A Computational Framework for Environment-Aware Robotic Manipulation Planning

Marco Gabiccini, Alessio Artoni, Gabriele Pannocchia and Joris Gillis

Abstract In this paper, we present a computational framework for direct trajectory optimization of general manipulation systems with unspecified contact sequences, exploiting *environmental constraints* as a key tool to accomplish a task. Two approaches are presented to describe the dynamics of systems with contacts, which are based on a penalty formulation and on a velocity-based time-stepping scheme, respectively. In both cases, object and environment contact forces are included among the free optimization variables, and they are rendered consistent via suitably devised sets of complementarity conditions. To maximize computational efficiency, we exploit sparsity patterns in the linear algebra expressions generated during the solution of the optimization problem and leverage Algorithmic Differentiation to calculate derivatives. The benefits of the proposed methods are evaluated in three simulated planar manipulation tasks, where essential interactions with environmental constraints are automatically synthesized and opportunistically exploited.

1 INTRODUCTION

Careful observation of how humans use their hands in grasping and manipulation tasks clearly suggests that their limbs extensively engage functional interactions with parts of the environment. The physical constraints imposed by the manipulandum and the environment are not regarded as obstacles, but rather as opportunities to guide functional hand pre-shaping, adaptive grasping, and affordance-guided manipulation of objects. The exploitation of these opportunities, which can be referred to as *environmental constraints* (EC), enables robust grasping and manipulation in dynamic and highly variable environments. When one considers the exploitation of EC, i.e. when manipulation actions are performed *with the help* of the environment, the boundary between grasping and manipulation is blurred, and traditional categories such as grasp and manipulation analysis, trajectory planning and interaction control appear somewhat artificial, as the problem we aim to solve seems to inextricably entangle all of them.

In this paper, we set out to formulate environment-aware manipulation planning as a nonlinear optimal control problem and discretize it according to a *direct transcription* scheme [3]. In Sec. 3, two approaches to describe the dynamics of systems with contacts are proposed and evaluated:

M. Gabiccini, A. Artoni, G. Pannocchia
Department of Civil and Industrial Engineering, Univ. of Pisa, Italy.
e-mail: name.lastname@unipi.it

J. Gillis
Department of Electrical Engineering, KU Leuven, Belgium.
e-mail: joris.gillis@kuleuven.be

in the first one, continuous contact reaction forces are generated by nonlinear virtual springs, and the requirement to avoid sliding contacts is handled in an apparently original way; in the second one, contact collisions are approximated as impulsive events causing discontinuous jumps in the velocities according to a modified version of the Stewart-Trinkle time-stepping scheme [47]. The introduction of two different models is motivated by the relative ease for the first (second) one to enforce EC exploitation primitives that avoid (profitably exploit) sliding motions during interaction.

Both formulations lead to a nonlinear programming (NLP) problem (Sec. 4) that we solve by using the Interior-Point (IP) method implemented in IPOPT [4] and discussed in Sec. 5. To improve computational efficiency, we also annotate sparsity in the linear algebra expressions and leverage algorithmic differentiation (AD) [22] to calculate derivatives both quickly and accurately: the adoption of the CasADi framework [2], described in Sec. 5, provides a profitable interface to all the above tools with a consistent API.

In Sec. 6, we evaluate our approaches in three simulated planar manipulation tasks: (i) moving a circular object in the environment with two independent fingers, (ii) rotating a capsule with an underactuated two-fingered gripper, and (iii) rotating a circular object in hand with three independent fingers. Tasks (i) and (ii) show that our algorithm quickly converges to locally optimal solutions that opportunistically exploit EC. Task (iii) demonstrates that even dexterous fingertip gaits can be obtained as a special solution in the very same framework. Conclusions are drawn in Sec. 7.

It is worth noting that with our method, approach planning, grasping, manipulation, and environment constraint exploitation phases occur automatically and opportunistically for a wide range of tasks and object geometries, with no *a-priori* specification of the ordering of the different stages required.

2 RELATED WORK

2.1 Exploitation of Environmental Constraints

The concept of exploiting environmental constraints is well-rooted in robotics. Pioneering work was performed already in the eighties in the context of motion planning [33] and manipulation [34]. However, these concepts did not have the proper influence on many of the recent developments on either area, perhaps due to the inadequacy of the mechanical impedance properties of contemporary industrial manipulators to achieve sliding motion primitives stably, thus precluding the adoption of strategies that exploit environmental constraints, e.g. by sliding one object over another [10]. Also the idea of programming using environmental constraints is well entrenched in robotics literature, starting with the seminal work [1], proceeding with [44], and culminating in the *iTaSC* framework [46].

The exploitation of complex interactions with the environment in a manipulation task also plays a central role in automation and manufacturing to design fixtures [7] and part feeders [6]. However, these works are highly specialized and they are limited to the case of handling a single object geometry.

2.2 Traditional Grasp Planners

State-of-the-art general grasping algorithms and dexterous mechanical hands lie at the opposite end of the spectrum: they are designed to perform grasping and manipulation of a wide range of object geometries and for many different tasks. Traditional grasp planners (such as OpenRAVE [11] and GraspIt! [35]) rely on precise finger-to-object contact points while avoiding the surrounding environment. In real-world scenarios these models, as well as the motion of the hand, will be highly uncertain leading to poor grasping performance for grasps that were deemed highly robust based on theoretical considerations.

The recent paper [5] has proposed a pipeline for automated grasp synthesis of common objects with a compliant hand by developing a full-fledged multi-body simulation of the whole grasping process in the presence of EC: however, to date, the approach seems time consuming and the sequence of primitive actions needed to perform complex tasks must be scripted in advance. Also recently, both grasp planning algorithms and grasping mechanisms have begun to take advantage of EC [13], albeit not systematically. While sequences of EC exploitation primitives have been shown to be robust and capable [9], [27], there has been no comprehensive research on how to enumerate and describe these primitives or how to sequence them into task-directed manipulation plans. A noteworthy exception where a sequence of task-directed manipulation plan exploiting EC is created and performed on-line with minimalistic sensory information and limited previous assumptions about the scene was recently presented in [15].

2.3 General Purpose Planning Algorithms

State-of-the-art sampling-based planning algorithms like RRT*[26] seem not tailored for situations where *a-priori* unknown contact interactions may cause a combinatorial explosion of system configurations. Recent extensions, initially presented in [19] for RRT, and successively devised for RRT* in [41], were able to cope with systems described by complex and underactuated dynamics. In [29], the authors presented an approach to moving an object with several manipulators. This problem presents some similarities to ours, since a sequence of phases has to be both planned and solved. The strong assumption that the plan called by the high-level scheduler will succeed — which is not always possible — has been removed in the recent contribution [8].

However, situations where intermitted contact sequences are not easily enumerated from the outset, but are key for the success of environment-aware manipulation plans, still appear to be out of their reach.

2.4 Machine Learning Approaches

Although significant progresses have been made in this area in recent years [28], learning robot motion skills still remains a major challenge, especially for systems with multiple intermittent contacts. Policy search is often the preferred method as it scales gracefully with system dimensionality, even if its successful applications typically rely on a compact representation that reduces the numbers of parameters to learn [30], [49], [42]. Innovative policy classes [24] have led to substantial improvements on real-world systems. However, designing the right low-dimensional representation often poses significant challenges. Learning a state representation that is consistent with physics and embeds prior knowledge about interactions with the physical world has recently been proposed in [25] and seems a promising venue to find effective methods to help improve generalization in reinforcement learning: however, the simulated robotic tasks solved by this methods are still far in complexity from environment-aware manipulation scenarios.

