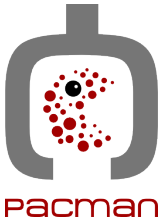

FP7-IST-60918
1 March 2013 (36months)

DR 4.3:
Methodologies for grasp acquisition and planning

H. Marino, C. Rosales, M. Kopicki, G. Santaera, E. Luberto, Y. Wu, T. Sartor, J.L. Wyatt, and M. Gabiccini

Università di Pisa, Pisa Italy
⟨m.gabiccini@ing.unipi.it⟩

Due date of deliverable:	2016-02-29
Actual submission date:	2016-02-29
Lead partner:	Università di Pisa
Revision:	final
Dissemination level:	PU

This report describes the activities related to the control of grasping actions given the detection of an object. It includes work on: modelling underactuated and adaptive robotic hands, acquiring target grasps learned from data gathered either from humans or simulations, adapting grasps via infra-red sensor readings to objects in an uncertain configuration with the respect to that perceived by vision, planning complex grasping and manipulation tasks either using discrete search methods or via direct trajectory optimization methods for systems with unspecified contact sequences.

Executive Summary

This report describes the activities — most of which are already performed, while some others scheduled for finalization in the weeks that precede the PaCMan review meeting in May 2015 — within the PaCMan consortium to define methodologies for *incipient grasping evaluation*. The material included in this report describes the results that fall under Task 4.2 (M 7-18), and the progress of Task 4.3 (M 7-30) at M 24.

Role of incipient grasp evaluation in PaCMan

This deliverable documents the effort of the consortium in devising new strategies to define *practical* approaches to tackle the problem of *grasping under uncertainty and novelty*, either caused by clutter, visual occlusions, or new object shapes. To this sake, the ability to evaluate *incipient grasps* in terms of their expected quality, robustness, and probability of success, is of paramount importance.

Contribution to the PaCMan scenario

To our eyes, grasp methodologies that try to cope with pose uncertainty, object shape estimation inaccuracies, clutter and partial occlusions, either (i) by accomodating grasp unmodelled dynamics by exploiting prior knowledge about the adaptivity and the compliance of the Pisa/IIT SoftHand (as mainly proposed by the UNIPI Team), or (ii) by inferring grasps for novel objects from as little as one training example (of a chosen grasp type) that are shown to be generalizable within and *across* object categories, even with partial shape information, (as mainly proposed by the UoB Team), represent central contributions in the scenario depicted in the PaCMan project.

1 Tasks, objectives, results

1.1 Planned work

The selection of correct grasp strategies when the object to be grasped in unknown or only partially visible seems a pretty easy task for humans. On the contrary, though robotics has made significant progress in the field of autonomous grasping, the problem of grasping novel objects in cluttered scenarios has not been completely solved yet. In this report, we document the activities of the consortium to tackle this problem.

The work to be documented in this report is concerned with the definition of: "Methodologies for grasp acquisition and planning". This should mostly contain the results of Task 4.4. In particular, the report should contain the definition of feasibility analysis of grasping under uncertainty as a belief space planning problem, feasibility analysis for extending Monte-Carlo tree search methods to continuous state-action space, algorithm design for the chosen approach.

1.2 Actual work performed

We have to clearly state, right away, that not all the pieces of work presented in this report, even if sharing the very same goals of WP4, i.e., "to develop methods to control grasping actions given a detection of an object using the compositional object model", fit exactly the breakdown structure suggested by the Tasks reported in the "Description of Work". In particular, the description of the activities to be performed in Tasks 4.1, 4.2 and 4.3, represents the sequential temporal stages that one could envision in performing a robust grasp when adopting a fully actuated and sensorized robotic hand. In fact, the pre-shaping of the hand to a first object contact (Task 4.1), the assessment of the first-order properties of the object surface, along with the local optimization of the contact point locations (Task 4.2), and the actual grasp acquisition with the optimal distribution of contact forces (Task 4.3), indeed assume that the action is performed by employing a dexterous hand with a plethora of force and position sensors. Moreover, here the robustness of the grasp was envisioned by reasoning, at each stage of the process, whether the previous step was successfully performed and taking proper corrective actions if that was not the case.

The approach that we came to pursue sees instead a central role of compliantly underactuated hands, where uncertainty in the relative pose between the hand and the object and object shape itself is mechanically absorbed by the adaptive behavior of the Pisa/IIT SoftHand. This new research direction brought about somewhat completely unplanned (and unexplored) issues that we considered central to the goals of WP4 and worth to be pursued. First of all, the sensorization of a compliantly adaptive hand

with dislocable joints asked for a whole new approach to provide proprioceptive feedback to the hand (this topic is actually matter of DR 3.1). The force sensorization problem, which is somewhat assumed in the Tasks 4.2 and 4.3, poses non trivial technological problems for the hand under investigation, and is still under development and testing. Moreover, the envisioned phases to transit from grasp formation to successful grasp acquisition are inextricably entangled and somehow blurred for an adaptive and underactuated hand with just one single Degree of Actuation, as the dynamic interaction between the hand and the object — and possibly the environment — is responsible for the global behavior of the system. This new perspective asked for a rescheduling and reorganization of the needed pieces of work.

For the reasons mentioned above, the work that is presented in this report is mostly focused on proposing robust grasping solution for an adaptive hand and, in particular, for our Pisa/IIT SoftHand. The learning and transfer of grasps within and across object categories, instead, makes use of the fully actuated and sensorized 20-DoF DLR-HIT Hand II, since a finer control over the fingertip position is required.

Sec. 1.2.1 presents our approach to solve high-level planning for grasping an object and passing it from one hand to another. This work constitutes the theoretical foundation for the high-level planning needed to perform the Task 5.3.

In Sec. **??**, we present our approach to define successful synthetic poses for the Pisa/IIT SoftHand in grasping the PaCMan Object Database [1]. In Sec. **??**, the previous approach is sharpened by employing a custom decomposition into cuboids of the object point cloud. Each cuboid suggests a number of grasps to be performed on the object. These grasps are used to warm start the search for successful grasps to be synthetically found in the simulation environment and to be attached to that object's grasp database. The feasibility of using this approach on-line to find graspable parts on objects in the scene with partial point cloud information and irrespective to their prior recognition is under investigation using the framework described in MS 4.2, Sec. A.1-A.4.

In Sec. **??**, we shift our attention onto the generalization and adaptation ability of the previously reported grasp planning algorithm for fully actuated hands showing, in experiments using the DLR-HIT Hand II, how it is possible to just use a few example grasps, taught in a kinesthetic manner, in order to be able to transfer this knowledge to unknown, even partially visible objects.

In Sec. **??**, we consider the grasps of the previous section as the starting point, but now wish to execute them more reliably than in an essentially open loop fashion. We present the progress on our method for planning tactile information gathering to reduce the uncertainty in the pose of an object. The method works by embedding the information to be gained in the physical space, using it to shorten distances, and thus make information

gathering trajectories of lower cost. We show improvement in our results over the previous year.

Sec. **??** presents a new approach to planning for soft robot manipulators under task constraints. This piece of work is central to motion planning in dual arm configurations, e.g. when an object is held simultaneously with two compliant underactuated hands. The corresponding control problem, which makes use of a geometric formulation and is based on a non-interacting scheme, is tested on a dual-arm (closed kinematic loop) configuration.

Sec. **??** presents our approach to solve high-level planning for grasping an object and passing it from one hand to another. This work constitutes the theoretical foundation for the high-level planning needed to perform the Task 5.3.

In Sec. 1.2.3, we describe a planning strategy based on direct trajectory optimization that does not require a-priori specification of the contact sequence and presents the benefits of providing dynamically consistent plans, from the outset, and allowing for opportunistic exploitation of environmental constraints.

1.2.1 On the Problem of Moving Objects with Autonomous Robots: a Unifying High-Level Planning Approach

In order to successfully complete Task 5.3, which requires the object to be passed between the hands of the robot, we describe in this section our approach to solve high-level planning for this matter.

Our idea considers the possibility of having the object in a known position, which can be recognized using vision, and a higher-level agent (such as a user) decides a target configuration the object has to reach.

From this information, a semantic graph is constructed which has as nodes a grasp id and a workspace id, and as arcs all possible known transitions between nodes.

Once a minimum cost path has been found, it is translated back into Cartesian information for collision-free motion planning, grasp commands, and all low-level requirements to execute the plan successfully.

While there are many previous works on combining high-level semantic reasoning and low-level cartesian path planning (see e.g. [2, 3, 4]) which mostly take advantage of object-specific reasoning introduced in [5], our main contribution is to include the possibility of passing an object between two end-effectors.

Specifically, considering end-effectors with different properties, the table (and possibly other non-movable surfaces) has its own set of grasps for an object, and is treated exactly like a hand when generating arcs in the graph, apart from the fact that it cannot be moved and, thus, there will be no arc connecting the same "table grasp" in adjacent workspaces (while there could be one if the grasp is performed via a hand of the robot).

Thus, considering that previous approaches mostly rely on a table in order to move an object from one arm workspace to another, we can simply classify the *primitives* which allow a transition from a grasp/workspace pair into another in three categories:

1. *pick* with an end-effector from a fixed surface
2. *move* the end-effector
3. *place* with an end-effector onto a fixed surface

Our approach uses instead a more general action of "grasp transfer", which can be specialized in:

1. moving actions of an end-effector among its reachable workspaces
2. pick and place actions if one end-effector involved is non-movable (such as a table)
3. a grasp-ungrasp sequence if both end-effectors involved are movable

This list is still under development, and we will try to generalize it even more in the future with other, more interesting *primitives* which could possibly exploit dynamics, friction, gravity, and so on.

A more detailed view on this method and on the achieved results can be seen in Sec. **??**.

1.2.2 Learning and Inference of Dexterous Grasps for Novel Objects with Underactuated Hands

In this section we show the extension of our approach to ...

1.2.3 A Computational Framework for Environment-Aware Robotic Manipulation Planning

Moving beyond with respect to what we committed to investigate in Tasks 4.2 and 4.3, we introduce, in this section, a computational framework for direct trajectory optimization of general manipulation systems without *a-priori* specified contact sequences, possibly exploiting *environmental constraints* as a tool to accomplish a task.

Originally, we planned to approach the problem of robust grasping by dividing it into three main stages: (i) move from hand pre-shape to first object contact (Task 4.1), (ii) perform an incipient grasp and assess the first-order properties of the surface of the object, possibly changing candidate contact locations so that the quality of the incipient grasp is maximized (Task 4.2), and (iii) perform the actual grasp acquisition by optimizing the distribution of the applied contact forces (Task 4.3).

