Sequence-of-Constraints MPC: Reactive Timing-Optimal Control of Sequential Manipulation

Marc Toussaint[1,2], Jason Harris[1], Jung-Su Ha[1], Danny Driess[1,2], Wolfgang Hönig[1]

***Abstract*— Task and Motion Planning has made great progress in solving hard sequential manipulation problems. However, a gap between such planning formulations and control methods for reactive execution remains. In this paper we propose a model predictive control approach dedicated to robustly execute a single sequence of constraints, which corresponds to a discrete decision sequence of a TAMP plan. We decompose the overall control problem into three sub-problems (solving for sequential waypoints, their timing, and a short receding horizon path) that each is a non-linear program solved online in each MPC cycle. The resulting control strategy can account for long-term interdependencies of constraints and reactively plan for a timing-optimal transition through all constraints. We additionally propose phase backtracking when running constraints of the current phase cannot be fulfilled, leading to a fluent re-initiation behavior that is robust to perturbations and interferences by an experimenter.**

I. Introduction

Task and Motion Planning (TAMP) [1] has made great progress in recent years in solving hard sequential manipulation problems. However, to bring such plans to reactive execution remains a challenge and typically relies on a predefined set of controller primitives to execute individual actions [2]. As an example for robust execution, Boston Dynamics has been showcasing several demonstrations where humans massively perturb a linear plan, not only in locomotion but also sequential manipulation.[1] Candidates to design such highly reactive and robust behavior include hierarchies of convergent controllers (funnels) [3], and the use of state machines to orchestrate transitioning between control modes [4], [5]. However, the latter again relies on a predefined set of controllers per action. We aim for a general approach to more directly translate a TAMP plan to a reactive execution strategy.

In the context of trajectory optimization, the translation from trajectory planning to control is immediate via repeated online planning, in particular model-predictive control (MPC) approaches. Here, the same underlying problem formulation (in terms of the path objectives) equally allows to derive optimal plans as well as robust control strategies. Given existing optimization-based formulations of TAMP [6], [7], [8], it might seem that reactive execution could easily be derived from the same underlying optimality formulation

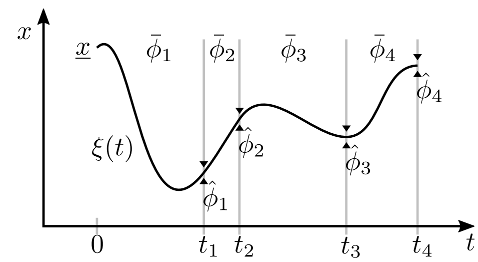

Fig. 1: Illustration of the Sequence-of-Constraints Optimal Control problem (1): We optimize a path $\xi(t)$ and timings $t_{1:K}$ subject to waypoint constraints $\hat\phi_{1:K}$ and running constraints $\bar\phi_{1:K}$. Constraints concern any non-linear differentiable features ϕ of the system configuration and/or velocity. The system configuration $\xi(t)$ includes manipulation dofs such as relative grasp, push and placement poses, which imply interdependencies between waypoint constraints (see Section III-B).

by a standard MPC approach [9]. However, two core challenges arise: 1) To achieve correct sequential manipulation, we require instant reactiveness to perturbations that concern constraints in the far future of the plan. For instance, when the target at the end of a pushing maneuver is perturbed, this might require an instant reaction in the approach angle in an early phase of the maneuver. Correctly accounting for such long-term interdependencies and coupling them to immediate reactiveness is beyond standard (e.g. fixed-horizon) MPC approaches and existing TAMP execution methods [2], [8], [10]. 2) TAMP plans include switches in kinematic and dynamic constraints [11], and the original plan might provide a temporal scheduling of such switches. However, in a reactive execution approach also the timing of these constraint switches needs to be reactive. This either requires employing optimal control methods invariant to such switches [12], [13] or to include explicit timing-estimation and -optimization as part of the MPC problem.

This paper is dedicated to address these two challenges. We assume that a TAMP plan was computed and is given in terms of a linear sequence of constraints (aka. “skeleton”). Plan switching or re-planning action sequences [2] is beyond the scope of this work. Instead we focus on deriving a Timing-Optimal Sequence-of-Constraints MPC (SecMPC) approach to robustly control through a sequence of constraints that are subject to perturbations during execution, which includes a backtracking mechanism when perturbations break the running constraints of a manipulation phase. Backtracking here means that we stay within the linear

The research has been supported by the German Research Foundation (DFG) under Germany’s Excellence Strategy EXC 2002/1390523135 “Science of Intelligence”

[1]Learning & Intelligent Systems Lab, TU Berlin
[2]Science of Intelligence Excellence Cluster, TU Berlin
toussaint@tu-berlin.de

[1]https://youtu.be/aFuA50H9uek

sequence of constraints, but back up to an earlier phase whenever constraints of the current phase are not fulfilled. This leads to an automatic reactive re-initiation of parts of the sequence.

Our MPC approach includes the timing of constraints as a decision variable, with the objective being a combination of total time and control costs, which we call *timing-optimal*.[2] However, unlike standard MPC approaches our formulation is not constrained to a short receding horizon, but instead approximates the full horizon problem to ensure timing and waypoint consistency, but with varying resolution. More specifically, each MPC cycle includes solving for 1) future waypoints to be consistent with future constraints and their dependencies in the sequence, 2) the timing of waypoints, and 3) a short receding horizon path to fulfill immediate collision and path constraints. The resulting overall system defines a *temporally consistent* controller, where, if there are no perturbations, the time-to-go and future path estimates of two consecutive MPC cycles are consistent modulo the shift in time, guaranteeing the conclusion of the full sequence in the originally estimated timing and along the originally estimated path.