The recent contribution [31] developed a policy search algorithm which combines a sample-efficient method for learning linear-Gaussian controllers with the framework of guided policy search, which allows the use of multiple linear-Gaussian controllers to train a single nonlinear policy with any parametrization, including complex and high-dimensional policies represented by large neural networks. In [32], this method has been recently applied, with some modifications that make it practical for deployment on a robotic platform, to solve contact-rich manipulation tasks with promising results.

2.5 Optimization-based Trajectory Planning

Various research groups are currently pursuing direct trajectory optimization to synthesize complex dynamic behaviors for systems with intermittent contacts. Those working in locomotion [45] mainly adopt a multi-stage hybrid-mode approach (usually employing multiple-shooting), where the optimization is constrained to operate within an *a priori* specification of the mode ordering. Interestingly, a recent contribution [52] explored the synthesis of optimal gaits for legged robots without the need to specify contact sequences. Certainly, the adoption of such an approach seems the only viable solution for a multi-fingered hand manipulating an object also by exploiting EC. Along this line, it is worth mentioning the contact-invariant approach originally proposed in [37] to discover complex behaviors for humanoid figures and extended to the context of manipulation in [36]. The previously described trajectory optimization method has been recently employed to gradually train a neural network by following an Alternating Direction Method of Multipliers (ADMM) strategy with interesting results [38].

The approach presented in [43] inspired our work and is the one which is definitely closest. However, remarkable differences in the formulation of the dynamics for systems with contacts, in the choice of the solution algorithm, solver and framework, and in the focus of the paper — here, EC exploitation is sought as a key factor — render our work significantly different.

3 DYNAMICS OF SYSTEMS WITH CONTACTS

3.1 Penalty-based contact model

To our eyes, manipulation planning has to rely on a dynamic model of the system, namely of manipulandum, manipulator, and the environment, and of their mutual interactions through contact. We consider here deterministic systems with continuous state and control spaces, denoted by x and u respectively. The dynamic evolution of our controlled (non-autonomous) system can be described in continuous time t by a set of Ordinary Differential Equations (ODEs):

$$\dot{x}(t) = F(x(t), u(t)) \quad \text{or} \quad \tilde{F}(\dot{x}(t), x(t), u(t)) = 0 \tag{1}$$

(explicit dependence on t is omitted hereafter). Together with an initial value $x(0) = x_0$, equation (1) defines an initial value problem. In general, additional algebraic dependencies may exist among $\dot{x}$, x and u leading to a dynamic system governed by differential-algebraic equations (DAEs). In the presence of contact, for instance, and if contact forces are included among the controls u, contact interactions establish functional dependencies between u and x through a set of algebraic equations/inequalities. As an example, if f_N is the normal contact force and $g_N = g_N(x)$ the normal gap (shortest distance) between a finger and the object being manipulated, the complementarity and non-negativity conditions $0 \leq f_N \perp g_N(x) \geq 0$ must hold. The contact model described in this section is based on a special treatment of the contact forces and the relative velocities that arise during interaction between manipulandum, manipulator, and environment. Such a model has proved to be successful in solving trajectory planning problems for manipulation tasks with *no sliding*, in the presence of EC.

For normal contact forces, the underlying idea is borrowed from classical penalty-based approaches, where contact interactions are modeled by spring-dampers. In our model, no damping is introduced, while a nonlinear exponential spring relates the normal contact force and the nor-

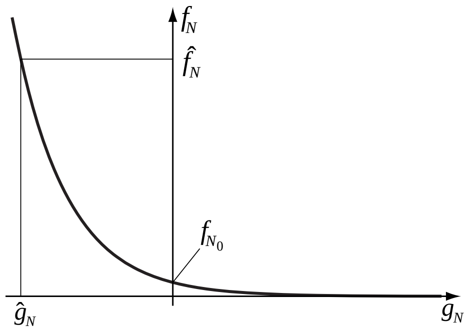

Fig. 1 Normal contact force as a function of the normal gap.

mal gap through the constitutive equation (Fig. 1)

$$f_N(g_N(x)) = f_{N_0}\left(\frac{\hat{f}_N}{f_{N_0}}\right)^{g_N(x)/\hat{g}_N} \tag{2}$$

where $g_N(x)$ is the normal gap function and $\hat{g}_N$ the negative normal gap value (penetration) corresponding to the normal force value $\hat{f}_N$. The (fixed) parameters $(\hat{f}_N, \hat{g}_N)$ provide a straightforward way to properly "calibrate" the model and adapt it to the problem at hand. Relation (2) is a relaxation of the above-stated complementarity condition: it avoids discontinuities while being sufficiently representative of physical reality. In order to describe (unilateral) contacts with friction, the classical Coulomb model is adopted. The focus is placed here on point-contact with *static* friction, whereby normal force f_N, tangential force f_T and the coefficient of static friction μ_s are related by the well known relation

$$f_T \leq \mu_s f_N \tag{3}$$

No-sliding conditions must be enforced by requiring that sliding velocities at contact points be zero. The normal gap $g_N(x)$ and the sliding velocity $\dot{g}_T(\dot{x}, x)$ (i.e., the time derivative of the tangential gap [51]) would call for an additional complementarity condition. We devised a smooth relaxation of this discontinuous condition through the *sliding velocity funnel* shown in Fig. 2 and described by:

$$\dot{g}_T^{(l)}(g_N) \leq \dot{g}_T \leq \dot{g}_T^{(u)}(g_N) \tag{4}$$

where the functions $\dot{g}_T^{(l)}(g_N)$ and $\dot{g}_T^{(u)}(g_N)$ are lower and upper bounds, respectively, for the sliding velocity. It is reasonable to have a symmetric funnel, hence $\dot{g}_T^{(l)}(g_N) = -\dot{g}_T^{(u)}(g_N)$. The bounds are modeled here as:

$$\dot{g}_T^{(u)}(g_N) = \exp\left(c_1(g_N + c_2)\right) + c_3 \tag{5}$$

where the three parameters (c_1, c_2, c_3) are used for proper, problem-dependent calibration. No-sliding manipulation planning benefits from this approach as the sliding velocity funnel smoothly and gradually guides the fingers towards the object, eventually driving relative velocities to (nearly) zero at their contact points. The trajectory planning methods employed in this work are based on numerical optimization techniques that require a discrete-time model of the dynamic system, therefore discrete-time versions of eqs. (1)–(4) are adopted in the following. In our implementation, the state vector $x_k = x(t_k)$ collects configuration and velocity of each body, while control vector $u_k = u(t_k)$ includes acceleration of each finger (or actuator) and contact forces at each candidate contact point. The dynamic equation (1) is used in a direct transcription scheme based on *collocation points*: using a single collocation point as midpoint in the interval $[t_k, t_{k+1}]$,

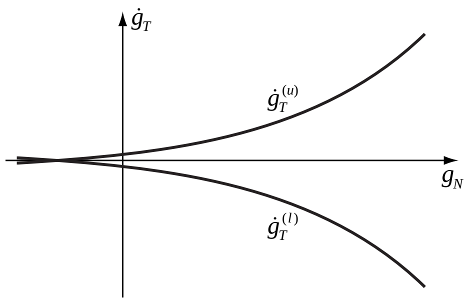

Fig. 2 Example of sliding velocity funnel.

and denoting $\bar{t}_k = (t_{k+1} + t_k)/2$, $\bar{x}_k = x(\bar{t}_k)$ and $h = t_{k+1} - t_k$ (fixed), the discretized dynamic equation becomes

$$x_{k+1} = x_k + hF(\bar{x}_k, u_k) \tag{6}$$

In the above implementation, states are linear and controls are constant over each discretization interval. The integration scheme (6) is known as implicit midpoint rule ($O(h^2)$, and it is the special one-point case of the Gauss-Legendre collocation method. As it is a symplectic integrator, it is suitable to cope with stiff, conservative mechanical systems. While eqs. (3)–(4) are straightforward to discretize, eq. (2) needs some attention: as it involves both states and controls, its discretization must adhere to the scheme dictated by eq. (6), to wit

$$f_{N_k} = f_N\left(g_N(\bar{x}_k)\right) = f_{N_0}\left(\frac{\hat{f}_N}{f_{N_0}}\right)^{g_N(\bar{x}_k)/\hat{g}_N} \tag{7}$$

Failing to do so (e.g., evaluating g_N at x_k) would result in a "causality violation", with contact forces being inconsistent with the interpenetrations between bodies governed by (6).