In this section, we report on our efforts to devise a framework to be employed not only for grasp planning, but also for manipulation planning, that should be able to merge all three previous phases in a systematic and coherent way. The user should be focused on providing high-level objectives, e.g. "move object A from pose 1 to pose 2 in a given amount of time", and the framework should be able to provide a manipulation plan that should take care of all the rest, e.g. should specify low-level actions and their correct sequence for the manipulation system at hand (single-arm configuration, dual-arm configuration), defining the whole contact sequence (where and when to make and to break contacts), should provide trajectories consistent with the dynamics of the manipulation system, unilateral contacts, friction constraints and actuation limits.

Related work to which we compare ours indeed include those using: (i) *traditional grasp planners*, such as [6] and [7]; (ii) *general purpose planning algorithms*, such as [8] and [9]; (iii) *machine learning approaches*, as [10] and [11]; (iv) *optimization-based trajectory planners*, such as [12], [13], and [14]. In particular, with respect to [14], which is the closest to ours, we can affirm that, besides a much more efficient computational pipeline, by explicitly instantiating environmental contact forces, we fabricate a strategy such that, if the task may be more efficiently and/or more robustly performed with the aid of constraints provided by the environment, the proposed approach can devise manipulation strategies that cleverly and opportunistically exploit environmental constraints.

Moreover, two approaches are presented to model the dynamics of systems with intermittent contacts: the first one, in which continuous contact reaction forces are generated by nonlinear virtual springs, is convenient to tackle scenarios where we try to avoid *sliding* contacts; the second one, which is based on a velocity-based time stepping scheme, is suitable in scenarios where *sliding* interaction primitives may lead to convenient interactions among the hand, the manipulandum and the environment.

In both cases, beside system's state and applied torques, 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, sparsity patterns in the linear algebra expressions generated during the solution of the optimization problem are exploited, and Algorithmic Differentiation (also known as Automatic Differentiation) is leveraged to calculate derivatives. These aspects appear completely unexplored in the literature of high-level planning for systems with intermittent contacts.

The approach is evaluated in three simulated planar manipulation tasks: (i) moving a circular object from an initial pose to a final pose in the workspace with two independent fingers, (ii) rotating a capsule-shaped object 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 environmental constraints. Task (iii) demonstrates that even dexterous fingertip gaits can be obtained as a special solution in the very same framework.

More details on this new approach and on the achieved results can be found in Sec. A.2.

1.2.4 Offset-free MPC Explained: novelties, subtleties, and applications

In this section we show ...

1.2.5 Infrared Sensor-based Grasp Planning

In this section we show ...

References

[1] PaCMan Object Datasets. [Online]. Available: http://www.pacman-project.eu/datasets/

[2] L. Karlsson, J. Bidot, F. Lagriffoul, A. Saffiotti, U. Hillenbrand, and F. Schmidt, "Combining task and path planning for a humanoid two-arm robotic system." Citeseer, 2012.

[3] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on.* IEEE, 2012, pp. 429–435.

[4] D. Leidner and C. Borst, "Hybrid reasoning for mobile manipulation based on object knowledge," in *Workshop on AI-based Robotics at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[5] L. Levison, "Connecting planning and acting via object-specific reasoning," Ph.D. dissertation, Citeseer, 1996.

[6] C. Rosales, "Grasp planning under task-specific contact constraints," Ph.D. dissertation, Universitat Politècnica de Catalunya, 2013.

[7] A. Miller and P. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics and Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[8] Y. Koga and J.-C. Latombe, "On multi-arm manipulation planning," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1994.

[9] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Robotics: Science and Systems (RSS)*, 2014.

[10] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Neural Information Processing Systems (NIPS)*, 2014.

[11] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," 2015, under review.

[12] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Eurographics/ACM Symposium on Computer Animation*, 2012.

[13] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

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

A Annexes

A.1 Article: On the Problem of Moving Objects with Autonomous Robots: a Unifying High-Level Planning Approach

Authors H. Marino, M. Ferrati, A. Settimi, C. Rosales, M. Gabiccini

Info Published in: Robotics and Automation Letters, IEEE. Date of Publication: January 18 2016. ISSN: 2377-3766;
DOI: 10.1109/LRA.2016.2519149

Abstract Moving objects with autonomous robots is a wide topic that includes single-arm pick-and-place tasks, object regrasping, object passing between two or more arms in the air or using support surfaces such as tables and similar. Each task has been extensively studied and many planning solutions are already present in the literature. In this paper we present a planning scheme which, based on the use of pre-defined elementary manipulation skills, aims to unify solutions which are usually obtained by means of different planning strategies rooted on hard-coded behaviors. Both robotic manipulators and environment fixed support surfaces are treated as end-effectors of movable and non-movable types, respectively. The task of the robot can thus be broken down into elementary building blocks, which are end-effector manipulation skills, that are then planned at the kinematic level. Feasibility is ensured by propagating unforeseen low-level failures at the higher level and by synthesizing different behaviors. The validity of the proposed solution is shown via experiments on a bimanual robot setup and in simulations involving a more complex setup similar to an assembly line.

Relation with the deliverable this work is concerned with the problem of planning a grasp sequence to move one object from an initial configuration to a final one. Since the initial configuration is not known a priori, but has to be estimated using vision, nor the sequence of handoffs is pre-specified (but automatically synthesized), the present work contributes to the goals of Task 4.4.

Attachment (following pages until next annex)

On the Problem of Moving Objects with Autonomous Robots: a Unifying High-Level Planning Approach

Hamal Marino[1,2], Mirko Ferrati[1], Alessandro Settimi[1,2], Carlos Rosales[1], and Marco Gabiccini[1,2,3]

***Abstract*—Moving objects with autonomous robots is a wide topic that includes single-arm pick-and-place tasks, object regrasping, object passing between two or more arms in the air or using support surfaces such as tables and similar. Each task has been extensively studied and many planning solutions are already present in the literature.**
In this paper we present a planning scheme which, based on the use of pre-defined elementary manipulation skills, aims to unify solutions which are usually obtained by means of different planning strategies rooted on hard-coded behaviors. Both robotic manipulators and environment fixed support surfaces are treated as end-effectors of movable and non-movable types, respectively. The task of the robot can thus be broken down into elementary building blocks, which are end-effector manipulation skills, that are then planned at the kinematic level. Feasibility is ensured by propagating unforeseen low-level failures at the higher level and by synthesizing different behaviors. The validity of the proposed solution is shown via experiments on a bimanual robot setup and in simulations involving a more complex setup similar to an assembly line.

***Index Terms*—Manipulation Planning; Cooperative Manipulators; Dual Arm Manipulation**

I. Introduction

INDUSTRIAL and service robots are often used to perform object moving tasks, using different strategies depending on the specific application. Behaviors shaped on the objects characteristics, typical of an industrial environment, are not suitable for a generic approach. As a matter of fact, exploiting all the robot capabilities like using one or more arms, passing the object between them, or using the environment to place the object and re-grasp it, is necessary to adapt to the large variety of possible situations.

Our work focuses on multi-robot coordinated manipulation as defined in [1]. Multiple end-effectors may interact with the same object during the task, either subsequently or at the same time. In this paper, a novel approach to perform object moving that can handle multi-arm robots is reported.

Manuscript received: August, 30, 2015; Revised November, 2, 2015; Accepted January, 6, 2016.
This paper was recommended for publication by Editor Antonio Bicchi upon evaluation of the Associate Editor and Reviewers' comments. This work is supported by the European commission projects PaCMan EU FP7-ICT 600918 and by the grant no. 645599 "SoMa" -Soft-bodied intelligence for Manipulation- within the H2020-ICT-2014-1 program.
[1] Centro di Ricerca "E. Piaggio", University of Pisa, 56122 Pisa, Italy.
Corr. auth.: hamal.marino@centropiaggio.unipi.it
[2] Dep. of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova
[3] DICI, University of Pisa, 56122 Pisa, Italy
Digital Object Identifier (DOI): see top of this page.

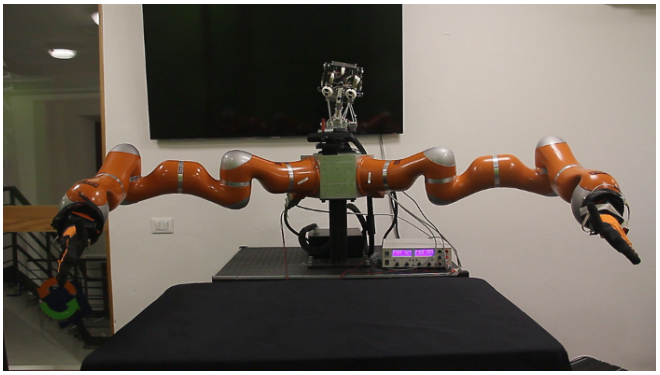

Fig. 1: Vito, the robot used to perform the reported experiments. Two Kuka Light-Weigth Robots (LWR) are mounted on a fixed torso and are equipped with a left and a right Pisa-IIT Soft-Hands. An Asus Xtion Pro-Live is used for vision.

A vision software recognizes the object present in the scene, then an operator selects the final goal for the object through a user interface. A high-level plan is generated on a suitable graph representation built using information stored in a database (DB), and it is then translated into a Cartesian plan that will be executed by the robot. The planning phase is capable of finding different strategies (pick-and-place with one or multiple arms, handoff, regrasp) and, exploiting the environment (e.g. using support surfaces), to move the object from the starting to the final position.
It is worth remarking that we do not use any *a priori* fixed strategies: instead, the strategy emerges from scratch as the outcome of the high level planning phase.

Manipulating objects usually involves a series of action in which objects are recognized, grasped, moved, and released. The simplest robot configuration that can perform these actions is a single arm in a single support surface setup, where an object is grasped, moved, and released in a different position. In such single arm setups, grasp planning, object recognition in a cluttered environment, optimization of the low-level planning, and optimization of the order of objects displacements have been the focus of recent research: [2], [3], [4], [5]. As described in [6], the exploration of a graph whose nodes represent a combination of grasp type and object position produces a sequence of multiple grasp-move-release actions, so that the same end-effector can move an object in a constrained environment, where a single pick-and-place strategy would fail.

Dual arm robot platforms gained an increasing interest because of their larger workspace and number of DoFs, but also

because of their similarity with humans, thus their capability of replacing them in already established assembly lines.
These platforms are capable of performing handoff-based manipulations: a sequence of grasp, move, handoff to the second arm, move, and release.
Planning algorithms that use handoff can be found in [7], [8], or even devising the grasp trajectories online in [9], [10]. An alternative to handoff is using a common support surface for passing an object: this approach can be interpreted as a connection between two single arms pick-move-release strategies, but it also poses some additional constraints on the kinematics and the collisions between the robots, the object and the surface. Approaches mainly focused on solving these motion planning issues are [11], [12], [13], [14].