We demonstrate the control framework on a pick-and-place and quadrotor scenario, but in particular on a pushing-with-stick scenario. The latter experiment is inspired by the observation in [11] that humans execute pushing tasks drastically different from optimal stable pushing plans: The overall behavior is dominated by frequent re-initiation of brief pushing phases, where each push is instable and rather imprecise, but the re-initiations robustly control the object to the target.[3] Our approach can be seen as a response to this observation, providing a model of how a sequence-of-constraints plan can lead to complex behavior reactively cycling through execution phases.

II. Related work

A. Bridging between TAMP and Reactive Execution

In [2], a reactive sequential manipulation framework is proposed that can reactively replan logic decisions, but requires a manual design of the controllers per logic action. [14] similarly combine linear temporal logic TAMP planning with behavior trees for reactive action selection and plan switching under interventions. Both works go beyond our work in terms of the reactiveness on the action level, but do not consider a coherent optimal control formulation, timing-optimality, or the interdependencies of future waypoints in the fluent execution of a fixed skeleton.

To enable reactive execution of TAMP plans, [10] proposes to interpret them in object-centric Cartesian coordinates and design corresponding perception-based operational space controllers for execution. [15] provides a novel method to automatically design a sequence of controllers using kino-dynamic planning. In contrast, our approach is a pure optimization-based MPC approach that can take the interdependencies of future waypoints into account.

Receding-Horizon TAMP [16], [17], [18] has been proposed to speed up and decompose long-horizon TAMP problems, but without bridging to low-level reactive control.

B. Timing Optimization & MPC

The timing of future constraints is a central concern in our approach, that has previously been studied in the context of locomotion [19]. In the context of quadrotor flight, [20] proposed an efficient time-optimal trajectory generation method, and [21] proposed an interesting alternative formulation based on progress variables which ensure that an optimal path transitions through a given sequence of constraints in a timing-optimal manner. [22], [23] proposed further novel methods for time-optimal path planning. However, these methods are used for trajectory optimization rather than as a reactive control framework.

In the context of MPC, [24] proposed a sampling-based MPC method for robotic manipulation, and [25], [26] further advanced the state in time-optimal MPC, but these works do not consider controlling through a sequence of switching constraints as they appear in TAMP plans.

Finally, hybrid control, and in particular trajectory optimization through switching constraints [12], [27] have in principle high potential for MPC through manipulation, but have to our knowledge not been extended to reactively account for interdependencies of future waypoint constraints and their timing.

III. Problem Formulation

A. Sequence-of-Constraints Optimal Control

We consider a system configuration space $X = \mathbb{R}^n$ which includes robot dofs as well as manipulation dofs such as relative grasp, push, and place poses. The latter typically underlies constancy constraints that imply long term dependencies [11], and we treat them in analogy to constant design parameters [28]. We use $\underline{x} = (x, \dot x)$ to denote the system state, with configuration $x \in X$ and velocity $\dot x \in \mathbb{R}^n$. A Sequence-of-Constraints optimal control problem is specified by a tuple

$$(\hat\phi_{1:K}, \bar\phi_{1:K}) \ ,$$

which is a sequence of K waypoint constraint functions $\hat\phi_k : X \to \mathbb{R}^{\hat d_k}$, as well as K running constraint functions $\bar\phi_k : X \times \mathbb{R}^n \to \mathbb{R}^{\bar d_k}$. Each constraint function ϕ may have a different output dimension (number of constraints) and is assumed to be smoothly differentiable. Fig. 1 illustrates the notation. The objective is to find a trajectory $\xi : [0, t_K] \to X$ and its timing $t_{1:K}, t_k \in \mathbb{R}_+$ to minimize

$$\min_{\xi, t_{1:K}} \ t_K + \alpha \int_0^{t_K} c(\xi(t), \dot\xi(t), \ddot\xi(t)) \ dt \tag{1a}$$

$$\text{s.t.} \quad \xi(0) = x, \ \dot\xi(0) = \dot x, \ \dot\xi(t_K) = 0 \ , \tag{1b}$$

$$\forall_k : 0 < t_k < t_{k+1} \ , \tag{1c}$$

[2]Note that the term *time-optimal* is typically used when the objective is *only* total time (typically with limits on controls), leading to bang-bang type solutions; we specifically use the term *timing-optimal* to emphasize that we optimize timing for a combination of total time and control costs.

[3]See the data at the end of https://youtu.be/-L4tCIGXKBE.

$$\forall_k : \hat{\phi}_k(\xi(t_k)) \le 0,\ \forall_{t\in[t_{k\text{-}1},t_k]} : \bar{\phi}_k(\underline{\xi}(t)) \le 0\ . \tag{1d}$$

In this notation, constraint functions ϕ define inequalities on system configurations x and velocities $\dot{x}$, but this is meant to include equality constraints.[4] For example, running constraints can impose dynamics and collision avoidance constraints, while waypoint constraints can impose constraints for transitioning between dynamics modes, e.g., constraints to initiate a stable grasp or push in our experiments. Relating to TAMP formalisms, waypoint constraints correspond to constraints for switching between modes, while running constraints encode the dynamics of a mode [11], [15], which are imposed by a skeleton (logical decision sequence). Finally, $c(\xi(t),\dot{\xi}(t),\ddot{\xi}(t))$ represents control costs scaled with $\alpha \in \mathbb{R}$: In our experiments we will simply penalize square accelerations $\|\ddot{q}\|^2$ in robot dofs $q \subseteq x$, but also add a small pose regularization $\|q - q_{\text{home}}\|^2$ that prefers robot poses close to the homing position and prevents the system from drifting through null-spaces in long manipulation sequences.