3.2 *Velocity-based time stepping scheme*

In this section, we present the complementarity formulation of the time-stepping scheme employed for the dynamic modeling of manipulation systems exploiting *sliding over* EC. To keep the treatment general enough, we refer to a 3D system of m bodies with c contacts. We assume a polyhedral approximation of the friction cone, with d friction directions uniformly distributed to positively span the contact tangent plane[1]. For ease of notation, we write $M_k = M(q_k)$ and likewise for other vector/matrix functions. Let h be again the time step, and let $\Delta v := v_{k+1} - v_k$. For $k \in \{0, \ldots, N-1\}$, we adopt a Backward-Euler transcription scheme by writing the *kinematic reconstruction* and the *dynamic* equations as

$$q_{k+1} - q_k - h v_{k+1} = 0 \tag{8a}$$

$$M_{k+1}\Delta v - h\left[\kappa(q_{k+1}, v_k) + B_{k+1}u_{k+1}\right] - G_{k+1}\lambda_{k+1} = 0, \tag{8b}$$

where: $q \in \mathbb{R}^{6m}$ and $v \in \mathbb{R}^{6m}$ represent system configuration and velocity, respectively, $M \in \mathbb{R}^{6m \times 6m}$ is the generalized mass matrix, $\kappa \in \mathbb{R}^{6m}$ collects centrifugal, Coriolis and gravitational forces, $u \in \mathbb{R}^t$ is the control torque vector, $B \in \mathbb{R}^{6m \times t}$ is the actuation matrix, $G = [N \ \ T]$ is a

[1] For simplicity of description, we assume that the number of friction directions d is the same at each contact, although this is not necessary.

generalized grasp matrix, where $N \in \mathbb{R}^{6m\times c}$ and $T \in \mathbb{R}^{6m\times nd}$ are normal and tangential wrench bases, and $\lambda = [\lambda_N^\top \;\; \lambda_T^\top]^\top$ is the generalized wrench impulse vector, wherein λ_N and λ_T are the normal and tangential contact wrench impulses. Matrix N appears as $N = [N^{(1)} \cdots N^{(c)}]$, and each column $N^{(i)} \in \mathbb{R}^{6m}$ corresponds to contact i and contains, for each body ℓ connected to contact i, a block of rows of the form $\pm[n_i^\top \;\; (p_{\ell,i} \times n_i)^\top]^\top$[2]. Since there are at most two bodies connected to a contact, each $N^{(i)}$ has at most 12 non-zero elements. Similarly, $T = [T^{(1)} \cdots T^{(c)}]$, and in the generic block $T^{(i)} \in \mathbb{R}^{6m\times d}$, each column $T^{(i,j)} \in \mathbb{R}^{6m}$ contains, for each body ℓ connected to contact i, a block of rows of the form $\pm[t_{i,j}^\top \;\; (p_{\ell,i} \times t_{i,j})^\top]^\top$, where $t_{i,j}$ denotes friction direction j at contact i. Opposite signs must be selected for each of the two bodies connected to contact i, and each column of T will contain, at most, 12 non-zero elements.

In partial accordance to [47], *unilateral* contacts with friction can be described by the following set of *inequality* and *complementarity* conditions

$$0 \le \lambda_{N_{k+1}} \perp g_N(q_{k+1}) \ge 0 \tag{9a}$$
$$0 \le \lambda_{T_{k+1}} \perp \left(\dot{g}_T(q_{k+1}, v_{k+1}) + E\gamma_{k+1}\right) \ge 0 \tag{9b}$$
$$0 \le \gamma_{k+1} \perp \left[\mu\lambda_{N_{k+1}} - (\lambda_{T_{k+1}}^\top H \lambda_{T_{k+1}})^{\frac{1}{2}}\right] \ge 0, \tag{9c}$$

where $g_N(\cdot)$ is the normal gap function and $\dot{g}_T(\cdot)$ is the time derivative of the tangential gap function [51], γ represents, in most cases[3], an approximation to the magnitude of the relative contact velocity, matrix $E := \mathrm{BlockDiag}(\mathbf{1}, \ldots, \mathbf{1}) \in \mathbb{R}^{(dc)\times c}$, with $\mathbf{1} \in \mathbb{R}^d$, $\mu \ge 0$ is the coefficient of friction, and $H := \mathrm{BlockDiag}(H^{(1)}, \ldots, H^{(c)})$, where the $H_{lm}^{(i)} = t_{i,l}^\top t_{i,m}$ $(l, m \in \{1, \ldots, d\})$ is the metric form [12, Sec. 2-5] of the basis $t_{i,l}$ that positively spans the tangent plane at contact i. Eqs. (9a) state that bodies cannot interpenetrate ($g_N(q_{k+1}) \ge 0$), normal impulses can only push objects away ($\lambda_{N_{k+1}} \ge 0$), and that, in order for the impulse to be non-zero in the interval $[t_k, t_{k+1}]$, the normal gap must be closed at t_{k+1}. This condition also implies that collisions are approximated here as inelastic ones, and interacting bodies may end up sticking together. Eqs. (9b) require tangential impulses to be directed along the positive tangential directions ($\lambda_{T_{k+1}} \ge 0$). The complementarity condition in (9b) selects, for sliding contacts, the tangential impulse that opposes the sliding velocity. This constraint is tightly coupled with the complementarity condition in eqs. (9c), and it ensures that, if a contact is sliding, the tangential force will lie on the boundary of the friction cone. It is worth noting that the bracketed term in eq. (9c) allows one to correctly define the Coulomb friction constraints even in sticking conditions, as it is robust to the physiological failure of eq. (9b) in selecting only one non-zero component in each $\gamma_{k+1}^{(i)}$ for adhesive contacts[4].

The choice of fully implicit integration schemes and nonlinear complementarity formulations, as described in Eqs. (8) and (9), can be justified in view of the increased numerical stability and modelling accuracy they bring about, while not hindering the general structure of the problem[5].

[2] The positive/negative sign must be chosen if, considering equilibrium of body ℓ, the unit normal vector n_i is facing into/away from body ℓ.

[3] In situations where the relative contact velocity and the friction vector are both zero, $\gamma \ge 0$ can be arbitrary and has no physical meaning.

[4] Replacing the bracketed term in (9c) with $[\mu\lambda_N - E^\top\lambda_T]$, as commonly performed in literature [48], would call for unrealistically strict and physically unmotivated conditions to ensure adhesive friction.

[5] Embedding contact dynamics into the numerical optimization problem as nonlinear constraints, where many other implicit constraints are already present, does not justify explicit or semi-implicit discretization schemes, which are, instead, legitimate when building fast simulators [20, Sec. 5].