As more robots or objects are added, a coordinated planning becomes the main requirement for such platforms.
An effort to handle this increased complexity is the integration of a semantic/symbolic reasoning into a manipulation planning, in order to perform a single task, such as making a pancake [15], moving cylinders on a cluttered table [16], solving a Tower of Hanoi [17].

Other works like [18], [19] focus mostly on reasoning, using formal languages, to plan a task by using spatial and temporal knowledge, assuming that the motion planning and grasping are handled in a perfect way.

A two-level approach similar to ours has already been presented in [20] and more recently extended in [21]: however, even if the approach in [21] may cope with the specific problem in our context, most of the experimental results reported there in manipulation do not include grasping capabilities of the robot (the robot can only push the object on a table). In order to include such capabilities, some challenges regarding the continuous modelling of grasping would arise, which we decided to handle by discretizing object grasps and, similarly, object passing poses; moreover, exploitation of support surfaces would require an adequate formalism in order to find feasible poses for an object lying on a support surface. We believe that the ability to cope with full grasping sequences with an anthropomorphic underactuated robotic hand and the backtracking ability of our method represent non trivial features of our work with respect to existing approaches.

Since our algorithm's main inputs are the starting and desired positions of an object, an autonomous procedure can be considered to provide such inputs instead of a human operator. For example, [2] and [18] deal with reordering a set of objects in order to access one of them obstructed by others, automatically resulting in a sequence of single object pick-move-place actions equivalent to multiple selections of target by the user in our framework.

Our main contribution with this paper is a graph-based modelization of the problem, associated to a novel planning algorithm, that effectively unifies previously described strategies. In this graph, each node represents the *state* of the object to be moved, while the arcs correspond to feasible *change of state* (more details on this are given in Sec. IV). In a hierarchical planning layered architecture, our planner needs to be placed below a symbolic reasoning planner as the ones cited above. The experimental validation of the proposed approach has been carried out on the Vito robot of Centro di Ricerca "E. Piaggio" (see Fig. 1). In order to test the planner on a more complex setup with multiple end-effectors, a simulation with five Kuka LWR and one conveyor belt has been also performed.

II. Problem Statement

The problem we are tackling deals with providing a robot the ability to autonomously answer the request: "move that object from the initial pose to a desired target pose", given any number of end-effectors (e.e.) and support surfaces.

Instead of pre-specifying a particular strategy to achieve this result, we only build a set of correspondences between end-effectors, objects, and grasps. Given this set, a task planning is executed that results in a feasible sequence of actions (*move*, *grasp*, and *ungrasp*) in Cartesian space to be performed in order to reach the goal.
A sample-based, low-level planner then finds collision-free joint space trajectories for the robot in order to obtain the desired motions.

In case of low-level planning failures, a recover procedure takes place in order to activate a new task planning, trying to find a different feasible sequence of actions.

Our working assumptions are that:

- a vision system provides all necessary information about the current state of the environment to the planner, handling object recognition and pose estimation with sufficient accuracy[1];
- all objects appearing in the scene can either be grasped or considered as fixed obstacles;
- a grasp database, to be provided to the algorithm, encodes information related on how each object can be grasped by each end-effector;
- information about the desired target configuration is provided by the user. Specifically, the final position and orientation of the object in Cartesian space and the desired final end-effector supporting it (any robot e.e. or support surface) have to be specified using the GUI (see Fig. 2).

III. Definitions and Database Generation

To perform the high-level task planning, the devised algorithm relies on fundamental information stored in a database.
We will now define the concepts used by the planner and how the information related to those concepts is generated and stored in the database. A very simple, exemplary database is shown in Fig. 3. We store geometric information about the support surfaces, approximations of each arm reachable region, the grasps associated between an object and an end-effector, and the feasible grasps that can be used simultaneously by two end-effectors during an hand-off.
In particular, grasp information about the end-effector poses

[1]We will not describe the pose estimation vision system here, as the implementation can be found in [22].

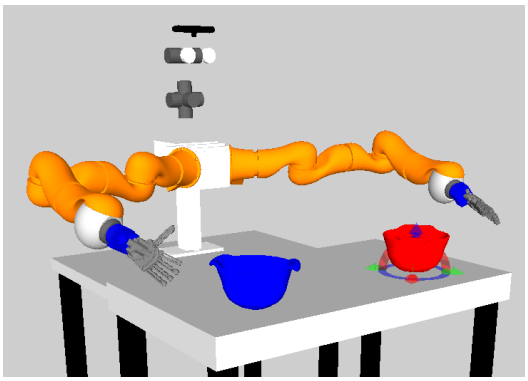

Fig. 2: RViz visualization. The blue object (left) represents the initial object configuration given by the vision system; the red object (right) represents the goal configuration selected by the user. For a picture of the whole User Interface see attached multimedia file.

w.r.t. the objects can be generated automatically for fully actuated hands (as in e.g. [23], [24]) or manually, like we did, using a motion capture system.

Objects		
Obj_id	3D views	Name
o1	PointClouds	Cylinder

End-Effectors			
EE_id	Reachable Workspaces	Movable	Name
e1	w1,w2,w3	FALSE	Table
e2	w1,w2	TRUE	LeftArm
e3	w2,w3	TRUE	RightArm

Workspaces			
Workspace_id	Adjacency	Geometry	Name
w1	w2	box1	Left
w2	w1,w3	box2	Middle
w3	w2	box3	Right

Grasp_id	EE_id	Name	Allowed_transitions
g1	e1	TopTable	g5 , g6 , g8 , g9
g2	e1	SideTable	g5 , g8
g3	e1	BottomTable	g4 , g5 , g7 , g8
g4	e2	TopLeftArm	g3, g8 , g9
g5	e2	SideLeftArm	g1 , g2 , g3 , g7, g9
g6	e2	BottomLeftArm	g1 , g7 , g8
g7	e3	TopRightArm	g3 , g5 , g6
g8	e3	SideRightArm	g1 , g2 , g3 , g4 , g6
g9	e3	BottomRightArm	g1 , g4 , g5

Fig. 3: Example of a database overall structure. Objects, End-Effectors, Workspaces, and Grasps tables are reported.

A. Objects

All (known) objects in a given set are included in the database; for each of them, we store multiple 3D views that are used for object recognition and pose estimation using a 3D vision library ([22]) to gather the necessary start state information before the planning starts.

B. End-Effectors

An *end-effector* is an entity which can act/apply a grasp/support on an object: there are both "movable" end-effectors (such as robot hands connected to arms) and "non-movable" ones (such as a fixed surface in the environment which can be exploited for statically stable object support). From a task planning perspective, there is no distinction between movable and non-movable end-effectors, as both types can interact with an object in specific ways, and can exchange the object with specific interaction transitions.

C. Workspaces

Our *environment* is the union of all the points reachable by all the end-effectors (movable and non-movable). For fixed surfaces (which are non-movable end-effectors), reachable points are represented by thin normal extrusions of the surfaces themselves.

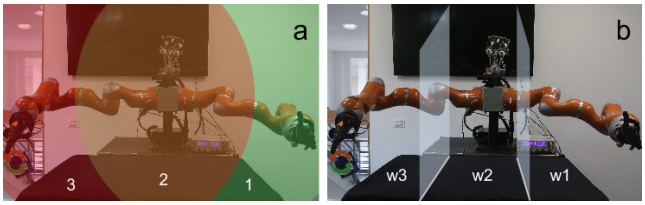

Fig. 4: (a) The reachable regions of the left and right arm are respectively reported in green and red, while the one of the table is represented by the table itself. (b) Using the intersection between simple approximations of these regions (cuboids), we defined for this scenario *w1*, *w2*, and *w3*.

In order to split the environment into regions useful for the high level planning, we approximate the reachable region of each end-effector with simpler convex models. We use cuboids because they are a natural 3D extension of support surfaces. We then check for intersections in order to find regions where multiple end-effectors can interact with each other. These intersections are called *workspaces*.

Workspaces are candidates for object *passing*, since they may be reached by more than one end effector simultaneously. Workspaces enjoy the additional property of *adjacency* if they can be reached by the same movable end-effector. Thus, adjacent ones are candidates for object *moving*.

With reference to Fig. 3 (tables) and Fig. 4(b), since workspaces *w1* (left workspace) and *w2* (middle workspace) are adjacent (check Workspaces table, rows 1 and 2), and also reachable by the movable e.e. *e2* (left arm), (check End-Effectors table, row 2), object *o1* (Cylinder) can be moved between these two workspaces. This allows to affirm that the Cylinder can be moved between the left and the middle workspace using the left arm with the grasps from *g4* to *g6* (check Grasps table, rows with *e2* in EE_id column).

Reachability information between end-effectors and workspaces is stored in End-Effectors table.

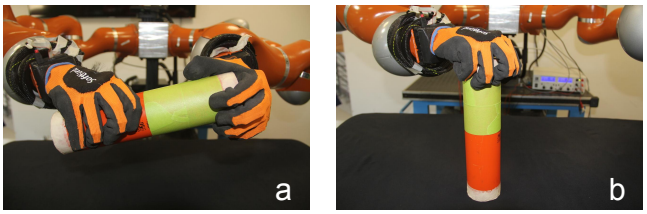

Fig. 5: (a) transition between Top-LeftArm and Side-RightArm grasps (g4-g8). (b) transition between Bottom-Table and Top-RightArm grasps (g3-g7).

D. Grasps

In this work, a *grasp* is defined as the relative configuration of the end-effector and a specific object once it is grasped. This Cartesian information is not shown in Fig. 3 for better readability. We associate to each object of our database one or more grasps.

If an object has to be handed off from an end-effector to another, there will be an instant in time where both e.e. are grasping the object.
A *transition* between two grasps is an interaction primitive that has no kinematic overlap between the two different end-effectors performing the grasps at the same time, i.e. there is no collision between the e.e.'s when they both are in their final grasp configuration with respect to the object.

The transitions we consider in this work are used by the high-level planner without any further distinction, but we can interpret them in the following way:

- pick and place actions when a non-movable end-effector is involved together with a movable one,
- a sequence of grasp-ungrasp actions when both end-effectors are movable.

No transition is allowed between any two non-movable end-effectors.