B. Interdependency of waypoint constraints

Sequential robotic manipulation implies long-term dependencies e.g. between the pick and place pose of an object via the constraint of a constant relative grasp pose, or the onset of a push and the final placement of the object via stable push constraints. To account for such dependencies we augment the system configuration space X to include manipulation dofs. For instance, such manipulation dofs represent a constant relative grasp pose or final push-placement. By imposing constancy constraints (either via running constraints $\bar{\phi}$ or by explicit sharing of dofs across time slices, see [28]), the sequence of waypoint constraints $(\hat{\phi}_1(\xi(t_1)),..,\hat{\phi}_K(\xi(t_K)))$ concerns parameters shared across waypoints to model their interdependencies.

IV. Decomposition

The full problem (1) would raise high computational costs within an MPC loop: The running constraints concern the continuous-time path and may require a fine resolution time discretization, e.g. to evaluate collisions. This would imply that we would have to jointly optimize over a fine resolution path through all waypoints and their timing, which is tractable offline but costly within an MPC loop.

We therefore propose to approximate and decompose the full path problem as follows. First note that a coordinate descent approach [29] to (1) would alternate between solving for the timing $t_{1:K}$ given the current path ξ, and vice versa. However, optimizing the full-horizon fine path ξ within each MPC cycle would still be computationally inefficient. We therefore decompose the problem further by representing the path coarsely in terms of the waypoints $x_{1:k} = \xi(t_1),..,\xi(t_k)$, and finely only in a short receding horizon of length H ($\approx$1sec). More specifically, this defines our three main decision variables,

1) the timing $t_{1:K}$,
2) the waypoints $x_{1:K} = \xi(t_1),..,\xi(t_K)$,
3) the short receding horizon path $\xi : [0,H] \to X$, in fine time discretization.

For each variable, we define a sub-problem assuming the other variables fixed, as detailed in the following sections. Each cycle iterates only once over the three sub-problems, but when in subsequent MPC cycles the optima become stable we have a joint and consistent solution. However, this joint solution still is only approximate to the full problem (1) as the waypoints only coarsely represent the full path ξ.

A. The waypoints problem

The waypoints problem assumes a fixed timing $t_{1:K}$ and solves for waypoints $x_{1:K}$ at these prescribed timings. This problem setting coincides with standard waypoint optimization, in particular the methods employed previously in the context of optimization-based TAMP: Namely, in [30], [11] we described the *sequence bound* as optimizing only over waypoints, but subject to manipulation constraints, e.g. that a relative grasp pose remains stable. As we discretized the path representation, the running constraints $\bar{\phi}$ in (1) now become constraints that couple two consecutive waypoints, $\bar{\phi}_k(x_{k\text{-}1},x_k)$, e.g. ensuring that the relative grasp in two consecutive waypoints remains equal. The waypoints problem therefore is of the form

$$\min_{x_{1:K}} \ \sum_{k=1}^{K} \tilde{c}(x_{k\text{-}1},x_k) \tag{2}$$

$$\text{s.t.}\ \ \forall_k : \hat{\phi}_k(x_k) \le 0,\ \bar{\phi}_k(x_{k\text{-}1},x_k) \le 0\ , \tag{3}$$

where $\tilde{c}$ subsumes control costs between waypoints, and which we tackle using a solver from previous work [11].

B. The timing problem under a cubic spline model

The timing problem assumes fixed waypoints $x_{1:K}$. To optimize the timing of waypoints we choose the control costs to be squared accelerations, neglecting the above mentioned pose regularization. Therefore, the timing sub-problem aims to find a path $\xi : [0,t_K] \to X$ through the given waypoints and its timing $t_{1:K}, t_k \in \mathbb{R}_+$ to minimize

$$\min_{t_{1:K},\xi}\ t_K + \alpha \int_0^{t_K} \ddot{\xi}(t)^2\ dt \tag{4}$$

$$\text{s.t.}\ \ \xi(0)=x,\ \dot{\xi}(0)=\dot{x},\ \dot{\xi}(t_K)=0, \tag{5}$$

$$\forall_k : \xi(t_k)=x_k,\ 0<t_k<t_{k+1} \tag{6}$$

In consistency with (1) we minimize for total time t_K, but simplify the control costs to smoothness $\ddot{\xi}^2$ and replace non-linear constraints with waypoint constraints that were assumed to solve for the non-linear constraints.[5]

[4] In practice, we label outputs of ϕ to be either equalities or inequalities, so that a solver can treat them accordingly, e.g. using an Augmented Lagrangian-term or a log barrier-term, respectively.

[5] Our framework would also allow to impose hard control limits on $\ddot{\xi}$ and have a pure time-optimal formulation, not mixing with control costs. We found this to be practical with a log-barrier methods as a solver; however robustly warm-starting log-barrier methods within an MPC cycle turned out a substantial challenge, which is why we decided to back away from hard control limits and a pure time-optimal formulation in this first work on SecMPC.

For square acceleration costs and given boundary conditions $\xi(t_k), \dot{\xi}(t_k)$ at each time step, the optimal path (minimizing $\int \ddot{\xi}$) between two consecutive steps is a cubic polynomial. The optimal solution to the above problem is therefore a piece-wise cubic spline, that is parameterized by $t_{1:K}$ and $\xi(t_k), \dot{\xi}(t_k)$, where $\xi(t_k) = x_k$ is fully known. Defining $v_k = \dot{\xi}(t_k)$ we therefore can rewrite the timing problem using the decision variables $t_{1:K}, v_{1:K\text{-}1}$, or alternatively, $\tau_{1:K}, v_{1:K\text{-}1}$ with the delta timings $\tau_{1:K}$ such that

$$t_k = \sum_{i=1}^{k} \tau_k \ . \tag{7}$$

We define the cubic piece cost as

$$\psi(x_0, v_0, x_1, v_1, \tau) \ = \min_z \int_0^\tau \ddot{z}(t)^2 \ dt$$
$$\text{s.t.} \ \begin{pmatrix} z(0) \\ \dot{z}(0) \end{pmatrix} = \begin{pmatrix} x_0 \\ v_0 \end{pmatrix}, \begin{pmatrix} z(\tau) \\ \dot{z}(\tau) \end{pmatrix} = \begin{pmatrix} x_1 \\ v_1 \end{pmatrix} \ . \tag{8}$$