4 TRAJECTORY PLANNING AS AN OPTIMIZATION PROBLEM

4.1 Penalty-based contact model

Within our direct transcription framework, for $k \in \{0,\dots,N-1\}$, eqs. (6), (7), and the discretized versions of eqs. (3) and (4) constitute a set of (equality and inequality) *nonlinear constraints* for the optimal control problem (OCP) we set out to formulate. Additional constraints include the (fixed and known) initial state values $x_0 = x(0)$ as well as a *terminal equality constraint* on (some) components of x_N: the latter provides a direct way to specify the required final state of the manipulated object at the final time $T = t_N$. Generally, no terminal constraints on the manipulation system configuration are imposed. Lower and upper bounds for $(x_k, \bar{x}_k, u_k)$ are also included: they act as operational constraints for actuators and the system's workspace, and they are useful to restrain contact forces within safety limits. Other constraints can be introduced to shape emergent behaviors and to render them intrinsically more robust or desirable for several reasons. As an illustrative example, in order to guarantee that any two fingers make always contact with an object in a three-fingered manipulation task, we add the following set of inequalities to the problem: $f^{(1)}_{N_k} + f^{(2)}_{N_k} \geq \varepsilon$, $f^{(2)}_{N_k} + f^{(3)}_{N_k} \geq \varepsilon$, and $f^{(3)}_{N_k} + f^{(1)}_{N_k} \geq \varepsilon$, with $\varepsilon > 0$.

We introduce the vector of decision variables $\boldsymbol{v} \in \mathbb{R}^n$, which collects the sequence of unknown $(x_k, \bar{x}_k, u_k)$ (i.e., configurations and velocities in $(x_k, \bar{x}_k)$, contact forces and actuator accelerations in u_k). All equality and inequality constraints can be written compactly as

$$g_{\min} \leq g(\boldsymbol{v}) \leq g_{\max}, \qquad \boldsymbol{v}_{\min} \leq \boldsymbol{v} \leq \boldsymbol{v}_{\max} \tag{10}$$

The general structure of the cost function takes the form

$$f(\boldsymbol{v}) = \sum_{k=0}^{N-1} \sum_{i \in \mathscr{I}} w_i \phi_i(x_k, u_k), \tag{11}$$

where each $\phi_i(\cdot)$ represents a peculiar type of cost. These have to be carefully selected according to the character of the manipulation action we desire to perform, along with the corresponding *weights* w_i (also acting as important scaling factors). Multiple cost terms $\phi_i(\cdot)$ can be used to shape different manipulation behaviors. The following terms (specifically, their squared 2-norm) have proved to be decisive in directing the optimization process: contact forces, and their variations from one interval to the next (to minimize jerk); accelerations of actuators; deviations of actual object trajectories from ideal, smooth trajectories (task-specific).

4.2 Velocity-based time stepping scheme

Similarly to the penalty-based contact model scheme, the discrete formulation of the dynamics of systems with contacts expressed by eqs. (8) and (9) can be used as a set of nonlinear constraints in an optimal control problem (OCP), involving the sequence of unknown $(q_k, v_{k+1}, \lambda_{k+1}, \gamma_{k+1})$. Additional equality constraints are introduced: for the manipulated object, both initial and final configurations and velocities are imposed, whereas only the initial conditions are specified for the manipulation system.

Inequality constraints are also introduced with similar intentions as in the penalty-based scheme. It is worth noting that, since in our applications the hand is velocity controlled, the hand dynamics is not included in the optimization constraints, and the hand velocities will play the role of control actions. Therefore, limited control authority is imposed as bounds on hand velocities and accelerations in the form $v_{\min}^{(l)} \leq v^{(l)} \leq v_{\max}^{(l)}$ and $a_{\min}^{l} h \leq \Delta v^{(l)} \leq a_{\max}^{(l)} h$, respectively, with l belonging to the index set corresponding to the hand.

Defining $\boldsymbol{v} \in \mathbb{R}^n$ as the multi-stage sequence of configurations, velocities, contact impulses and inputs, $(q_k, v_{k+1}, \lambda_{k+1}, \gamma_{k+1})$, all constraints are still expressed in the form (10), whereas the cost function takes the form

$$f(\boldsymbol{v}) = \sum_{k=0}^{N-1} \sum_{i \in \mathscr{I}} w_i \phi_i(q_k, v_{k+1}, \lambda_{k+1}, \gamma_{k+1}) \tag{12}$$

4.3 *Final optimization problem*

From the previous discussion, we now present the nonlinear program (NLP) that has to be solved to generate optimal trajectories. With $\boldsymbol{v} \in \mathbb{R}^n$ previously defined, we consider the following optimization problem

$$\min_{\boldsymbol{v}} f(\boldsymbol{v}), \qquad \text{subject to} \quad g_{\min} \leq g(\boldsymbol{v}) \leq g_{\max} \quad \boldsymbol{v}_{\min} \leq \boldsymbol{v} \leq \boldsymbol{v}_{\max} \tag{13}$$

in which: $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $g : \mathbb{R}^n \to \mathbb{R}^m$ is the nonlinear constraint function, $g_{\min} \in [-\infty, \infty)^m$ and $g_{\max} \in (-\infty, \infty]^m$ (with $g_{\min} \leq g_{\max}$) are, respectively, lower and upper bound vectors of the nonlinear constraints, $\boldsymbol{v}_{\min} \in [-\infty, \infty)^n$ and $\boldsymbol{v}_{\max} \in (-\infty, \infty]^n$ (with $\boldsymbol{v}_{\min} \leq \boldsymbol{v}_{\max}$) are, respectively, lower and upper bound vectors of the decision variables. Problem (13) is a *large-scale*, but *sparse* NLP, that should be solved by structure-exploiting solvers. To this end, as detailed in the next section, we resorted to the IPOPT [50] implementation of the interior-point method within the CasADi framework. As initial guess required by the algorithm, we somewhat crudely mapped the initial state x_0 and a rough estimate of the controls to all the $(N-1)$ variable instances. Obviously, better initial guesses should be provided whenever possible. With the penalty-based approach, (partial) solutions of the NLP (13) have also been used as initial guesses for a subsequent optimization according to a *homotopy* strategy [39, Sec. 11.3], thereby maximizing physical realism while facilitating convergence.

5 NONLINEAR PROGRAMMING VIA AN INTERIOR-POINT ALGORITHM

5.1 *The barrier problem formulation*

Problem (13) is equivalently rewritten as

$$\min_{x} f(x), \qquad \text{subject to} \tag{14a}$$
$$c(x) = 0 \tag{14b}$$
$$x_{\min} \leq x \leq x_{\max} \tag{14c}$$

in which x is formed by augmenting the decision variable vector $\boldsymbol{v}$ with suitable slack variables that transform inequality constraints in (13) into equality constraints.[6]

Let $I_{\min} = \{i : x_{\min}^{(i)} \neq -\infty\}$, and $I_{\max} = \{i : x_{\max}^{(i)} \neq \infty\}$. We consider the barrier function

[6] With a slight abuse of notation, we still use n and m to denote the dimension of x and $c(x)$, respectively, and $f(x)$ to denote $f(\boldsymbol{v})$.

$$\varphi_\mu(x) = f(x) - \mu \left(\sum_{i \in I_{\min}} \ln(x^{(i)} - x^{(i)}_{\min}) + \sum_{i \in I_{\max}} \ln(x^{(i)}_{\max} - x^{(i)}) \right)$$

in which $\mu > 0$ is a (small) barrier parameter. Instead of solving (14), IPOPT performs iterations to achieve an approximate solution of the equality constrained NLP

$$\min_x \varphi_\mu(x), \qquad \text{subject to: } c(x) = 0 \tag{15}$$

Note that $\varphi_\mu(x)$ is well defined if and only if $x_{\min} < x < x_{\max}$, i.e. if x is in the *interior* of its admissible region. The value of μ is progressively reduced so that $\varphi_\mu(x) \to f(x)$, and in this way solving (15), in the limit, becomes equivalent to solving (14). Clearly, as $\mu \to 0$, any component of x can approach its bound if this is required by optimality.