As an example, with reference to Fig. 3, End-Effectors table, *e1* (Table) and *e3* (RightArm) can both reach *w2* (Middle workspace) (check End-Effectors table). *e1* can support *o1* (Cylinder) with *g3* (Bottom-Table grasp), while *e3* can grasp *o1* with *g7* (Top-RightArm). Since a transition between *g3* and *g7* exists (check Grasps table, row 3), the cylinder can be safely grasped with the right arm from the table, as shown in Fig. 5, right panel.

The Cartesian product between reachable workspaces, adjacency, and grasp transitions, allows to create a graph as the one depicted in Fig. 6.

IV. Algorithm Description

An object is in a certain state $S_{e,g,w}$ when: (i) it is being grasped by a particular end-effector e, with (ii) a specific grasp g, and (iii) it is inside a specific workspace w. Moving an object means making a transition from its current state to a different one. In this work, a high-level plan is a sequence of states and transitions.

We show in Algorithm 1 the complete proposed algorithm, denoting with:

- C_{Init}, C_{Current}, and C_{Final} the initial, current and final object configurations in Cartesian space,
- G the graph,
- P_C the Cartesian plan,
- $S_{\text{Init}} = S_{e_i,g_i,w_i}$ and $S_{\text{Final}} = S_{e_t,g_t,w_t}$ the initial and final object configurations in high-level space,
- P_{HL} the high-level plan,
- Arc_{Id} the id of the arc that makes the planning fail,
- p_i a single item of the Cartesian plan,
- M a joint space motion plan.

Algorithm 1: MoveObject

```
(states,transitions) = GetInformationFromDB();
G= GenerateGraph(states,transitions);
C_Current= GetInitialStateFromVision();
C_Final= GetFinalStateFromUser();
repeat
    G_HL= G;
    C_Init= C_Current;
    S_Init= ConvertCartesianToHighLevel(C_Init);
    S_Final= ConvertCartesianToHighLevel(C_Final);
    repeat
        P_HL= ComputeShortestPath(G_HL,S_Init,S_Final);
        if P_HL is empty then
            GlobalFailure;
        end
        P_C= ConvertHighLevelToCartesian(P_HL);
        if P_C is empty then
            Arc_Id= GetFailedConversion();
            G_HL= Backtracking(G_HL,Arc_Id);
        end
    until P_C is not empty;
    /* Executing the cartesian plan    */
    for p_i ∈ P_C do
        M=MotionPlanning(p_i);
        if M is valid then
            Execute(M);
            C_Current= CurrentObjectPosition();
        else if C_Current== C_Init then
            GlobalFailure;
        else
            break for;
        end
    end
until C_Final== C_Current;
```

A. Graph Generation

Using the full DB structure as illustrated in Sec. III, a graph G is generated by intersecting transitions between grasps, reachability from end-effectors to workspaces and workspaces adjacency information. For the exemplary database content illustrated in Fig. 3, the generated graph is depicted in Fig. 6. Note that this is a simplified example: a real experiment graph could have hundreds of nodes and thousands of arcs, as reported in TABLE I. For a picture of a real generated

graph see attached multimedia file.
Each node of the graph represents a state $S_{e,g,w}$ of the object. Each arc represents one of the two different kinds of transition (akin to "transit" and "transfer" concepts of [6]):

1) from S_{e,g,w_i} to S_{e,g,w_j}: change of workspace with the same grasp (thus the same end-effector), where e must be movable and w_i and w_j must be adjacent (in the sense of Sec. III-C) and reachable by e. This is represented by the dashed lines in Fig. 6.
2) from $S_{e_i,g_k,w}$ to $S_{e_j,g_l,w}$: change of end effector inside a certain workspace, where a transition between grasp g_k and g_l must exist (as described in Sec. III-D). This is represented by the solid lines in Fig. 6.

The weight of any arc could be chosen based on the failure probability of the associated transition or on the stability of the grasps involved. In this work we make the simple consideration that a change of end effector (transition from $S_{e_i,g_k,w}$ to $S_{e_j,g_l,w}$) has a higher failure rate w.r.t. a change of workspace (transition from S_{e,g,w_i} to S_{e,g,w_j}), thus we use weights with different order of magnitude, e.g. 1.0 and 0.1, respectively.

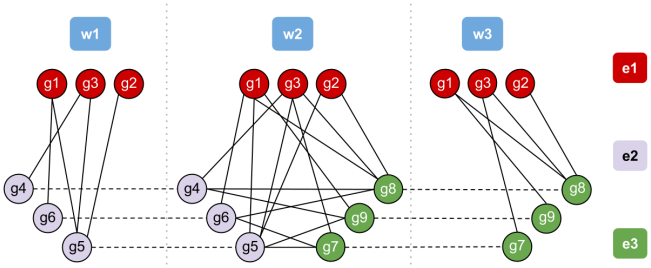

Fig. 6: Graph generated from the exemplary DB in Fig. 3: each node represents a state of the object being grasped by the end effector e_i ($i = 1, \ldots, 3$) with the grasp g_k ($k = 1, \ldots, 9$), in workspace w_m ($m = 1, \ldots, 3$). Solid lines represent grasp transitions within the same workspace; dashed lines represent object displacements between workspaces with the same grasp. Simultaneous changes in workspace and grasp being applied to the object are not allowed in the current framework.

Note that the built graph no longer distinguishes between the type of end effectors or transitions.

B. Cartesian-to-High-Level Pose Conversion

In order to provide the initial and final states $S_{\text{Init}} = S_{e_i,g_i,w_i}$ and $S_{\text{Final}} = S_{e_t,g_t,w_t}$ our algorithm uses the information gathered from the vision system and the input by the user, respectively.
Workspaces w_i and w_t are found by using the database geometry, searching for those containing the required initial and target poses (note that workspaces do not overlap because they are the cells of a grid).
Initial end-effector e_i is assumed to be known, while target end-effector e_t is assumed to be known and non-movable in order to avoid ambiguous situations where multiple end effectors could simultaneously hold the object in the target position (see Fig. 7).

Finally, a search for grasps (g_i, g_t) that fulfill the initial and final pose is performed; this involves searching through all possible grasps associated to the end-effectors (e_i, e_t), which are checked for similarity with respect to the initial and target poses of the object.

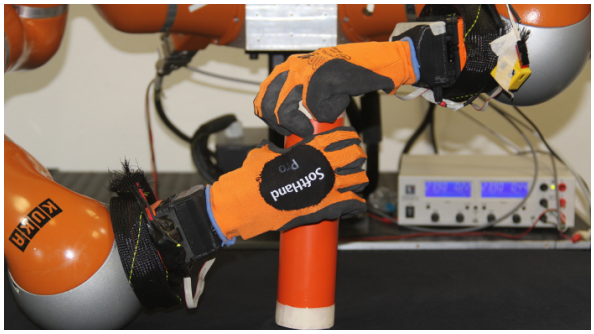

Fig. 7: The object in its final position can be grasped by three different end-effector (table, left hand, right hand)

C. Computation of the shortest path

We want to find a sequence of states from S_{Init} to S_{Final}, while minimizing the number of grasp transitions between end-effectors. To this sake any shortest path algorithm can be used: we choose Dijkstra [25] as one of the most widely known and easiest to implement.
In Fig. 8 we show the solution (path) for given initial and final states, which uses a direct handoff between the two end-effectors, while in Fig. 9 we show a plan with different initial and target states that automatically uses the table as intermediate end-effector in order to minimize path length. Each path represents a set of workspace/grasp transitions, which need to be converted into a discrete sequence of Cartesian position of the object and its grasping end-effector.

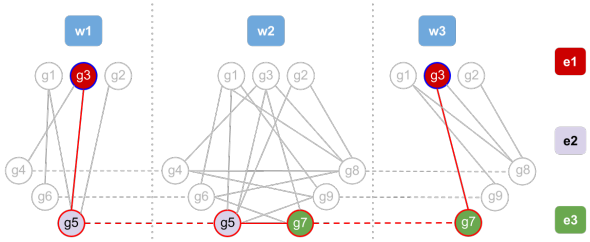

Fig. 8: Planning given by Dijkstra, the transition on the table is avoided because it would require more grasp transitions for the task.

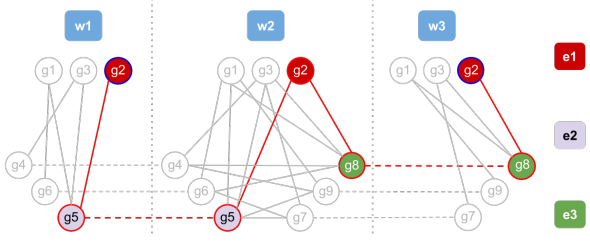

Fig. 9: Planning given by Dijkstra, the transition on the table is used because it is the shortest one for these initial and final states.

D. High-Level-to-Cartesian Conversion

The aforementioned conversion from the high-level plan P_{HL} to a Cartesian sequence P_C is a critical aspect of our approach, since it allows the decoupling between high-level object passing/moving and low-level motion planning. Each arc in the high-level plan is associated to low-level commands: in particular, a change of workspace generates a *move* command for an end-effector, while a change of grasp generates a pair of *grasp-ungrasp* commands for the end-effectors involved.

The discrete high-level plan is first simplified by removing unnecessary intermediate transitions, so that an end-effector $\hat{e}$ moving an object through multiple workspaces $S_{\hat{e},g,w_1}...S_{\hat{e},g,w_n}$ generates a single *move* command for the end effector to reach the final workspace w_n.
After reaching that workspace, the object is either in its final target position, or needs to be passed to another end-effector by a sequence of *grasp-ungrasp* commands. In order to find a valid position for the object to be grasped by the next end-effector while still being grasped by the previous one, the following procedure has been applied.

- Create a virtual serial chain connecting the initial and final end-effectors together as if they were rigidly attached to each other by a constant $SE(3)$ matrix M. This represents the poses the end-effectors have to maintain in order to exchange the object.
- Consider as a virtual base-link the base-link of the initial arm, and as a virtual end-effector the base-link of the final arm.
- Solve the IK for the virtual chain with any jacobian-based approach avoiding self-collisions and collisions with environment. The target of this IK corresponds to the base-link of the final arm: in this way the closed-loop kinematic chain given by the two-arms connected through the object is respected.

E. Backtracking procedure

In case the conversion between high-level and Cartesian plans fails due to the inability of finding a collision-free configuration for performing a particular transition, a backtracking procedure is performed; this removes the arc associated to the failing transition and goes back to computing a shortest path on the updated graph.
The loop between phases in sub-sections IV-C, IV-D and backtracking is iterated until no backtracking is necessary (i.e. all states and transitions of a high-level plan are found to be feasible) or no valid high-level plan is found, in which case the target state is not kinematically reachable from the initial state with a collision free set of transitions.