For boundary conditions $(x_0, v_0, x_1, v_1, \tau)$, this is solved by a cubic spline $z(t) = at^3 + bt^2 + ct + d$ with

$$d = x_0 \ , \quad c = \dot{x}_0 \tag{9}$$
$$b = \frac{1}{\tau^2}\Big[3(x_1 - x_0) - \tau(\dot{x}_1 + 2\dot{x}_0)\Big] \tag{10}$$
$$a = \frac{1}{\tau^3}\Big[-2(x_1 - x_0) + \tau(\dot{x}_1 + \dot{x}_0)\Big] \ , \tag{11}$$

and the minimal cost ψ of (8) is

$$\psi = \int_0^\tau \ddot{z}(t)^2 \ dt = 4\tau b^2 + 12\tau^2 ab + 12\tau^3 a^2 \tag{12}$$
$$= \frac{12}{\tau^3} \ [(x_1 - x_0) - \frac{\tau}{2}(v_0 + v_1)]^2 + \frac{1}{\tau}(v_1 - v_0)^2 \tag{13}$$
$$= \frac{12}{\tau^3} D^\top D + \frac{1}{\tau} V^\top V = \tilde{D}^\top \tilde{D} + \tilde{V}^\top \tilde{V} \tag{14}$$
$$D := (x_1 - x_0) - \frac{\tau}{2}(v_0 + v_1), \ V := v_1 - v_0, \tag{15}$$
$$\tilde{D} := \sqrt{12} \ \tau^{-\frac{3}{2}} \ D, \ \tilde{V} := \tau^{-\frac{1}{2}} \ V \ . \tag{16}$$

Here, (14) expresses the cubic piece cost as a least-squares of differentiable features of $(x_0, v_0, x_1, v_1, \tau)$, where D can be interpreted as distance to be covered by accelerations, and V as necessary total acceleration. The Jacobians of $\tilde{D}$ and $\tilde{V}$ w.r.t. all boundary conditions are trivial. Exploiting the least-squares formulation of ψ we can use the Gauss-Newton approximate Hessian.

Therefore, the timing sub-problem (4) is equivalent to optimizing over the boundary conditions of cubic pieces,

$$\min_{\tau_{1:K}, v_{1:K\text{-}1}} \sum_{k=1}^{K} \tau_K + \alpha \sum_{k=1}^{K} \psi(x_{k\text{-}1}, v_{k\text{-}1}, x_k, v_k, \tau_k) \ , \tag{17}$$

where $x_0 = x$, $v_0 = \dot{x}$ and $v_K = 0$. Note that in (4) and also the original problem (1) we required $\dot{\xi}(t_K) = 0$ at the end of the manipulation, which is why we do not have a decision variable v_K for the last waypoint.

Due to the least-squares nature of all objectives, the timing sub-problem (17) is efficient to solve for and yields a timing as well as waypoint velocities v_k, which together define a piece-wise cubic spline ξ^* through the optimized waypoints. While the waypoints problem and the short horizon problem below can be tackled by standard trajectory optimization methods used in previous work, we transcribed the timing problem into a non-linear least squares formulation and employed a sparse Gauss-Newton method to solve it.

C. The short receding horizon problem

Finally, as is standard in MPC approaches, we solve for a short receding horizon path that is more finely discretized, e.g. to account for collisions. Let ξ^* be the cubic spline that results from the timing solution based on the previous waypoint solutions. This ξ^* incorporates the full-horizon information from the constraints. We formulate the short horizon MPC problem to track this reference modulo fulfilling running constraints. Therefore, if no running constraints (e.g. collisions) are active, the short horizon MPC will reproduce the reference ξ^*.

Specifically, we solve for a path $\xi^H : [0, H] \to X$ to minimize

$$\min_\xi \ \int_0^H \alpha \ \ddot{\xi}(t)^2 + \|\xi(t) - \xi^*(t)\|^2 \ dt \tag{18}$$
$$\text{s.t.} \ \ \xi(0) = x, \ \dot{\xi}(0) = \dot{x} \ , \tag{19}$$
$$\forall_{t \in [0,H]} : \bar{\phi}_{k(t)}(\underline{\xi}(t)) \le 0 \ , \tag{20}$$

which is the short horizon version of the original problem (1), but replacing waypoint constraints and timing by the reference tracking cost $\|\xi^H(t) - \xi^*(t)\|^2$.

We use standard trajectory optimization to solve the short horizon problem, time-discretizing the horizon $[0, H]$. Note that this time interval may "cross" a waypoint and corresponding waypoint constraint. However, as the timing $t_{1:K}$ is given and the reference $\xi^*(t)$ is adapted to cross the waypoint with an optimal timing, the short horizon solution will aim to adopt this timing and we have a clear time embedding of the time-discretization.

V. Sequence-of-Constraints MPC

Algorithm 1 describes the concrete MPC cycle we propose and evaluate in our experiments. The core of each cycle is to iteratively solve the three sub-problems described in the previous section, which are indicated with bold comments in the pseudo code. However, as the pseudo code indicates, there are additional details that concern phase management and stability that we found essential to have a practical MPC system, and are discussed in the following. We use the term *phase* to denote the integer $\kappa \in \{1, .., K\}$ that indicates which future constraints $\phi_{\kappa:K}$ are remaining.