5.2 *Interior-point approach to NLP*

Any local minimizer to (15) must satisfy the following Karush-Kuhn-Tucker (KKT) conditions [39, Sec. 12.2]

$$\nabla f(x) + \nabla c(x)\lambda - \underline{z} + \bar{z} = 0 \tag{16a}$$
$$c(x) = 0 \tag{16b}$$
$$\underline{z}^{(i)}(x^{(i)} - x^{(i)}_{\min}) - \mu = 0 \quad \forall i \in I_{\min} \tag{16c}$$
$$\bar{z}^{(i)}(x^{(i)}_{\max} - x^{(i)}) - \mu = 0 \quad \forall i \in I_{\max} \tag{16d}$$

for some vectors $\lambda \in \mathbb{R}^m$, $\underline{z} \in \mathbb{R}^n$, and $\bar{z} \in \mathbb{R}^n$ (for completeness: $\underline{z}^{(i)} = 0 \quad \forall i \notin I_{\min}$, $\bar{z}^{(i)} = 0 \quad \forall i \notin I_{\max}$). Notice that, if $\mu = 0$, then (16) together with $\underline{z} \geq 0$ and $\bar{z} \geq 0$ represent the KKT conditions for NLP (14). The KKT conditions (16) form a nonlinear algebraic system $F(\xi) = 0$ in the unknown $\xi = (x, \lambda, \underline{z}, \bar{z})$, which is solved in interior-point algorithms via Newton-like methods. If we denote by $E_\mu(\xi)$ the maximum absolute error of the KKT equations (16) (appropriately scaled), the basic algorithm implemented in IPOPT is summarized in Table 1 (in which j is the index of the outer loop, k is the index of the inner loop, and $\varepsilon > 0$ is a user-defined convergence tolerance).

Table 1 Basic Algorithm implemented in IPOPT.

1. Define $\mu_0 > 0$, x_0 ($x_{\min} \leq x_0 \leq x_{\max}$), λ_0, $\underline{z}_0 \geq 0$, $\bar{z}_0 \geq 0$, and form ξ_0 accordingly. Set: $j = 0$, $k = 0$.
2. Given the current iterate ξ_k, compute a Newton step p_k for $F(\xi) = 0$. Compute the new iterate performing a line search: $\xi_{k+1} = \xi_k + \alpha_k p_k$ (for some $\alpha_k > 0$).
3. If $E_0(\xi_{k+1}) \leq \varepsilon$, exit: ξ_{k+1} is a local solution to NLP (14). Otherwise, proceed to Step 4.
4. If $E_{\mu_j}(\xi_{k+1}) \leq \kappa\mu_j$ (for some $\kappa > 0$) proceed to Step 5. Otherwise, update $k \leftarrow k+1$ and go to Step 2.
5. Set $\mu_{j+1} = \mu_j/\rho$ (for some $\rho > 1$), update $j \leftarrow j+1$, $k \leftarrow k+1$ and go to Step 2.

5.3 Main computational aspects: calculating derivatives and solving (sparse) linear systems

The most expensive computation step in the basic interior-point algorithm is the computation of the Newton step p_k for the KKT system $F(\xi) = 0$, i.e. Step 2. We first note that evaluation of $F(\xi)$, at each iteration, involves the computation of the cost function gradient $\nabla f(x) \in \mathbb{R}^n$ and of the constraint Jacobian $\nabla c(x) \in \mathbb{R}^{n\times m}$. Then, the Newton step is found from the solution of the following linear system:

$$\begin{bmatrix} W_k & A_k & -I & I \\ A_k^T & 0 & 0 & 0 \\ \underline{Z}_k & 0 & \underline{X}_k & 0 \\ -\overline{Z}_k & 0 & 0 & \overline{X}_k \end{bmatrix} \begin{bmatrix} p_k^x \\ p_k^\lambda \\ p_k^{\underline{z}} \\ p_k^{\overline{z}} \end{bmatrix} = - \begin{bmatrix} \nabla \varphi_{\mu_j}(x_k) + A_k \lambda_k \\ c(x_k) \\ \underline{X}_k \underline{Z}_k \mathbf{1} - \mu_j \mathbf{1} \\ \overline{X}_k \overline{Z}_k \mathbf{1} - \mu_j \mathbf{1} \end{bmatrix} \tag{17}$$

in which: $W_k = \nabla^2_{xx} \mathscr{L}(x_k, \lambda_k, \underline{z}_k, \overline{z}_k)$, with $\mathscr{L}(x, \lambda, \underline{z}, \overline{z}) = f(x) + c(x)^T \lambda - (x - x_{\min})^T \underline{z} - (x_{\max} - x)^T \overline{z}$ the Lagrangian function associated with NLP (14); $A_k = \nabla c(x_k)$ the constraint Jacobian; $\underline{Z}_k = \mathrm{diag}(\underline{z}_k)$, $\overline{Z}_k = \mathrm{diag}(\overline{z}_k)$, $\underline{X}_k = \mathrm{diag}(x_k - x_{\min})$, and $\overline{X}_k = \mathrm{diag}(x_{\max} - x_k)$ diagonal matrices; $\nabla \varphi_{\mu_j}(x_k) = \nabla f(x_k) - \underline{z}_k + \overline{z}_k$. In order to generate the entries of system (17), it is necessary to evaluate the cost function gradient, the constraint Jacobian, as well as the Hessian of the Lagrangian (or a suitable approximation to it). Partial derivatives can be computed numerically by finite differentiation or analytically (for simple functions). A third approach is by means of so-called *Automatic Differentiation* (or Algorithmic Differentiation) techniques, which generate a numerical representation of partial derivatives by exploiting the chain rule in a numerical environment. Different approaches exist for AD, which are tailored to the computation of first-order and second-order derivatives. The interested reader is referred to [21]. A final computation observation is reserved to the numerical solution of system (17). First, it is transformed into a symmetric (indefinite) linear system via block elimination. Then, symmetry can be exploited by symmetric LDL factorizations. Furthemore, it should be noted that in trajectory planning problems considered here (and in general in optimal control problems) the Hessian W_k and the constraint Jacobian A_k are significantly sparse and structured. Exploiting these features can reduce the solution time significantly. To this effect, the MA57 multifrontal solver [14] from the Harwell Software Library [23] is used.

5.4 The CasADi framework

The transcribed optimal control problem is coded in a scripting environment using the Python [40] interface to the open-source CasADi framework [2], which provides building blocks to efficiently formulate and solve large-scale optimization problems.

In the CasADi framework, symbolic expressions for objective and constraints are formed by applying overloaded mathematical operators to symbolic primitives. These expressions are represented in memory as computational graphs, in contrast to tree representations common to computer algebra systems. The graph is sorted into an in-memory algorithm which can be evaluated numerically or symbolically with an efficient stack-based virtual machine or be exported to C code. Forward and backward source-code transforming AD can be performed on such algorithm at will, such that derivatives of arbitrary order can be computed. The sparsity pattern of the constraint Jacobian is computed using hierarchical seeding [18] and its unidirectional graph coloring is used to obtain the Jacobian with a reduced number of AD sweeps [17]. Regarding expressions and algorithm inputs and outputs, everything is a sparse matrix in CasADi. Yet the underlying computational graphs may be of either a type with scalar-valued (SX) nodes or a type with matrix-valued (MX) nodes. The combined usage of these two types amounts to a checkpointing scheme [21]: low-level functions are constructed with the SX type algorithm, which is optimized for speed. These algorithms are in turn embedded into a graph of the MX type, which is optimized for memory usage, to form the expression of objective and constraints.