F. Execution of the Plan

The execution of the *move*, *grasp*, *ungrasp* commands requires the use of a low level motion planning. If more than a single end-effector *move* command has to be performed, they are performed simultaneously, computing collision-free trajectories that involve the various end-effectors. This can be efficiently done using, e.g. a sample-based random planner.
In our implementation we used RRT* [26] with RRT-Connect [27] as a backup, both implemented in MoveIt! [28].
Even though the initial and final joint configurations have been tested to be collision-free, a collision-free motion plan is not granted to be found. In this unlikely case, the current object position is considered: if it is different from the starting one, a new high-level action plan is initialized with the current object position as the new starting one; otherwise, the algorithm ends with a failure to avoid an infinite loop.

V. Experimental Results

In this section the experiments to validate our approach are reported. Experiments have been performed without specifying any preferred strategy. The algorithm elaborates a plan and, depending on the situation, it produces a strategy with either a single or dual arm, with or without handoffs. We will now describe the different emerging behaviors obtained during the experiments. A video of these experiments is available in [29].

A. Experiments

Moving an object using handoff: In the reported examples (see Fig. 10) the robot uses two movable end effectors to move the object to the target position: this behavior is preferred since less e.e. switches are needed to perform it.

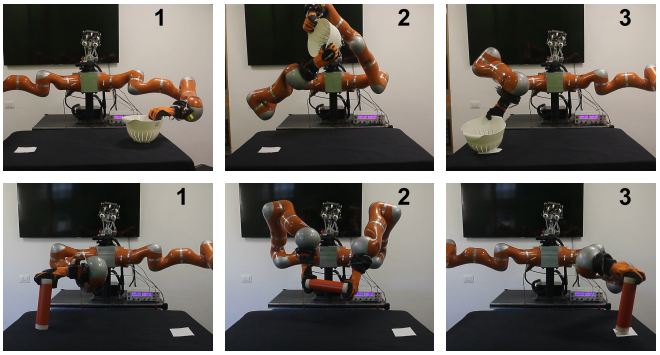

Fig. 10: Handoff experiments with a colander (on the top) and a cylinder (on the bottom).

Using support surfaces: Support surfaces are exploited in the case one single handoff is not possible. In this way the robot can re-grasp the object and then achieve the goal. In Fig. 11 single arm regrasping are reported, while in Fig. 12 the robot uses both arms to perform the task.

Pick-and-place: In the experiments reported in Fig. 13 the robot can execute the moving task using only one arm without any re-grasping. The particular case of trashing an object is shown in Fig. 14: the trash bin is being considered as a non-movable e.e. as well.

Assembly Line Simulation: To validate the generality and scalability of our approach with respect to the number and type of robots, we simulated kinematically the scenario depicted in Fig. 15. Here, five Kuka LWR arms are placed in front

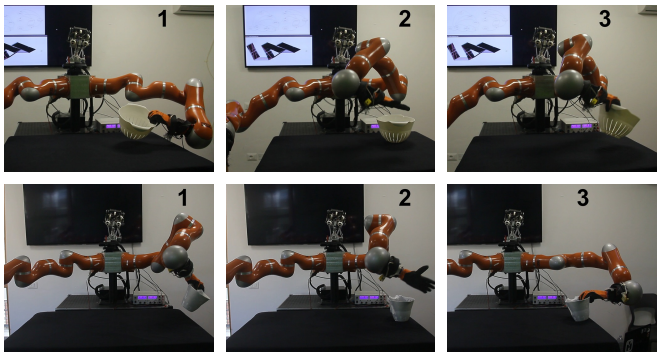

Fig. 11: Single arm object regrasping with a colander (on the top) and a pitcher (on the bottom).

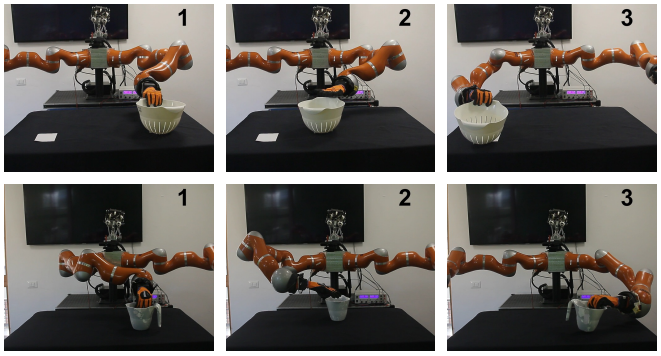

Fig. 12: Dual arm object regrasping with a colander (on the top) and a pitcher (on the bottom).

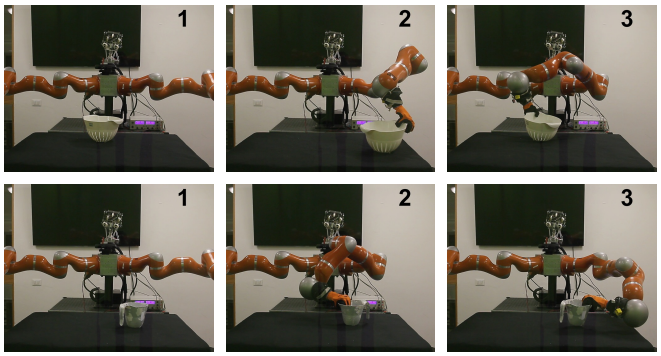

Fig. 13: Single arm pick-and-place experiments with a colander (on the top) and a pitcher (on the bottom). On the left the desired position is reported, then the initial and the final one are respectively in the middle and on the right.

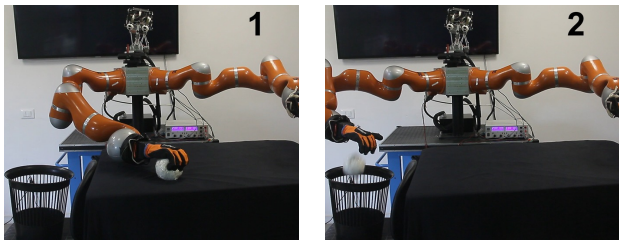

Fig. 14: Trashing a ball experiment.

of four workbenches, one of which contains a conveyor belt, which is modeled as a 1 DoF prismatic robot which can move horizontally objects that are placed on its surface. Using the procedure described in Sec. III-C a high-level description with 9 workspaces was obtained. The cylindrical object, initially resting on the first bench below a shelf (blue cylinder), has to find itself in the final configuration resting on top of the last bench (red cylinder). The outcome of the planning algorithm is a sequence consisting of the following actions: pick from the source location with first arm, handoff between first and second arm, place on the table, pick with third arm, handoff between third arm and fourth arm, handoff between fourth arm and conveyor belt, handoff between conveyor belt and fifth arm, place in the final location. Please refer also to the video attachment for more details.

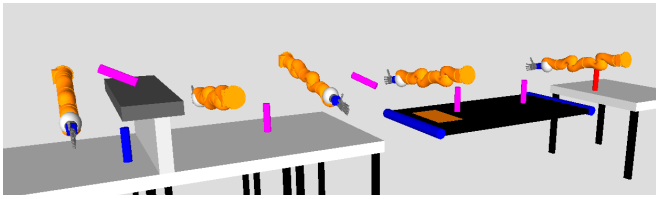

Fig. 15: An assembly line simulation with 5 Kuka LWR and a conveyor belt. The cylinder is moved from the initial position on the left (blue) to the final one on the right (red). The intermediate positions in which the object is exchanged between two e.e.'s are depicted in magenta.

B. Computational Costs

All experiments have been performed on a Intel Core i7-4770K machine with 16GB RAM using a single thread. The algorithm implementation could be highly parallelized thanks to its many parts based on randomized approaches, thus we expect a 3x speed-up to be a realistic value using 4 threads. Please consider this in comparison with other similar works.
In TABLE I we report computation times (in seconds) for the planning procedure described in Algorithm 1, both for the Vito robot and the assembly line scenarios, averaged on 100 trials with 5 different initial and target configurations for each task (table column).

	Vito Robot				Assembly Line
	Pitcher	Colander	Ball	Cylinder	Cylinder
GRAPH					
Nodes	108	144	80	224	736
Arcs	1609	3426	1056	5248	14016
HL-TIME					
avg	**3.2**	**5.4**	**1.9**	**2.6**	**8.5**
max	4.6	9.4	2.4	3.2	21.3
min	1.8	2.5	1.1	2.1	3.1
TOT-TIME					
avg	24.2	28.5	17.6	21.5	53.0

TABLE I: High level planning and total computation times (s).

While a high number of nodes and arcs should intuitively increase the computational time, indeed most of the CPU is used for the collision checking inside HighLevel_to_Cartesian conversion. Objects characterized by a simple geometry such as a ball or a cylinder are usually easier to handle and move, thus less arcs are tested in the loop of Algorithm 1.

The assembly line experiment requires more handoffs and surface supports (transitions), thus it is more likely to fail and cause a replan, as we can see from the greater max-min range. Obviously, the number of transitions depends on the strategy chosen by the high level planner, e.g. 2 for a Pick-and-Place and 3 for an handoff.
For each transition, the motion planning uses five seconds for

RRT* and, as a backup solution, ten attempts of RRT-Connect (up to a total of three more seconds). In our experience, and given the time allotted for the first phase, RRT* fails in about 40% of the transitions, mostly during planning approach trajectories to the object in a pose difficult for the robot to reach.

VI. Conclusions

In this work we proposed a high-level planner that unifies different object moving strategies such as pick-and-place and handoff, and exploits support surfaces if required to achieve the goal. Both a real and simulated examples with different objects and multiple manipulators show the efficacy and scalability of this approach.

Ongoing work is focused on extending the approach to planning for simultaneously moving more than a single object, on parallelizing planning and execution phases, and on considering more complex interactions between possibly multiple end-effectors and the object to be moved.

All the code of this framework is available at http://dualmanipulation.bitbucket.org.

Acknowledgments

Authors thank Gian Maria Gasparri and Federico Spinelli for fruitful discussions on the preliminary version of this work.

References

[1] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation a survey," *Robotics and Autonomous Systems*, vol. 60, no. 10, p. 1340, 2012.

[2] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, "Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 445–452.

[3] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," *Robotics: Science and Systems VII*, 2011.

[4] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation." in *ICAPS*, 2010, pp. 254–258.