A. Temporal Consistency of Timing MPC

We first establish a basic property of our MPC approach, namely temporal consistency under the assumption of no perturbations, i.e., for a deterministic control system and no external object interventions. Under this assumption, waypoint optimization will consistently converge to the same waypoints, and for this analysis we assume the waypoints $x_{1:K}$ are fixed. Further, without perturbations the deterministic control of ξ^H will coincide with the timing-optimized cubic spline ξ^*, and we have:

Algorithm 1 SECMPC Cycle

Maintained from last cycle: last time $\mathcal{T}'$, phase κ, waypoints $x_{\kappa:K}$, delta timings $\tau_{\kappa:K}$, velocities $v_{\kappa:K}$, short path ξ^H

Input: state $\underline{x} = (x, \dot{x})$ (including external objects) measured at real time $\mathcal{T}$

1: $\delta \leftarrow \mathcal{T} - \mathcal{T}'$ *// real time since last cycle*
2: $\tau_\kappa \leftarrow \tau_\kappa - \delta$ *// shift timing*
3: **if** $\tau_\kappa < 0$ **then** *// expected transition*
4: **if** $\|\hat{\phi}_\kappa(\underline{x})\| \leq \hat{\theta}$ **then**
5: **If** $\kappa < K$ **then** $\kappa \leftarrow \kappa + 1$ *// phase progression*
6: **else**
7: $\tau_\kappa \leftarrow \tau_{\text{init}}$
8: **end if**
9: **end if**
10: **while** $\|\bar{\phi}_\kappa(\underline{x})\| > \bar{\theta}$ **and** $\kappa > 1$ **do** *// phase backtracking*
11: $\tau_{\kappa\text{-}1}, \tau_\kappa \leftarrow \tau_{\text{init}}$, $\kappa \leftarrow \kappa - 1$
12: **end while**
13: $\underline{x} \leftarrow \text{Filter}(\underline{x}, \xi^*(\mathcal{T}), \dot{\xi}^*(\mathcal{T}))$
14: $x_{\kappa:K} \leftarrow \text{solve}(2)$ *// **waypoints**, given $\underline{x}, \tau_{\kappa:K}$*
15: **if** $\tau_\kappa > \epsilon$ **then**
16: $\tau_{\kappa:K}, v_{\kappa:K} \leftarrow \text{solve}(17)$ *// **timing**, given $x_{\kappa:K}$*
17: **end if**
18: $\xi^* \leftarrow \text{CubicSpline}(\tau_{\kappa:K}, x_{\kappa:K}, v_{\kappa:K})$
19: $\xi^H \leftarrow \text{solve}(18)$ *// **short path**, given $\underline{x}, \xi^*$*
20: $\mathcal{T}' \leftarrow \mathcal{T}$, $\mathcal{T} \leftarrow \text{realTime}()$
21: Return ξ^H for execution

Property 1 (Temporal (Bellman) Consistency)**.** *Consider the solutions $(\tau^{(1)}_{1:K}, v^{(1)}_{1:K\text{-}1})$ and $(\tau^{(2)}_{1:K}, v^{(2)}_{1:K\text{-}1})$ to the MPC problem in two consecutive MPC iterations with real time gap δ. If the system is deterministic and the expected time to the first waypoint is $\tau^{(1)}_1 > \delta$, it holds*

$$\tau^{(2)}_1 = \tau^{(1)}_1 - \delta \ , \tag{21}$$

$$\tau^{(2)}_{2:K} = \tau^{(1)}_{2:K} \ , \tag{22}$$

$$v^{(2)}_{1:K\text{-}1} = v^{(1)}_{1:K\text{-}1} \ . \tag{23}$$

That is, the solutions are identical except for the timing $\tau^{(2)}_1$ of the first waypoint reflecting the real time δ that has passed.

Proof. The property is a direct consequence of Bellman's optimality principle: Both $(\tau^{(1)}_{1:K}, v^{(1)}_{1:K\text{-}1})$ and $(\tau^{(2)}_{1:K}, v^{(2)}_{1:K\text{-}1})$ encode the cubic splines $\xi^{(1)}$ and $\xi^{(2)}$. Since after the first cycle we execute $\xi^{(1)}$ deterministically, Bellman's principle applied to the path optimality (4) requires the remaining optimal path $\xi^{(2)}$ to coincide with the previously optimal path $\xi^{(1)}$. □

Further we have the property that, if $\tau^{(1)}_1 \leq \delta$, the deterministic system will transition through the first waypoint x_1 exactly at time $\tau^{(1)}_1$. When transitioning through a waypoint, the next MPC solution will be temporally consistent to the previous, in the sense that $\tau_{2:K}$ becomes the new decision variable and $\tau^{(2)}_2 = \tau^{(1)}_2 - (\delta - \tau^{(1)}_1)$ will be reduced by the real time after waypoint transitioning.

B. *Waypoint Transitioning under Stochasticity*

In contrast to the above, exact waypoint transitioning is in principle impossible (of measure zero) when we have the slightest system stochasticity. Fig. 5(a) displays timing-optimal paths when the waypoint is offset for varying start conditions. We see a clear discontinuity in behavior between directly steering to the waypoint and deciding to pass the waypoint and looping back. In the stochastic case, the system will eventually always experience a sufficient offset and MPC will be stuck in indefinitely looping to re-target the same waypoint, trying to thread the infinitesimal needle.

A rigorous treatment of the stochastic case, e.g. with a tube MPC and quantile dynamics formulation [31], would require to make explicit quantile dynamics assumptions on the system stochasticity and external perturbations. Such a treatment is beyond the scope of this paper and we want to avoid formulating a priori probabilistic assumptions about external perturbations.