In the context of optimal control problems, the CasADi framework offers several advantages over other AD tools: it comes bundled with common algorithms that can be embedded into an infinitely differentiable computional graph (e.g. numerical integrators, root-finding and linear solvers), and takes care of constructing and passing sensitivity information to various NLP solvers backends. Since CasADi does not impose an OCP solution strategy and allows fine-grained speed-memory trade-offs, it is suited more than black-box OCP solvers to explore non-standard optimal control problem formulations or efficient solution strategies.

6 APPLICATION EXAMPLES

6.1 Environment-aware manipulation

6.1.1 Penalty-based approach with disk and two independent fingers

Figure 3 shows a first example of EC-exploiting manipulation. In a vertical plane, the two independent fingers H_0 and H_1, initially away from the circular object, must interact with the object and have it interact with the environment (edges e_0 and e_1) so that it will be in the shown final position, with any orientation but zero velocity, at the end of a prescribed time horizon T. All contact interactions must occur without slippage (static friction). The object's initial state corresponds to a configuration of static equilibrium. The fingers have limitations on their horizontal workspace: as a result, grasping and lifting of the object is inhibited, and an environment-exploiting policy needs to be discovered in order to accomplish the task. Also, object-passing between fingers needs to emerge. This planning problem has been formulated and solved using the penalty-based approach. The resulting trajectories in terms of normal contact forces are shown in the first two plots of Fig. 4: finger H_0 approaches the object first (whose weight is symmetrically supported by e_0 and e_1), then rolls it on edge e_1 (without slipping) until it reaches its workspace limit and hands it over to finger H_1, which completes the task. Friction forces (not shown for brevity) satisfy constraint (3), where $\mu_s = 2$ was used. The third plot shows the actual x-component trajectory of the object versus a suggested trajectory, included as a hint in the objective function to facilitate convergence of the algorithm, but with a low weight (to avoid forcing such trajectory against dynamic constraints). With $N = 180$ discretization intervals (time step $h = 45$ ms) and considering the prescribed initial and terminal conditions, the problem size is $n = 8810$ decision variables. To obtain a solution for this example, starting from an initial guess built by repeating the initial condition for all the time steps, IPOPT took 1062 iterations, which correspond to ~ 23 min on a 2.70 GHz Intel(R) Core(TM) i7-4800MQ CPU with 32 GB of RAM. An animation of the obtained results can be found in part A.1 of the accompanying video [16] which also shows the grasping-and-lifting behavior that is discovered if finger workspace limitations are removed and the contact force exerted by edge e_1 is penalized.

6.1.2 Velocity-based time-stepping scheme with capsule and two-fingered underactuated gripper

With reference to Fig. 5, a capsule-shaped object, starting from an equilibrium configuration in contact with segment P_2P_3 (of a six-edged polygonal environment), has to find itself rotated by 180 deg at the end of the planning horizon T. Since the object is passive, a manipulation gait has to emerge for the gripper. Moreover, since we penalize high contact impulses and the gripper has a reduced mobility due to underactuation — it has symmetrically closing jaws — it turns out

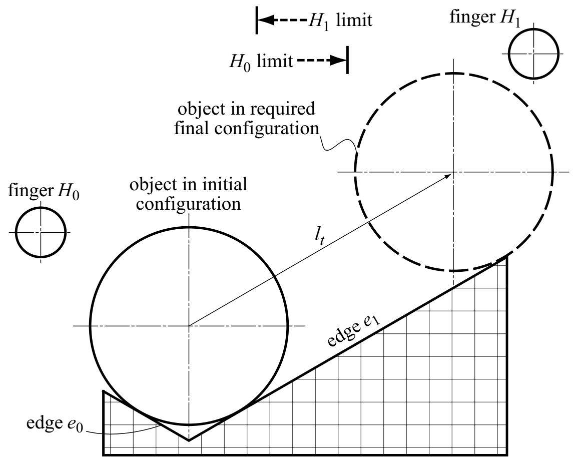

Fig. 3 EC manipulation scenario: the object must reach its final configuration at the prescribed final time $T = 8$ s.

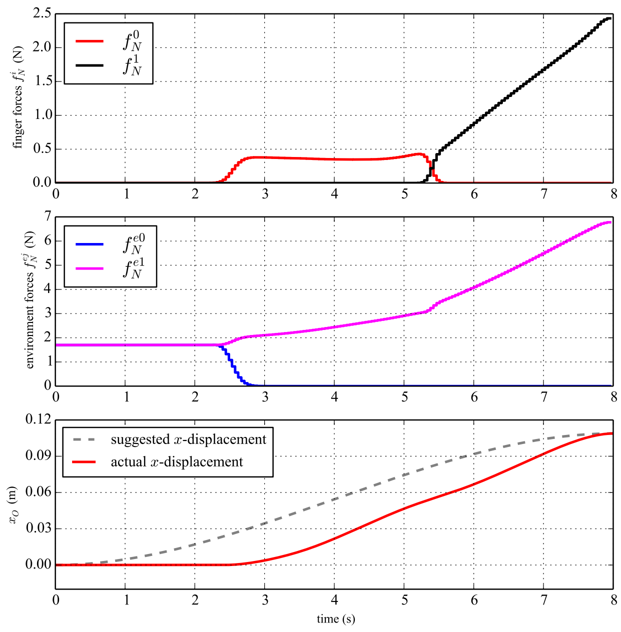

Fig. 4 Trajectories for a circular object manipulated by two independent fingers: normal contact forces f_N^i applied by the fingers; normal contact forces f_N^{ej} applied by the environment (segments e_0, e_1); suggested and actual x-displacement x of the object.

that convenient EC-exploiting behaviors are indeed automatically synthesized by the optimizer. In fact, with reference to Fig. 6, beside finger impulses (first plot), which represent standard grasping/manipulation actions, contact interactions generated by collisions of the object with the environment (second plot) play a role of paramount importance in shaping the object motion. More in detail, with reference to part A.2 of the accompanying video [16], the object is initially grasped and lifted, then it is gently dropped so that it lays on segment P_2P_3 after hitting segment P_3P_4 (see the corresponding bumps in λ_N^{23} and λ_N^{34}). Then, with the circular part of the capsule pushed to corner P_3, the object is rotated with only one finger by sliding it on edges P_2P_3 and

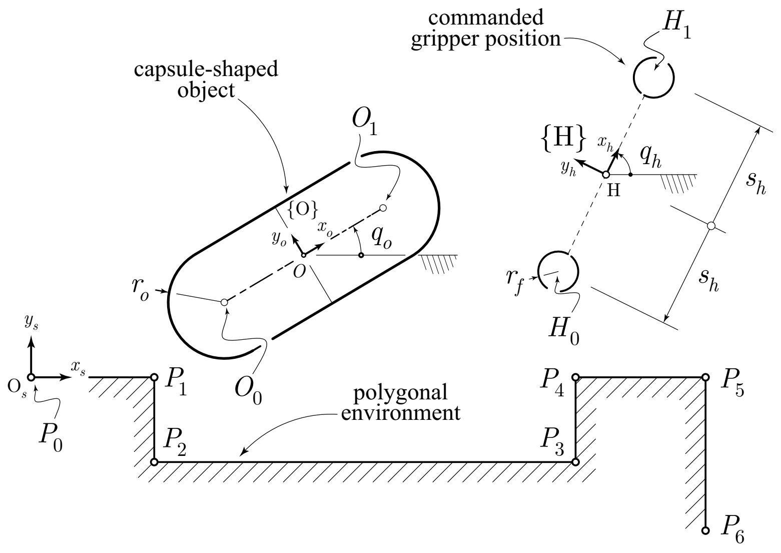

Fig. 5 EC manipulation scenario. Starting at $q_O(0) = \pi/2$, in contact with segment P_2P_3, the capsule must be placed, at time $T = 5$ s, in the same position but with $q_O(T) = -\pi/2$.