[5] J. A. Bagnell, F. Cavalcanti, L. Cui, T. Galluzzo, M. Hebert, M. Kazemi, M. Klingensmith, J. Libby, T. Y. Liu, N. Pollard, *et al.*, "An integrated system for autonomous robotics manipulation," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 2955–2962.

[6] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004.

[7] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[8] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks," in *Intelligent Robots and Systems. IROS 2009. IEEE/RSJ International Conference on*, pp. 2464–2470.

[9] J.-P. Saut, M. Gharbi, J. Cortés, D. Sidobre, and T. Siméon, "Planning pick-and-place tasks with two-hand regrasping," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4528–4533.

[10] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Simultaneous grasp and motion planning: Humanoid robot armar-iii," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 2, pp. 43–57, 2012.

[11] L. Karlsson, J. Bidot, F. Lagriffoul, A. Saffiotti, U. Hillenbrand, and F. Schmidt, "Combining task and path planning for a humanoid two-arm robotic system," in *Proceedings of TAMPRA: Combining Task and Motion Planning for Real-World Applications (ICAPS workshop)*. Citeseer, 2012, pp. 13–20.

[12] B. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for dual-arm manipulation with upright orientation constraints," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3784–3790.

[13] F. Zacharias, D. Leidner, F. Schmidt, C. Borst, and G. Hirzinger, "Exploiting structure in two-armed manipulation tasks for humanoid robots," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5446–5452.

[14] B. Cohen, S. Chitta, and M. Likhachev, "Single-and dual-arm motion planning with heuristic search," *The International Journal of Robotics Research*, p. 0278364913507983, 2013.

[15] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth, "Robotic roommates making pancakes," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 2011, pp. 529–536.

[16] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 639–646.

[17] G. Havur, K. Haspalamutgil, C. Palaz, E. Erdem, and V. Patoglu, "A case study on the tower of hanoi challenge: Representation, reasoning and execution," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4552–4559.

[18] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1470–1477.

[19] L. Lindzey, R. A. Knepper, H. Choset, and S. S. Srinivasa, "The feasible transition graph: Encoding topology and manipulation constraints for multirobot push-planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 301–318.

[20] K. Hauser, T. Bretl, J. Latombe, and B. Wilcox, "Motion planning for a six-legged lunar robot," *Algorithmic Foundation of Robotics*, 2008.

[21] K. Hauser and V. Ng-Thow-Hing, "Randomized multi-modal motion planning for a humanoid robot manipulation task," *The International Journal of Robotics Research*, 2011.

[22] F. Spinelli, "Pose estimation library," https://bitbucket.org/Tabjones/pose-estimation-library.

[23] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics & Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110–122, 2004.

[24] D. Berenson and S. S. Srinivasa, "Grasp synthesis in cluttered environments for dexterous hands," in *Humanoid Robots. Humanoids 2008. 8th IEEE-RAS International Conference on*, pp. 189–196.

[25] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[26] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, p. 1478.

[27] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, pp. 995–1001.

[28] I. A. Sucan and S. Chitta, "Moveit!" http://moveit.ros.org.

[29] H. Marino, M. Ferrati, A. Settimi, C. Rosales, and M. Gabiccini, "RA-L submission accompanying video," https://youtu.be/vIHg6gzIemQ, Sept. 2015.

A.2 Article: A computational framework for environment-aware robotic manipulation planning

Authors M. Gabiccini, A. Artoni, G. Pannocchia, J. Gillis

Info Under review

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.

Relation with the deliverable grasp and manipulation planning for systems with a-priori unspecified contact sequences.

Attachment (following pages until next annex)

A Computational Framework for Environment-Aware Robotic Manipulation Planning

Marco Gabiccini Alessio Artoni Gabriele Pannocchia 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.**

I. 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 [1]. In Sec. III, two approaches to describe the dynamics of systems with contacts are proposed and evaluated: 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 [2]. 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. IV) that we solve by using the Interior-Point (IP) method implemented in IPOPT [3] and discussed in Sec. V. To improve computational efficiency, we also annotate sparsity in the linear algebra expressions and leverage algorithmic differentiation (AD) [4] to calculate derivatives both quickly and accurately: the adoption of the CasADi framework [5], described in Sec. V, provides a profitable interface to all the above tools with a consistent API.

In Sec. VI, 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. VII.

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.

II. RELATED WORK

A. 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 [6] and manipulation [7]. 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 [8]. Also the idea of programming using environmental constraints is well entrenched in robotics literature, starting with the seminal work [9], proceeding with [10], and culminating in the *iTaSC* framework [11].

The exploitation of complex interactions with the environment in a manipulation task also plays a central role

M. Gabiccini, A. Artoni and G. Pannocchia are with Department of Civil and Industrial Engineering, Univ. of Pisa, Italy ({firstname.lastname}@unipi.it). J. Gillis is with the Department of Electrical Engineering, KU Leuven, Belgium (joris.gillis@kuleuven.be). Corresponding author is M. Gabiccini.

in automation and manufacturing to design fixtures [12] and part feeders [13]. However, these works are highly specialized and they are limited to the case of handling a single object geometry.

B. 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 [14] and GraspIt! [15]) 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 [16] 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 [17], albeit not systematically. While sequences of EC exploitation primitives have been shown to be robust and capable [18], [19], there has been no comprehensive research on how to enumerate and describe these primitives or how to sequence them into task-directed manipulation plans.

C. General Purpose Planning Algorithms

State-of-the-art sampling-based planning algorithms like RRT*[20] seem not tailored for situations where *a-priori* unknown contact interactions may cause a combinatorial explosion of system configurations. Recent extensions, initially presented in [21] for RRT, and successively devised for RRT* in [22], were able to cope with systems described by complex and underactuated dynamics. In [23], 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 [24].

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.

D. Machine Learning Approaches

Although significant progresses have been made in this area in recent years [25], 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 [26], [27], [28]. Innovative policy classes [29] 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 [30] 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.

E. 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 [33] 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 [34] 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 [35] 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 [37].

The approach presented in [38] 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.

III. DYNAMICS OF SYSTEMS WITH CONTACTS

A. 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

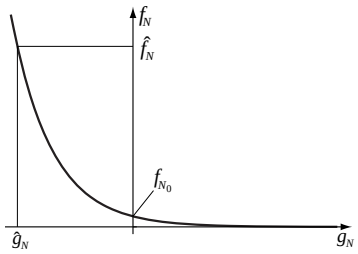

Fig. 1. Normal contact force as a function of the normal gap.

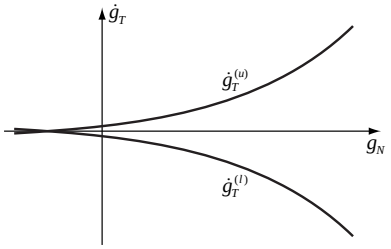

Fig. 2. Example of sliding velocity funnel.

(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 normal 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 [39]) 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:

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

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 $[t_k, t_{k+1}]$, 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

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

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).

B. 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 $k \in \{0,\dots,N-1\}$, we adopt a Backward-Euler transcription scheme by writing the *kinematic reconstruction* and the *dynamic* equations as

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

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

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 *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 [2], *unilateral* contacts with friction can be described by the following set of *inequality* and *complementarity* conditions

$$0 \le \lambda_{N_{k+1}} \perp g_N(q_{k+1}) \ge 0 \tag{9a}$$

$$0 \le \lambda_{T_{k+1}} \perp \big(\dot g_T(q_{k+1}, v_{k+1}) + E\gamma_{k+1}\big) \ge 0 \tag{9b}$$

$$0 \le \gamma_{k+1} \perp \big[\mu\lambda_{N_{k+1}} - (\lambda_{T_{k+1}}^\top H\, \lambda_{T_{k+1}})^{\frac{1}{2}}\big] \ge 0, \tag{9c}$$

where $g_N(\cdot)$ is the normal gap function and $\dot g_T(\cdot)$ is the time derivative of the tangential gap function [39], γ represents, in most cases[3], an approximation to the magnitude of the relative contact velocity, matrix $E := \mathrm{BlockDiag}(\mathbf{1},\dots,\mathbf{1}) \in \mathbb{R}^{(dc)\times c}$, with $\mathbf{1} \in \mathbb{R}^d$, $\mu \ge 0$ is the coefficient of friction, and $H := \mathrm{BlockDiag}(H^{(1)},\dots,H^{(c)})$, where the $H_{lm}^{(i)} = t_{i,l}^\top t_{i,m}$ $(l,m \in \{1,\dots,d\})$ is the metric form [40, 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 bjects 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].

IV. TRAJECTORY PLANNING AS AN OPTIMIZATION PROBLEM

A. 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: $f_{N_k}^{(1)} + f_{N_k}^{(2)} \ge \varepsilon$, $f_{N_k}^{(2)} + f_{N_k}^{(3)} \ge \varepsilon$, and $f_{N_k}^{(3)} + f_{N_k}^{(1)} \ge \varepsilon$, with $\varepsilon > 0$.

[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.

[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 [41], 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 [42, Sec. 5].

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).

B. 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 $v^{(l)}_{\min} \leq v^{(l)} \leq v^{(l)}_{\max}$ and $a^{l}_{\min} h \leq \Delta v^{(l)} \leq a^{(l)}_{\max} h$, 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

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

C. 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

$$\min_{\boldsymbol{v}} f(\boldsymbol{v}), \qquad \text{subject to} \tag{13a}$$

$$g_{\min} \leq g(\boldsymbol{v}) \leq g_{\max} \tag{13b}$$

$$\boldsymbol{v}_{\min} \leq \boldsymbol{v} \leq \boldsymbol{v}_{\max} \tag{13c}$$

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 [43] 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 [44, Sec. 11.3], thereby maximizing physical realism while facilitating convergence.

V. NONLINEAR PROGRAMMING VIA AN INTERIOR-POINT ALGORITHM

A. 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 (13b) into equality constraints.[6]

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

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

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.

B. Interior-point approach to NLP

Any local minimizer to (15) must satisfy the following Karush-Kuhn-Tucker (KKT) conditions [44, Sec. 12.2]

$$\nabla f(x) + \nabla c(x) \lambda - \underline{z} + \bar{z} = 0 \tag{16a}$$

$$c(x) = 0 \tag{16b}$$

$$\underline{z}^{(i)} (x^{(i)} - x^{(i)}_{\min}) - \mu = 0 \quad \forall i \in I_{\min} \tag{16c}$$

$$\bar{z}^{(i)} (x^{(i)}_{\max} - x^{(i)}) - \mu = 0 \quad \forall i \in I_{\max} \tag{16d}$$

[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 $f(\boldsymbol{v})$.