We therefore propose a simple cutoff and backtracking approach: When $\tau_1 \leq \epsilon$ we do not re-optimize the timing and let the system instead continue to track the last spline reference ξ^* until the expected time of waypoint passage. Then, in the first MPC cycle after the expected waypoint transitioning, we check whether the waypoint was passed with sufficient accuracy to progress phase, or otherwise keep the phase and old waypoint active, which automatically leads to a looping spline and substantial increase of τ_1 for re-targeting the old waypoint. In the pseudo code, the cutoff is realized in line 15, and the waypoint check in line 5, or alternatively also in 10 which would backtrack if the subsequent running constraint is missed and encodes the desired criteria for waypoint collection.[6]

C. *Phase Backtracking & Overall* SECMPC *Cycle*

We add phase backtracking to our SECMPC cycle (line 10), which re-initiates the previous phase and waypoint if the current running constraints are missed. Reassigning $\theta_\kappa \leftarrow \tau_{\text{init}}$ means to reinitialize the delta timings away from zero, so that subsequent timing optimization can better converge. Further, line 13 allows to include a filter on the actuated dofs to stabilize the system. In particular, if the inverse dynamics used to execute ξ^H has a systematic error (as is the case for our used robotic system, a Franka Panda), an MPC that fully adopts the true current state as the start state for a new reference can be too compliant to the tracking error. Instead of fully adopting the current state as reference start state, we choose a reference start state in the interpolation between the old reference point $\xi^*(\mathcal{T})$ and the true current state x, with at max distance r to the current state, where r equals the typical joint space tracking error of the system.

[6]If we had quantile assumptions on the system stochasticity and a waypoint margin θ, we could choose ϵ based on this, namely choose ϵ such that $P(|x(t+\epsilon) - x^*(t+\epsilon)| > \delta) < \alpha$, i.e. the probability of missing the reference with margin at time $t + \epsilon$ is lower than a given α. By choosing α we choose how often (theoretically) the system will fail and have to loop back. However, rather than making explicit assumptions about system (and external) stochasticity, we heuristically fix ϵ to a time horizon, typically of 0.1 seconds.

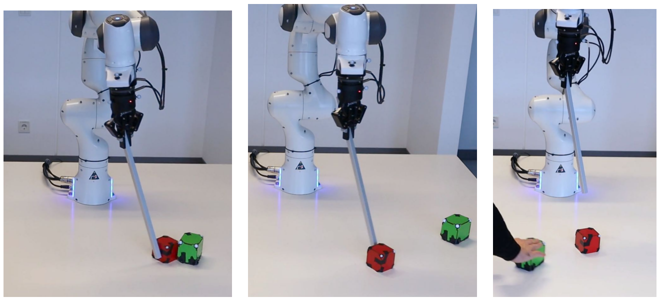

Fig. 2: Pushing scenario, where the task is to push a red box into contact with a green (left). Fluent pushes frequently let the red box rotate out of the push (middle), and the experimenter interferes by replacing the target box during execution (right), both of which leads to robust re-initiation of the manipulation sequence.

VI. Experiments

A. *Reactive Pushing under Interventions*

We start with discussing a real-world pushing scenario that best motivates our approach and highlights the properties of the SecMPC system. Below we discuss two more real-world scenarios, pick-and-place and quadrotor through gates, as well as studies on simplistic settings to analyze SecMPC properties. Please see the accompanying video as well as raw video footage and the fully open code for the following experiments.[7]

We use Optitrack to sense the pose of objects and execute SecMPC on a standard i8 core RT Linux machine without parallelization or use of a GPU with a cycle time of 30msec. We choose a short horizon of $H = 1$sec and a time resolution of 100msec within the short horizon sub-problem, and a cutoff of $\epsilon = 100$msec. Each sub-problem is addressed using a 2nd-order Gauss-Newton method. The actually required number of Newton steps can be high (above 50, in each of the three sub-problems) at initialization or with significant perturbations, but very low (below 5) when perturbations are low and the warmstarts are close to the solution. In the latter case, the actual compute time for a SecMPC cycle is less than 10msec; in the first case, the actual compute time can exceed the 30msec and thereby slow down the MPC cycle. Due to its general non-linearity, waypoint optimization is in principle prone to converging to an infeasible point or aborting (after 300 Newton steps). However, in our experimental setups we did not experience local optima and this never occurred.

Fig. 2 shows a pushing scenario where a Panda robot holds a stick 50cm long with which it aims to push the red block to establish contact with the green block. The potential perturbations in this scenarios are abundant:

1) The red and green blocks can be replaced at any point during the execution by an interfering experimenter.
2) The experimenter can manually hold back the robot arm, perturb it from its path, or manually bring a collision obstacle (additional stick) into the scene to further interfere with the execution.
3) The real-world behavior of the red block under pushing contact is rather unpredictable.[8]

Therefore, rather than hoping that we could have a (probabilistically) accurate model of the dynamics, we aim for a system that is robust to such interferences and failures of intermediate phases, in particular by re-initiating the manipulation sequence as needed.

We model the pushing behavior as a sequence of four constraints, $\hat\phi_{1:4}$, where the first three are simple geometric constraints on approach waypoints, and the pushing itself occurs from the 3rd waypoint. As the final target constraint is highly coupled to all previous constraints, we start explaining the constraints backward:

- $\hat\phi_4$ constrains the distance between red and green box to zero; this implies the waypoint solver to maintain an estimate of the final place pose of the red box.
- $\hat\phi_3$ constrains the stick tip to be in contact with the red box at a position that is *opposite* to the final place pose relative to the red box center – this describes a central push on red toward the final pose.
- $\hat\phi_2$ is almost identical to $\hat\phi_3$, but 3cm away from contact (modeling a pre-push pose waypoint); and $\hat\phi_1$ is similar to $\hat\phi_2$, but 10cm higher.

The pre-push pose constraints $\hat\phi_{1,2}$ imply approach waypoints that can efficiently and fluently be transitioned by SecMPC. Such approach constraints could have been designed in other ways, e.g. imposing constraints on approach velocities or directions, but we found this easiest and exploited SecMPC's fluency in transitioning waypoints.