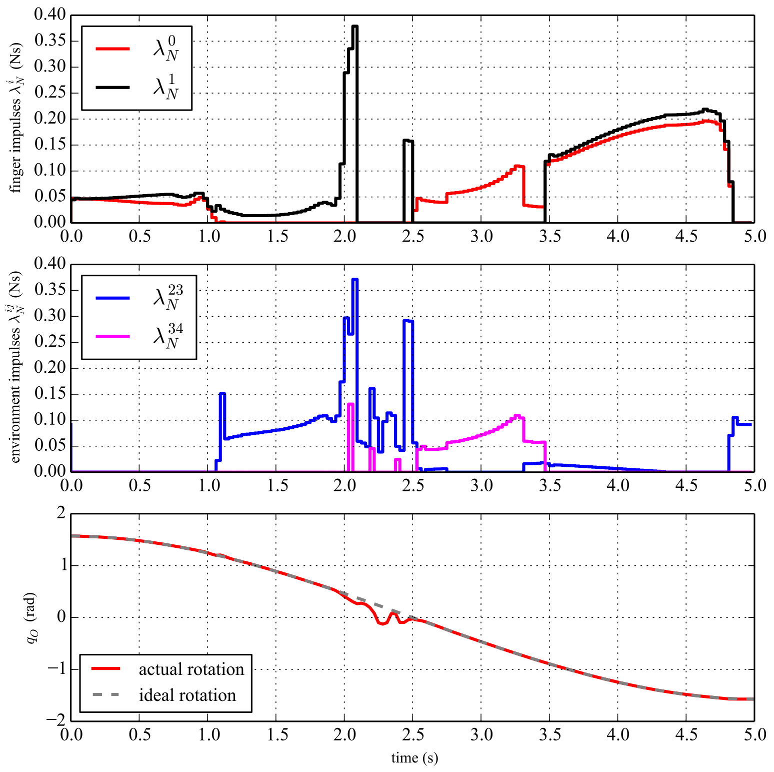

Fig. 6 Motion trajectories of a capsule-shaped object manipulated by an underactuated gripper: contact normal impulses λ_N^0 and λ_N^1 applied by the fingers; contact normal impulses λ_N^{ij} applied by the environment through segment P_iP_j; ideal and actual rotation q_O of the object-fixed frame.

P_3P_4. Finally, both fingers grasp the object and, slightly lifting it up, they slide it on edge P_2P_3 to the initial position. It is worth noting that, with a wise exploitation of EC, the actual rotation of the object can closely follow the desired one (third plot). This condition can be violated in general, since the trajectory prescribed from the outset only constitutes a suggested behavior for some components of the system. Regarding the underlying numerical OCP, at each time step $k \in \{0, \ldots, N-1\}$, the problem variables have the following dimensions: $q_k \in \mathbb{R}^{4+3}$, $v_{k+1} \in \mathbb{R}^{4+3}$, $\lambda_{N_{k+1}} \in \mathbb{R}^{6+2}$, $\lambda_{T_{k+1}} \in \mathbb{R}^{12+4}$, $\gamma_{k+1} \in \mathbb{R}^{6+2}$. With $N = 160$ discretization intervals ($h = 30$ ms) and considering the prescribed boundary conditions on the object/gripper, the problem size is $n = 7375$. To obtain a solution for this example, starting from an initial guess built by repeating the initial condition for all the time steps, IPOPT took 920 iterations, which correspond to ~ 25 min on a 2.70 GHz Intel(R) Core(TM) i7-4800MQ CPU with 32 GB of RAM. An animation of the results can be found in part A.2 of the accompanying video [16].

6.2 *Dexterous manipulation*

With reference to Fig. 7, a circular object, starting from a configuration where it is held in equilibrium by three independent fingers in a force-closure grasp, has to find itself rotated by 360 deg at the end of the planning horizon T, with zero velocity. Since, again, the object itself is passive and each finger has workspace limitations (see Fig. 7), a relatively complex (dexterous) manipulation gait has to be discovered for the fingers. To obtain an in-place manipulation, a box constraint has also been assigned on the position of the object center. In order to obtain a relatively robust manipulation, the constraint described in section 4.1 has been included to guarantee that any two fingers are always in contact. As we want no slipping between the fingers and the object during manipulation, the penalty-based approach has been used (with a coefficient of friction $\mu_s = 1.5$). The resulting optimal trajectories in terms of finger forces are shown in the first two plots of Fig. 8: intermittent contacts due to the discovered manipulation gait can be clearly seen. The third plot shows the object's actual rotation versus a smooth (third-order), suggested rotation trajectory. With $N = 200$ discretization intervals ($h = 30$ ms) and accounting for the fixed initial and terminal conditions, the problem size is $n = 12585$. To obtain a solution for this example, starting from an initial guess built by repeating the initial condition for all the time steps, IPOPT took 1661 iterations, which correspond to ~ 81 min on a 2.70 GHz Intel(R) Core(TM) i7-4800MQ CPU with 32 GB of RAM. An animation is provided in part B.1 of the accompanying video [16], while part B.2 shows the results obtained by solving the same problem with the velocity-based time-stepping scheme, where sliding between fingers and object is allowed and exploited.

7 CONCLUSIONS AND FUTURE WORK

This paper proposed a computational framework to plan environment-aware manipulation behaviors that do not rely on an a-priori defined sequences of contacts. To this end, we framed the problem as a numerical optimal control one, including contact forces among the optimization variables as a key factor, and we sharpened the algorithmic pipeline by exploiting structural sparsity and leveraging Automatic Differentiation. Two contact models were proposed that best fit manipulation scenarios where sliding primitives need to be avoided or sought, respectively. These proved effective in solving manipulation planning problems where essential interactions with the environment had to be synthesized to accomplish a task (sub-section 6.1). The results presented in sub-section 6.2 demonstrated that the very same method is able to perform successfully in discovering non-trivial gaits also in dexterous manipulation tasks. Current research is devoted to extending the method to 3D scenarios, the major thrust being the synthesis of EC-exploiting, whole-body manipulation strategies for humanoid platforms. Injection of motion

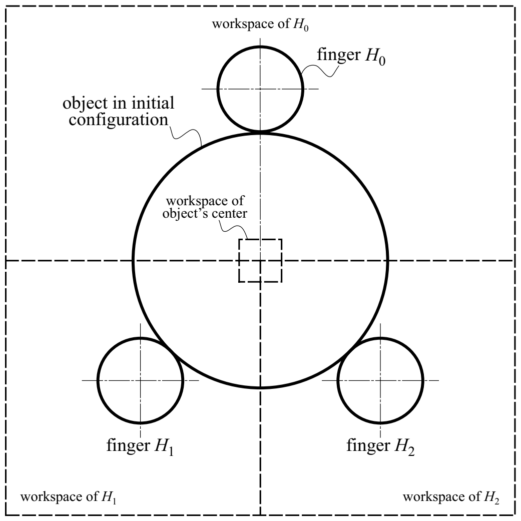

Fig. 7 Dexterous manipulation scenario: the object must find itself rotated by 360 deg at the final time $T = 6$ s.