TABLE I

BASIC ALGORITHM IMPLEMENTED IN IPOPT.

1) Define $\mu_0 > 0$, x_0 ($x_{\min} \leq x_0 \leq x_{\max}$), λ_0, $\underline{z}_0 \geq 0$, $\overline{z}_0 \geq 0$, 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.

for some vectors $\lambda \in \mathbb{R}^m$, $\underline{z} \in \mathbb{R}^n$, and $\overline{z} \in \mathbb{R}^n$ (for completeness: $\underline{z}^{(i)} = 0 \quad \forall i \notin I_{\min}$, $\overline{z}^{(i)} = 0 \quad \forall i \notin I_{\max}$). Notice that, if $\mu = 0$, then (16) together with $\underline{z} \geq 0$ and $\overline{z} \geq 0$ represent the KKT conditions for NLP (14). The KKT conditions (16) form a nonlinear algebraic system $F(\xi) = 0$ in the unknown $\xi = (x, \lambda, \underline{z}, \overline{z})$, 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 I (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).

C. 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 [45]. 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 [46] from the Harwell Software Library [47] is used.

D. The CasADi framework

The transcribed optimal control problem is coded in a scripting environment using the Python [48] interface to the open-source CasADi framework [5], 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 [49] and its unidirectional graph coloring is used to obtain the Jacobian with a reduced number of AD sweeps [50]. 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 [45]: 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.

VI. APPLICATION EXAMPLES

A. Environment-aware manipulation

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

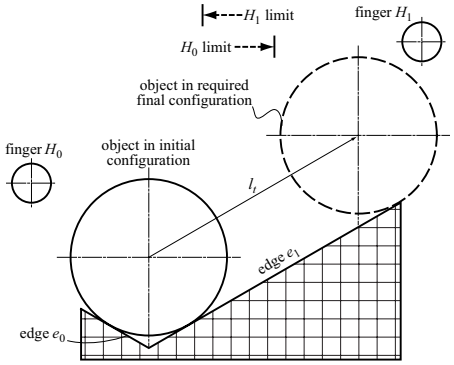

Fig. 3. EC manipulation scenario: the object must reach its final configuration at the prescribed final time $T = 8$ s.

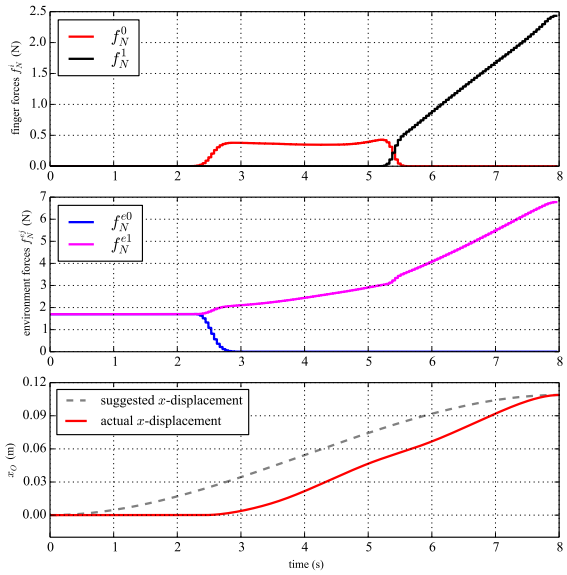

Fig. 4. Trajectories for a circular object manipulated by two independent fingers: normal contact forces f_N^i applied by the fingers; normal contact forces f_N^{ej} applied by the environment (segments e_0, e_1); suggested and actual x-displacement x of the object.

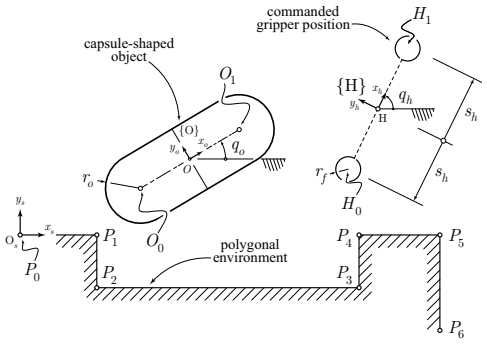

Fig. 5. EC manipulation scenario. Starting at $q_O(0) = \pi/2$, in contact with segment P_2P_3, the capsule must be placed, at time $T = 5$ s, in the same position but with $q_O(T) = -\pi/2$.

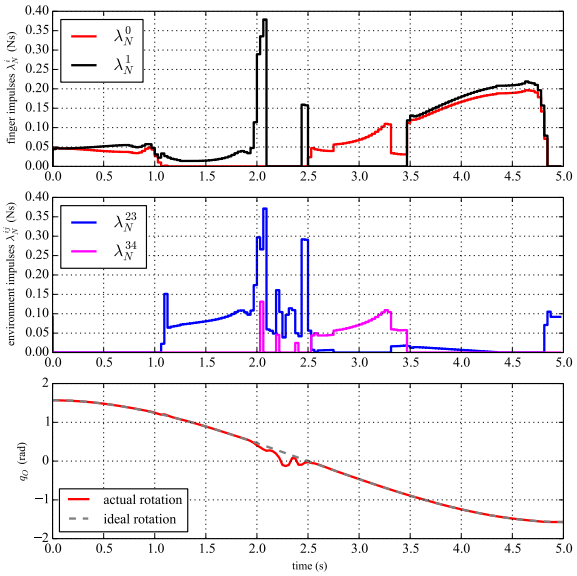

Fig. 6. Motion trajectories of a capsule-shaped object manipulated by an underactuated gripper: contact normal impulses λ_N^0 and λ_N^1 applied by the fingers; contact normal impulses λ_N^{ij} applied by the environment through segment P_iP_j; ideal and actual rotation q_O of the object-fixed frame.

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. An animation of the obtained results can be found in part A.1 of the accompanying video, 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.

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 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, 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 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 $k \in \{0,\dots,N-1\}$, 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 $n = 7375$. An animation of the results can be found in part A.2 of the accompanying video.

B. 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 IV-A 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 $n = 12585$. An animation is provided in part B.1 of the accompanying video, 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.

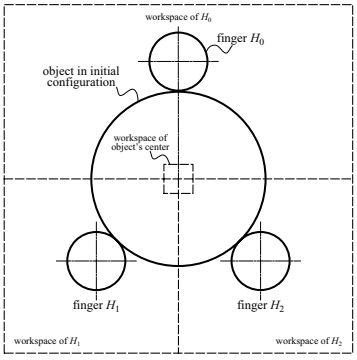

Fig. 7. Dexterous manipulation scenario: the object must find itself rotated by 360 deg at the final time $T = 6$ s.

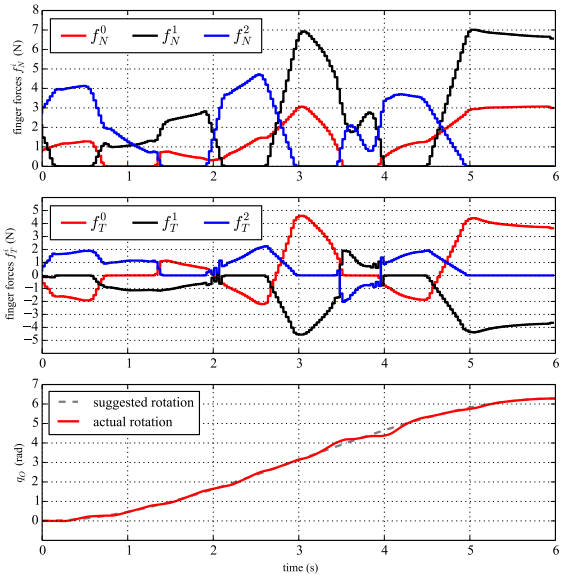

Fig. 8. Trajectories for a circular object manipulated by three independent fingers. Shown are: normal contact forces f_N^i and tangential contact forces f_T^i applied by the fingers; suggested and actual rotation q_O of the object.

VII. 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 VI-A). The results presented in sub-section VI-B 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. Injec-

tion of motion 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.

REFERENCES

[1] J. T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM, 2001.

[2] 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*, vol. 39, pp. 2673–2691, 1996.

[3] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575 – 582, 2009.

[4] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, ser. Frontiers in Appl. Math. Philadelphia, PA: SIAM, 2000, no. 19.

[5] J. 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.

[6] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *Int. Journal of Robotics Research (IJRR)*, vol. 3, no. 1, pp. 3–24, Mar. 1984.

[7] M. T. Mason, "The mechanics of manipulation," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 2, 1985, pp. 544–548.

[8] G. 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*, vol. 28, no. 4, pp. 343–399, 2000.

[9] A. P. Ambler and R. J. Popplestone, "Inferring the positions of bodies from specified spatial relationships," *Artificial Intelligence*, vol. 6, pp. 157–174, 1975.

[10] C. Samson, M. L. Borgne, and B. Espiau., *Robot Control, the Task Function Approach*. Clarendon Press, Oxford, England, 1991.

[11] J. D. Schutter, T. D. Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, 2007.

[12] I. Boyle, Y. Rong, and D. C. Brown, "A review and analysis of current computer-aided fixture design approaches," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 1–12, Feb. 2011. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0736584510000499

[13] G. Boothroyd, C. Poli, and L. Murch, "Handbook of feeding and orienting techniques for small parts," University of Massachusetts at Amherst, Dept. of Mechanical Engineering, Automation Project, 1981.

[14] R. Diankov, "Automated construction of robotic manipulation programs," PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[15] A. Miller and P. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics and Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[16] 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.

[17] M. R. Dogar and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Autonomous Robots*, vol. 33, no. 3, pp. 217–236, June 2012.

[18] N. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. Srinivasa, M. Erdmann, M. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Regrasping objects using extrinsic dexterity," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2014, pp. 2560–2560.

[19] M. Kazemi, J.-S. Valois, J. Bagnell, and N. Pollard, "Human-inspired force compliant grasping primitives," *Autonomous Robots*, pp. 1–17, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10514-014-9389-9

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[21] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Anchorage, Alaska, USA, 5 2010, pp. 5021–5028.

[22] 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.

[23] Y. Koga and J.-C. Latombe, "On multi-arm manipulation planning," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1994.

[24] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Robotics: Science and Systems (RSS)*, 2014.

[25] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. Journal of Robotics Research (IJRR)*, vol. 32, no. 11, pp. 1238–1274, 2013.

[26] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2004.

[27] 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.

[28] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[29] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," 2003.

[30] R. Jonschkowski and O. Brock, "State representation learning in robotics: Using prior knowledge about physical interaction," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[31] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Neural Information Processing Systems (NIPS)*, 2014.

