity $|\dot{q}| < 0.1$ and an acceleration limit $|\ddot{q}| < 0.2$; with a varying number of via-points. This time-optimal control problem is known to be solved by a bang-bang acceleration profile, such that we know the analytic limit of the minimal time to be $c_{\text{bang-bang}} = T_{\text{bang-bang}} = 10.5$, which is depicted as dashed black line in the upper-left plot. We observe that the solution cost exponentially converges to $c_{\text{bang-bang}}$ as we increase the number of via-points. The lower-left plot shows the number of CMA-ES-iterations required to converge as a function of the number of via-points. Here, we detect convergence if $|c_k - c_{k-1}| < 10^{-6}$ in the k-th iteration. Interestingly, the number of iterations grows linearly with the number of via-points. Note that this does not mean that the computational cost grows linearly with the number of via-points, since the computational cost for a single iteration is either linear (sep-CMA-ES) or quadratic (CMA-ES) in the number of via-points. Nevertheless, those results motivate to use a low number of via-points as with a growing number of via-points, the benefit of adding a via-point is not worth the extra computational cost.

2) Impact of the Cholesky Factorization of the Covariance Matrix: In this ablation study, we look at a 2D minimal-time planning problem including an obstacle that is to be avoided. We fix the number of via-points to $N = 6$ and set up four different optimization loops that are supposed to solve the same problem. Each setup uses either CMA-ES or sep-CMA-ES and runs with or without the Cholesky factorization of the covariance matrix as described in Sec. V-C. For comparison, we look at the cost evolution over the number of iterations. The dashed black line in all plots
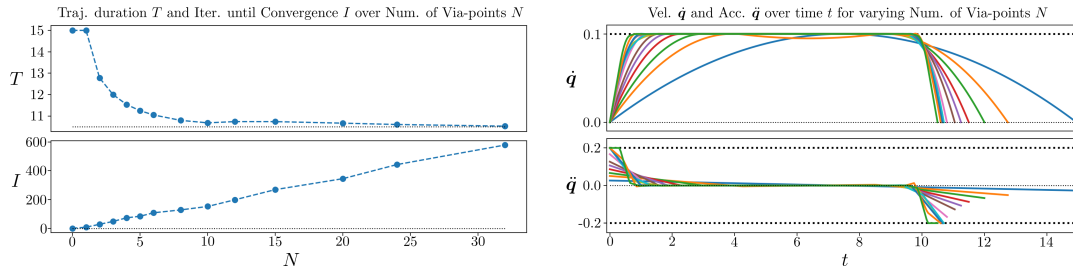
Fig. 6. A study of the impact of the number of via-points in a 1D time-optimization problem. **Top-Left:** Impact on the resulting movement duration. The dotted black line illustrates the duration of the optimal *bang-bang* solution. **Bottom-Left:** Impact on the number of iterations required until convergence. **Right:** Velocity and acceleration profiles for evaluated numbers of via-points.
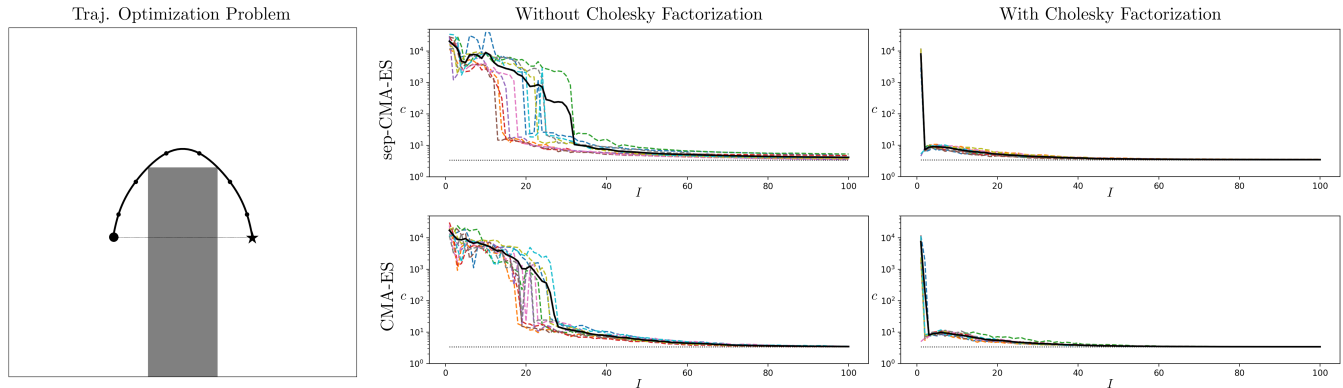

Fig. 7. A study of the impact of the Cholesky factorization of the Covariance Matrix $\mathbf{\Sigma}_{\mathrm{via}}$ in a 2D time-optimization problem with obstacle avoidance. **Left:** The configuration space including the obstacle in gray, the initial guess as dashed line, and the optimal solution around the obstacle as solid line together with the corresponding via-points as circles. **Center:** The via-point covariance matrix is explicitly updated, *i.e.*, $\mathbf{\Sigma}_{\mathrm{via}} = \mathbf{\Sigma}_{\mathrm{CMA}}$. **Right:** The via-point covariance matrix is updated through a Cholesky factorization, *i.e.*, $\mathbf{\Sigma}_{\mathrm{via}} = \mathbf{L}\mathbf{\Sigma}_{\mathrm{CMA}}\mathbf{L}^{\top}$. **Top:** sep-CMA-ES iterates on diagonal covariance matrices only, *i.e.*, $\mathbf{\Sigma}_{\mathrm{CMA}} = \mathrm{diag}(\boldsymbol{\sigma}_{\mathrm{CMA}})$, with linear computational complexity $\mathcal{O}(ND)$. **Bottom:** CMA-ES iterates on full covariance matrices $\mathbf{\Sigma}_{\mathrm{CMA}}$ with quadratic computational complexity $\mathcal{O}(N^2 D^2)$.

(except for the left-hand plot) indicates the minimum cost measured in any experiment. Note also the jump in all the cost profiles from $\approx 10^3 - 10^4$ to $\approx 10^0 - 10^1$, which reflects if the updated solution is collision-free. We observe that the choice of CMA-ES vs. sep-CMA-ES does not have a substantial impact on the cost evolution for this particular problem, indicating that it is justified to use sep-CMA-ES with linear complexity. However, we observe a substantial impact when using the presented Cholesky factorization, imposing smoothness on the candidate trajectories. In all experiments using the Cholesky factorization, it converged to a collision-free solution after 3 iterations at maximum. This is an especially important result justifying the use of the Cholesky factorization inside the MPC loop, as the real-time requirements limit the number of iterations.