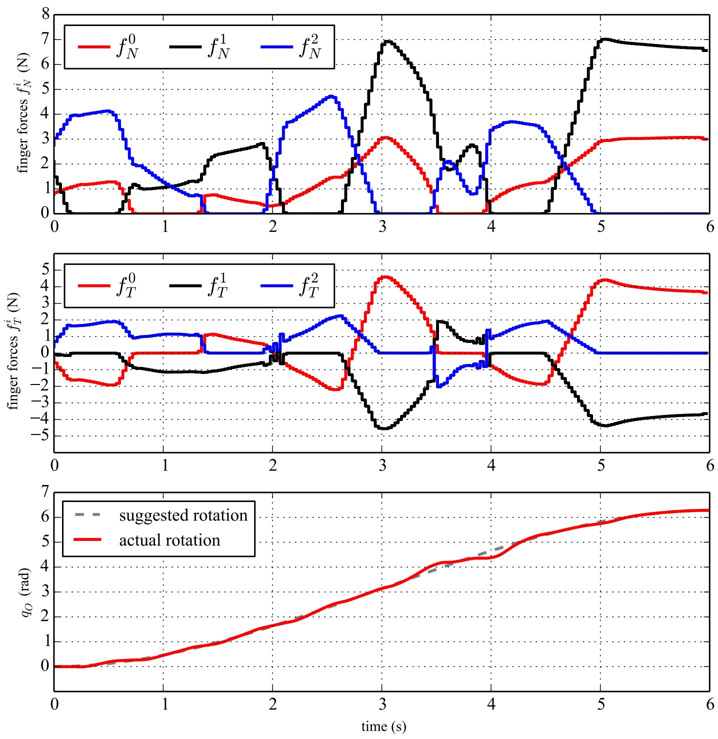

Fig. 8 Trajectories for a circular object manipulated by three independent fingers. Shown are: normal contact forces f_N^i and tangential contact forces f_T^i applied by the fingers; suggested and actual rotation q_O of the object.

primitives/synergies into the model are also being considered, and proper model scaling and tuning of IPOPT convergence parameters are under way to maximize computational efficiency.

8 ACKNOWLEDGEMENT

This work is supported by the grant no. 600918 "PACMAN" - Probabilistic and Compositional Representations of Object for Robotic Manipulation - within the FP7-ICT-2011-9 program "Cognitive Systems". European Research Council and under the ERC Advanced Grant no. 291166 SoftHands (A Theory of Soft Synergies for a New Generation of Artificial Hands).

References

1. A. P. Ambler and R. J. Popplestone. Inferring the positions of bodies from specified spatial relationships. *Artificial Intelligence*, 6:157–174, 1975.
2. Joel Andersson. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.
3. J. T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, 2001.
4. L. T. Biegler and V. M. Zavala. Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575 – 582, 2009.
5. M. Bonilla, E. Farnioli, C. Piazza, M. Catalano, G. Grioli, M. Garabini, M. Gabiccini, and A. Bicchi. Grasping with soft hands. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2014.
6. G. Boothroyd, C. Poli, and L.E. Murch. Handbook of feeding and orienting techniques for small parts. University of Massachusetts at Amherst, Dept. of Mechanical Engineering, Automation Project, 1981.
7. Iain Boyle, Yiming Rong, and David C. Brown. A review and analysis of current computer-aided fixture design approaches. *Robotics and Computer-Integrated Manufacturing*, 27(1):1–12, February 2011.
8. B. Cohen, M. Phillips, and M. Likhachev. Planning single-arm manipulations with n-arm robots. In *Robotics: Science and Systems (RSS)*, 2014.
9. N.C. Dafle, A. Rodriguez, R. Paolini, Bowei Tang, S.S. Srinivasa, M. Erdmann, M.T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge. Regrasping objects using extrinsic dexterity. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2560–2560, May 2014.
10. G.E. Deacon. An attempt to raise the level of software abstraction in assembly robotics through an apposite choice of underlying mechatronics. *Journal of Intelligent and Robotic Systems*, 28(4):343–399, 2000.
11. R. Diankov. *Automated Construction of Robotic Manipulation Programs*. Phd thesis, Carnegie Mellon University, Robotics Institute, August 2010.
12. Manfredo Perdigao Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall, Englewood Cliffs, 1976.
13. Mehmet R. Dogar and Siddhartha S. Srinivasa. A planning framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots*, 33(3):217–236, June 2012.
14. Iain S Duff. Ma57 – a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software (TOMS)*, 30(2):118–144, 2004.
15. Clemens Eppner and Oliver Brock. Planning grasp strategies that exploit environmental constraints. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2015.
16. M. Gabiccini, A. Artoni, G. Pannocchia, and J. Gillis. ISRR 2015 submission accompanying video. http://youtu.be/ozDwPnS00fI, April 2015.
17. Assefaw Hadish Gebremedhin, Fredrik Manne, and Alex Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review*, 47:629–705, 2005.
18. J. Gillis and M. Diehl. Hierarchical seeding for efficient sparsity pattern recovery in automatic differentiation. In *CSC14: The Sixth SIAM Workshop on Combinatorial Scientific Computing*, 2014.
19. E. Glassman and R. Tedrake. A quadratic regulator-based heuristic for rapidly exploring state space. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 5021–5028, Anchorage, Alaska, USA, 5 2010.
20. C. Glocker and C. Studer. Formulation and preparation for numerical evaluation of linear complementarity systems in dynamics. *Multibody System Dynamics*, 13:447–463, 2005.
21. A. Griewank and A. Walther. *Evaluating Derivatives*. SIAM, 2 edition, 2008.
22. Andreas Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, PA, 2000.
23. HSL. A collection of fortran codes for large scale scientific computation, 2014.

24. A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. 2003.
25. Rico Jonschkowski and Oliver Brock. State representation learning in robotics: Using prior knowledge about physical interaction. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.
26. S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, June 2011.
27. Moslem Kazemi, Jean-Sebastien Valois, J.Andrew Bagnell, and Nancy Pollard. Human-inspired force compliant grasping primitives. *Autonomous Robots*, pages 1–17, 2014.
28. J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *Int. Journal of Robotics Research (IJRR)*, 32(11):1238–1274, 2013.
29. Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1994.
30. N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2004.
31. Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Neural Information Processing Systems (NIPS)*, 2014.
32. Sergey Levine, Nolan Wagener, and Pieter Abbeel. Learning contact-rich manipulation skills with guided policy search. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.
33. Tomás Lozano-Pérez, Matthew T. Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. *Int. Journal of Robotics Research (IJRR)*, 3(1):3–24, March 1984.
34. Matthew T. Mason. The mechanics of manipulation. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 544–548, 1985.
35. A.T. Miller and P.K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics and Automation Magazine*, 11(4):110–122, 2004.
36. I. Mordatch, Z. Popović, and E. Todorov. Contact-invariant optimization for hand manipulation. In *Eurographics/ACM Symposium on Computer Animation*, 2012.
37. I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. In *ACM Transactions on Graphics*, 2012.
38. Igor Mordatch and Emo Todorov. Combining the benefits of function approximation and trajectory optimization. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.
39. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Verlag, New York, 2006.
40. Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.
41. A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez. LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012.
42. J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.
43. M. Posa, C. Cantu, and R. Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *Int. Journal of Robotics Research (IJRR)*, 33(1):69–81, 2014.
44. C. Samson, M. Le Borgne, and B. Espiau. *Robot Control, the Task Function Approach*. Clarendon Press, Oxford, England, 1991.
45. G. Schultz and K. Mombaur. Modeling and optimal control of human-like running. *IEEE/ASME Transactions on Mechatronics*, 15(5):783–792, 2010.
46. Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decré, Ruben Smits, Erwin Aertbeliën, Kasper Claes, and Herman Bruyninckx. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *International Journal of Robotics Research*, 26(5):433–455, 2007.
47. D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *Int. Journal for Numerical Methods in Engineering*, 39:2673–2691, 1996.
48. David Stewart and Jeffrey C Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 162–169. IEEE, 2000.
49. R. Tedrake, T. Zhang, and H. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2004.
50. A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.
51. Peter Wriggers. *Computational contact mechanics*. Springer, Berlin, New York, 2006.
52. W. Xi and C.D. Remy. Optimal gaits and motions for legged robots. In *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3259–3265, 2014.