[32] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," 2015, under review.

[33] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 5, pp. 783–792, 2010.

[34] W. Xi and C. Remy, "Optimal gaits and motions for legged robots," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 3259–3265.

[35] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," in *ACM Transactions on Graphics*, 2012.

[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 and E. Todorov, "Combining the benefits of function approximation and trajectory optimization," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

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

[39] P. Wriggers, *Computational contact mechanics*. Berlin, New York: Springer, 2006. [Online]. Available: http://opac.inria.fr/record=b1120968

[40] M. P. Do Carmo, *Differential Geometry of Curves and Surfaces*. Englewood Cliffs: Prentice Hall, 1976.

[41] D. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with coulomb friction," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 1. IEEE, 2000, pp. 162–169.

[42] C. Glocker and C. Studer, "Formulation and preparation for numerical evaluation of linear complementarity systems in dynamics," *Multibody System Dynamics*, vol. 13, pp. 447–463, 2005.

[43] 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*, vol. 106, pp. 25–57, 2006.

[44] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer Verlag, 2006.

[45] A. Griewank and A. Walther, *Evaluating Derivatives*, 2nd ed. SIAM, 2008.

[46] I. S. Duff, "Ma57 – a code for the solution of sparse symmetric definite and indefinite systems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 30, no. 2, pp. 118–144, 2004.

[47] HSL, "A collection of fortran codes for large scale scientific computation." 2014. [Online]. Available: http://www.hsl.rl.ac.uk

[48] T. E. Oliphant, "Python for scientific computing," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, 2007.

[49] 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.

[50] A. H. Gebremedhin, F. Manne, and A. Pothen, "What color is your Jacobian? Graph coloring for computing derivatives," *SIAM Review*, vol. 47, pp. 629–705, 2005.

APPENDIX

A. Transforming problem (13) into (14)

When all components of $g_{\min}$ and $g_{\max}$ are finite and $g_{\min} < g_{\max}$ we can write

$$g_{\min} \le g(\boldsymbol{v}) \Leftrightarrow \begin{cases} g_{\min} + s_{\min} - g(\boldsymbol{v}) = 0 \\ s_{\min} \ge 0 \end{cases} \tag{18a}$$

$$g(\boldsymbol{v}) \le g_{\max} \Leftrightarrow \begin{cases} g(\boldsymbol{v}) + s_{\max} - g_{\max} = 0 \\ s_{\max} \ge 0 \end{cases} \tag{18b}$$

Hence

$$x = \begin{bmatrix} \boldsymbol{v} \\ s_{\min} \\ s_{\max} \end{bmatrix}, \quad x_{\min} = \begin{bmatrix} v_{\min} \\ 0 \\ 0 \end{bmatrix}, \quad x_{\max} = \begin{bmatrix} v_{\max} \\ \infty \mathbf{1} \\ \infty \mathbf{1} \end{bmatrix} \tag{18c}$$

$$c(x) = \begin{bmatrix} g_{\min} + s_{\min} - g(\boldsymbol{v}) \\ g(\boldsymbol{v}) + s_{\max} - g_{\max} \end{bmatrix} \tag{18d}$$

When some components of $g_{\min}$ or $g_{\max}$ are unbounded, those constraints are discarded and no slack components are necessary. Similarly, when for some i, there holds $g^{(i)}_{\min} = g^{(i)}_{\max}$, the transformation of $g^{(i)}_{\min} \le g(\boldsymbol{v}) \le g^{(i)}_{\max}$ into a component of the equality constraint vector $c(x)$ is obvious, and again no slack component is necessary.

B. Further comments to the Interior-Point method discussed in V-B

We observe that the inner loop (Steps 2–4) is exited when the current iterate satisfies $E_{\mu_j}(\xi_{k+1}) \le \kappa\mu_j$. Then the value of μ_j is reduced in Step 5 performing an iteration of the outer loop (Steps 2–5). This implies that the solution of the inner loop becomes more and more accurate as the outer loop is executed. We also notice that $E_0(\cdot)$, i.e. $E_\mu(\cdot)$ with $\mu = 0$, represents the error of the KKT system for NLP (14). Hence, the overall algorithm is terminated when the KKT system is satisfied within a tolerance ε, specified by the user.

A crucial part of an effective interior-point algorithm is the line search that is performed in Step 2 using backtracking, i.e. starting from a large candidate value for α_k and reducing it when suitable conditions for acceptance are not satisfied. In general a step is acceptable if it makes some progress towards feasibility, i.e. reducing $\|c(x)\|$, and/or achieves an improvement in the objective function $\varphi_{\mu_j}(x)$. This is typically achieved in IPOPT by also including, in the line search, a so-called second-order correction Newton step which aims to solve solely $c(x) = 0$. In particular situations, the achieved value of α_k is too small, and this forces the algorithm to enter a *restoration* phase. For details and reasoning of this phase the interested reader is referred to [43].

C. Symmetric linear system solved in place of (17)

System (17) is non-symmetric; IPOPT performs elimination of the last two row blocks (and elimination of p^z_k and $p^{\bar{z}}_k$) obtaining the following symmetric system to be solved

$$\begin{bmatrix} W_k + \underline{\Sigma}_k + \overline{\Sigma}_k & A_k \\ A_k^T & 0 \end{bmatrix} \begin{bmatrix} p^x_k \\ p^\lambda_k \end{bmatrix} = - \begin{bmatrix} \nabla\varphi_{\mu_j}(x_k) + A_k\lambda_k \\ c(x_k) \end{bmatrix} \tag{19}$$

in which $\underline{\Sigma}_k$ and $\overline{\Sigma}_k$ are diagonal matrices with entries

$$\sigma^{(i)}_{\min} = \begin{cases} \underline{z}^{(i)} / (x^{(i)} - x^{(i)}_{\min}) & \text{if } i \in I_{\min} \\ 0 & \text{if } i \notin I_{\min} \end{cases} \tag{20a}$$

$$\sigma^{(i)}_{\max} = \begin{cases} \overline{z}^{(i)} / (x^{(i)}_{\max} - x^{(i)}) & \text{if } i \in I_{\max} \\ 0 & \text{if } i \notin I_{\max} \end{cases} \tag{20b}$$

APPENDIX

A. Transforming problem (13) into (14)

When all components of $g_{\min}$ and $g_{\max}$ are finite and $g_{\min} < g_{\max}$ we can write

$$g_{\min} \le g(\boldsymbol{v}) \Leftrightarrow \begin{cases} g_{\min} + s_{\min} - g(\boldsymbol{v}) = 0 \\ s_{\min} \ge 0 \end{cases} \tag{18a}$$

$$g(\boldsymbol{v}) \le g_{\max} \Leftrightarrow \begin{cases} g(\boldsymbol{v}) + s_{\max} - g_{\max} = 0 \\ s_{\max} \ge 0 \end{cases} \tag{18b}$$

Hence

$$x = \begin{bmatrix} \boldsymbol{v} \\ s_{\min} \\ s_{\max} \end{bmatrix}, \quad x_{\min} = \begin{bmatrix} v_{\min} \\ 0 \\ 0 \end{bmatrix}, \quad x_{\max} = \begin{bmatrix} v_{\max} \\ \infty \mathbf{1} \\ \infty \mathbf{1} \end{bmatrix} \tag{18c}$$

$$c(x) = \begin{bmatrix} g_{\min} + s_{\min} - g(\boldsymbol{v}) \\ g(\boldsymbol{v}) + s_{\max} - g_{\max} \end{bmatrix} \tag{18d}$$

When some components of $g_{\min}$ or $g_{\max}$ are unbounded, those constraints are discarded and no slack components are necessary. Similarly, when for some i, there holds $g^{(i)}_{\min} = g^{(i)}_{\max}$, the transformation of $g^{(i)}_{\min} \le g(\boldsymbol{v}) \le g^{(i)}_{\max}$ into a component of the equality constraint vector $c(x)$ is obvious, and again no slack component is necessary.

B. Further comments to the Interior-Point method discussed in V-B

We observe that the inner loop (Steps 2–4) is exited when the current iterate satisfies $E_{\mu_j}(\xi_{k+1}) \le \kappa\mu_j$. Then the value of μ_j is reduced in Step 5 performing an iteration of the outer loop (Steps 2–5). This implies that the solution of the inner loop becomes more and more accurate as the outer loop is executed. We also notice that $E_0(\cdot)$, i.e. $E_\mu(\cdot)$ with $\mu = 0$, represents the error of the KKT system for NLP (14). Hence, the overall algorithm is terminated when the KKT system is satisfied within a tolerance ε, specified by the user.

A crucial part of an effective interior-point algorithm is the line search that is performed in Step 2 using backtracking, i.e. starting from a large candidate value for α_k and reducing it when suitable conditions for acceptance are not satisfied. In general a step is acceptable if it makes some progress towards feasibility, i.e. reducing $\|c(x)\|$, and/or achieves an improvement in the objective function $\varphi_{\mu_j}(x)$. This is typically achieved in IPOPT by also including, in the line search, a so-called second-order correction Newton step which aims to solve solely $c(x) = 0$. In particular situations, the achieved value of α_k is too small, and this forces the algorithm to enter a *restoration* phase. For details and reasoning of this phase the interested reader is referred to [43].

C. Symmetric linear system solved in place of (17)

System (17) is non-symmetric; IPOPT performs elimination of the last two row blocks (and elimination of p^z_k and $p^{\bar{z}}_k$) obtaining the following symmetric system to be solved

$$\begin{bmatrix} W_k + \underline{\Sigma}_k + \overline{\Sigma}_k & A_k \\ A_k^T & 0 \end{bmatrix} \begin{bmatrix} p^x_k \\ p^\lambda_k \end{bmatrix} = - \begin{bmatrix} \nabla\varphi_{\mu_j}(x_k) + A_k\lambda_k \\ c(x_k) \end{bmatrix} \tag{19}$$

in which $\underline{\Sigma}_k$ and $\overline{\Sigma}_k$ are diagonal matrices with entries

$$\sigma^{(i)}_{\min} = \begin{cases} \underline{z}^{(i)} / (x^{(i)} - x^{(i)}_{\min}) & \text{if } i \in I_{\min} \\ 0 & \text{if } i \notin I_{\min} \end{cases} \tag{20a}$$

$$\sigma^{(i)}_{\max} = \begin{cases} \overline{z}^{(i)} / (x^{(i)}_{\max} - x^{(i)}) & \text{if } i \in I_{\max} \\ 0 & \text{if } i \notin I_{\max} \end{cases} \tag{20b}$$