Throughout all phases we have the same running constraints $\bar\phi_{1:4}$, which constrain the stick tip to be on the opposite side to the final place pose relative to the red box center, as well as constrain the stick tip, red box center, and final placement to be aligned on a single line. Whenever these running constraints are missed, SecMPC will initiate phase backtracking.

SecMPC achieves a highly robust behavior capable to cope with the mentioned interferences and perturbations. In particular, without interference by the experimenter, the system shows fluent push motions that are aborted and re-initiated whenever the red object rotates out of the push, robustly leading to fulfilling the task. When the experimenter interferes by replacing the green box, this changes the final target constraint $\hat\phi_4$ and thereby all waypoint and running constraints, typically also leading to a phase backtracking and re-initiation of the maneuver to ensure alignment to an updated red box target pose.

B. *Reactive Pick-and-Place*

In a second scenario we consider pick-and-place under interferences, see Fig. 3. Here, the constraints and geom-

[7] https://www.user.tu-berlin.de/mtoussai/22-SecMPC/

[8] The grip of the stick in the robot hand is not perfectly stable and imprecise; the red block pose estimation underlies stochasticity from Optitrack; the precise push contact is unknown; the precise ground-box interaction via an inhomogeneous and not perfectly even box lower side is uncertain.

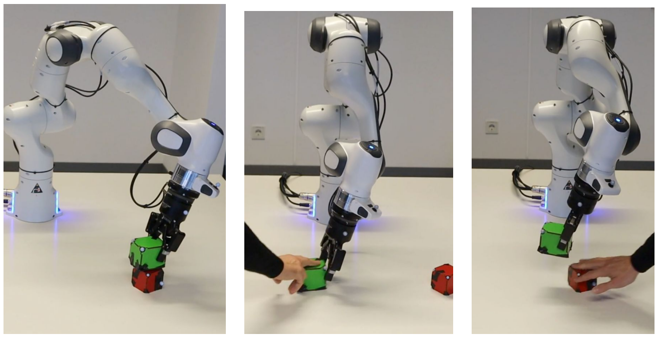

Fig. 3: Pick-and-place scenario, where the task is to stack a red box onto a green (left). The experimenter interferes by stealing, replacing and rotating the box during pre-grasp (middle), and replacing the target during pre-place (right).

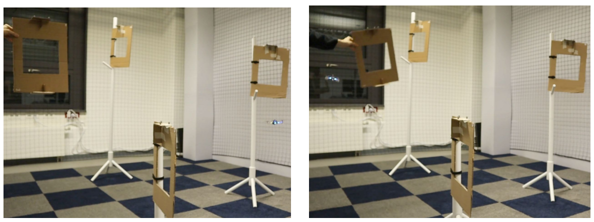

Fig. 4: Quadrotor scenario, where the task is to (indefinitely) transition through the gates (left). The experimenter interferes by displacing the left gate during the approach (right).

etry were modeled exactly as in previous work on TAMP [11], showing that TAMP planning models can directly be transferred to a reactive execution system using SECMPC. Again, the system exhibits robustness to interferences by the experimenter when stealing and replacing the box during pre-grasp, replacing the target during placement, or directly pushing the arm or intervening with a manual obstacle stick. For brevity, we refer to the accompanying video to showcase this scenario.

C. Quadrotor through Gates

To highlight that our system is not specific to robotic manipulation, we demonstrate SECMPC also on a quadrotor scenario, see Fig. 4. Again, for brevity we refer to the accompanying video to showcase this scenario.

D. Analysis on Analytic Settings

We add results on simplistic low-dimensional settings to gain more insights in the timing-optimal behavior of SECMPC. Fig. 5(a) displays timing-optimal trajectories (solving (4)) in 2D configuration space for two aligned waypoints. When the system is close to the first waypoint with straight velocity but lateral offset, we see the discontinuous switch in deciding to pass the waypoint and loop back (Sec. V-B).

Figs. 5(b,c) compare the convergence to a single waypoint in a 1D configuration space to a critically damped regulator. A naive linear regulator would have undesirable high

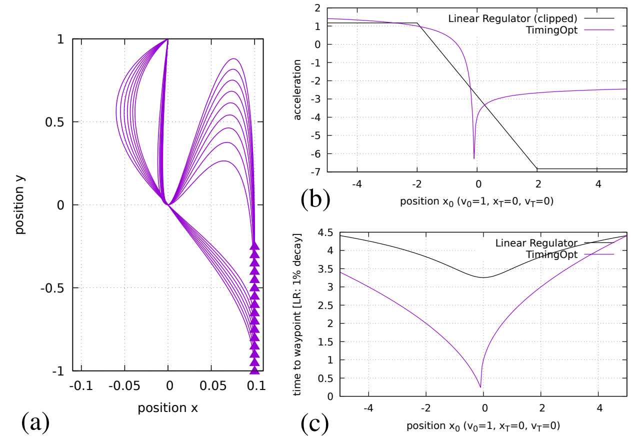

Fig. 5: (a) Timing-optimized trajectories in 2D through two waypoints $(0,0)$ and $(0,1)$, starting with upward velocity $(0,1)$, slight x-offset, and different y-offset (triangles are starting points). (b) Acceleration of TimingOpt vs. a critically damped clipped Linear Regulator, when starting in 1D at $(x, \dot{x}=1)$ to waypoint $(x=0, \dot{x}=0)$. (c) Exact time-to-go for TimingOpt vs. time-to-1%decay for the Linear Regulator

feedback for large initial error; a clipped regulator (where the error or feedback is upper bounded, as typically used in practice) would have acceptable behavior far from the waypoint, but still "only" exponential convergence near the waypoint. A timing-optimal MPC approach to the waypoint generates moderate feedback far and near the waypoint while at the same time providing an exactly timed and definite convergence to the waypoint if no perturbations.

We also compared the average total time to transition through 5 perturbed waypoints when using SECMPC through the full sequence vs. sequencing five independent 1-phase approaches. Each 1-phase approach is realized with SECMPC that only knows about the next waypoint (which is already an improvement over using a linear regulator to individually approach each waypoint, as shown above). As expected, when waypoints are somewhat aligned, the full SECMPC can fluently transition them and requires significantly less total time for transition. For fully random waypoints $\sim [-1,1]^3$, we still found a reduction in total time from 14.1 ± 0.6sec to 11.2 ± 0.6sec, averaged over 20 random trials.

VII. CONCLUSION

In this paper we derived a reactive control strategy to execute a given linear sequence of constraints. The scope of this paper is more narrow than previous work on full online replanning in a TAMP setting [2], [14], [8]. However, the focus on a given sequence of constraints allows us to derive a control strategy that encompasses the interdependencies between future waypoints and their timing and includes timing optimization and phase backtracking as an integral part of reactiveness. A core limitation of the approach arises from the approximate decomposition of the full problem, in particular from representing the long term path only coarsely with waypoints. Running constraints between future waypoints, such as collision avoidance, are therefore not evaluated for waypoint estimation, but only accounted for in the short receeding horizon path.

On a more conceptual level, it is a standard paradigm to think of long term manipulation behavior as composed of individual primitive controllers, or motor skills, that are sequenced e.g. by RLDS [2], other state machines [4], or tree structures [3], and where each primitive has its own control law, e.g. acting as a funnel, or option (in the context of hierarchical RL). The presented system provides an alternative view, where not individual control laws are the basic building blocks, but constraints. Compositionality (and decision abstraction) is achieved on the level of constraints, and instead of composing primitive control laws, here we derive control directly from the composition of constraints. The constraint-based view on control is not novel [32], but our system provides an explicit timing-optimal MPC realization for a sequential composition of constraints.

REFERENCES

[1] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[2] C. Paxton, N. Ratliff, C. Eppner, and D. Fox, "Representing robot task plans as robust logical-dynamical systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5588–5595.

[3] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[4] M. Egerstedt, "Behavior based robotics using hybrid automata," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2000, pp. 103–116.

[5] C. Eppner, S. Höfer, R. Jonschkowski, R. Martín-Martín, A. Sieverling, V. Wall, and O. Brock, "Lessons from the amazon picking challenge: Four aspects of building robotic systems." in *Robotics: Science and Systems*, 2016.

[6] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[7] T. Stouraitis, L. Yan, J. Moura, M. Gienger, and S. Vijayakumar, "Multi-mode Trajectory Optimization for Impact-aware Manipulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9425–9432.

[8] T. Stouraitis, I. Chatzinikolaidis, M. Gienger, and S. Vijayakumar, "Online hybrid motion planning for dyadic collaborative manipulation via bilevel optimization," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1452–1471, 2020.

[9] J.-S. Ha, D. Driess, and M. Toussaint, "A probabilistic framework for constrained manipulations and task and motion planning under uncertainty," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6745–6751.

[10] T. Migimatsu and J. Bohg, "Object-centric task and motion planning in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 844–851, 2020.

[11] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," 2018.

[12] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[13] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–8, 2012.

[14] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy, "Reactive task and motion planning under temporal logic specifications," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 12 618–12 624.

[15] P. S. Schmitt, F. Wirnshofer, K. M. Wurm, G. v Wichert, and W. Burgard, "Modeling and planning manipulation in dynamic environments," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 176–182.

[16] N. Castaman, E. Pagello, E. Menegatti, and A. Pretto, "Receding Horizon Task and Motion Planning in Changing Environments," *Robotics and Autonomous Systems*, vol. 145, p. 103863, 2021.

[17] C. V. Braun, J. Ortiz-Haro, M. Toussaint, and O. S. Oguz, "RHH-LGP: Receding Horizon And Heuristics-Based Logic-Geometric Programming For Task And Motion Planning," *arXiv preprint arXiv:2110.03420*, 2021.

[18] V. N. Hartmann, O. S. Oguz, D. Driess, M. Toussaint, and A. Menges, "Robust task and motion planning for long-horizon architectural construction planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6886–6893.

[19] S. Caron and Q.-C. Pham, "When to make a step? tackling the timing problem in multi-contact locomotion by topp-mpc," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 522–528.

[20] J. Lin, N. Somani, B. Hu, M. Rickert, and A. Knoll, "An efficient and time-optimal trajectory generation approach for waypoints under kinematic constraints and error bounds," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5869–5876.

[21] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[22] H. Pham and Q.-C. Pham, "On the structure of the time-optimal path parameterization problem with third-order constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 679–686.

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

[24] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. 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*. PMLR, 2022, pp. 750–759.

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

[26] 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.

[27] J. Moura, T. Stouraitis, and S. Vijayakumar, "Non-prehensile Planar Manipulation via Trajectory Optimization with Complementarity Constraints," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 970–976.

[28] M. Toussaint, J.-S. Ha, and O. S. Oguz, "Co-Optimizing Robot, Environment, and Tool Design via Joint Manipulation Planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6600–6606.

[29] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[30] M. Toussaint and M. Lopes, "Multi-bound tree search for logic-geometric programming in cooperative manipulation domains," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4044–4051.

[31] M. Cannon, Q. Cheng, B. Kouvaritakis, and S. V. Raković, "Stochastic tube MPC with state estimation," *Automatica*, vol. 48, no. 3, pp. 536–541, 2012.

[32] E. Aertbeliën and J. De Schutter, "eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1540–1546.