A multi-level optimization framework for simultaneous grasping and motion planning

Simon Zimmermann[1], Ghazal Hakimifard[1], Miguel Zamora[1], Roi Poranne[1,2], Stelian Coros[1]

Abstract—**We present an optimization framework for grasp and motion planning in the context of robotic assembly. Typically, grasping locations are provided by higher level planners or as input parameters. In contrast, our mathematical model simultaneously optimizes motion trajectories, grasping locations, and other parameters such as the pose of an object during handover operations. The input to our framework consists of a set of objects placed in a known configuration, their target locations, and relative timing information describing when objects need to be picked up, optionally handed over, and dropped off. To allow robots to reason about the way in which grasping locations govern optimal motions, we formulate the problem using a *multi-level* optimization scheme: the top level optimizes grasping locations; the mid-layer level computes the configurations of the robot for pick, drop and handover states; and the bottom level computes optimal, collision-free motions.**

We leverage sensitivity analysis to compute derivatives analytically (how do grasping parameters affect IK solutions, and how these, in turn, affect motion trajectories etc.), and devise an efficient numerical solver to generate solutions to the resulting optimization problem. We demonstrate the efficacy of our approach on a variety of assembly and handover tasks performed by a dual-armed robot with parallel grippers.

Index Terms—**Grasping, Motion and Path Planning, Assembly**

I. Introduction

GRASP planning and motion planning are two major challenges in robotics. They are commonly addressed separately, and indeed, in many cases, this treatment proves to be sufficient. For example, simple pick-and-place tasks in environments clear of obstacles can reliably be addressed by solving the two problems independently. As such, many grasping techniques focus solely on the grasping itself, without accounting for the rest of the task. As a result, a motion planning following grasp planning might return with no feasible solution. In other words, the existence of obstacles might cause an uninformed grasp planner to provide a grasp for which it is impossible to find a collision-free motion. This type of challenge occurs often in assembly tasks, where the environment is initially free of obstacles, but becomes increasingly constrained as construction progresses (Fig. 1). Evidently, as assembly tasks become more and more complex, a more comprehensive approach is required.

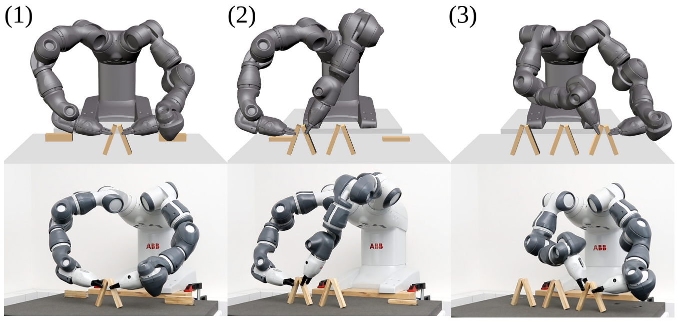


Fig. 1. Assembly of a simple triple truss structure. As the assembly process progresses, the environment becomes more and more constrained. While part (1) can be assembled without concerns, for part (2) and (3) the grip must be adapted to avoid collisions.

Our goal in this paper is to lay an alternative, foundational framework for simultaneous optimization of grasps and motions. The main challenge is that motion optimization assumes that the grasp is already known, but the grasp optimization must be *motion aware* in order to account for potential collisions. This chicken-and-egg problem has lead us to formulate it as a *multilevel* optimization [1]. The special case of an unconstrained *bilevel* problem is generally expressed as follows:

$$\min_{\mathbf{x}_1,\mathbf{x}_2} \quad \mathcal{O}_2(\mathbf{x}_1,\mathbf{x}_2) \tag{1a}$$

$$\text{s.t.} \quad \mathbf{x}_1 = \arg\min_{\mathbf{x}_1^*} \mathcal{O}_1(\mathbf{x}_1^*,\mathbf{x}_2), \tag{1b}$$

where $\mathcal{O}_1, \mathcal{O}_2$ are objective functions, and $\mathbf{x}_1, \mathbf{x}_2$ are two distinct subsets of variables. Essentially, the problem in (1) asks to minimize $\mathcal{O}_1(\mathbf{x}_1,\mathbf{x}_2)$ w.r.t. both sets of variables, under the assumption that for any $\mathbf{x}_2$, the subset $\mathbf{x}_1$ minimizes $\mathcal{O}_1(\mathbf{x}_1,\mathbf{x}_2)$. The problems in (1a) and (1b) are called the top and bottom layer, respectively. A multilevel problem is the straightforward generalization of (1).

Why multilevel? To illustrate the benefit of a multilevel formulation, we consider the problem of optimal grasping in isolation. One can potentially first search for an optimal grasping location, and then attempt to find the corresponding robot configuration via inverse kinematics (IK). This is an attractive approach because the burden of finding joint angles falls on a "black box" IK solver. Clearly, the whole approach will fail if the grasping location is not reachable, however, or if the robot has to get into awkward configurations due to its

Manuscript received September, 10, 2019; Revised January, 3, 2020; Accepted January, 29, 2020.

This paper was recommended for publication by Editor Hong Liu upon evaluation of the Associate Editor and Reviewers' comments. *Here you can knowledge the organizations/grants which supported the work

[1] The authors are with the Department of Computer Science, ETH, Zurich, Switzerland. simon.zimmermann@inf.ethz.ch; hghazal@student.ethz.ch; mimora@ethz.ch; roi.poranne@inf.ethz.ch; scoros@gmail.com

[2]Roi Poranne is with the Department of Computer Science, University of Haifa, Haifa, Israel. roiporanne@cs.haifa.ac.il

Digital Object Identifier (DOI): see top of this page.

kinematic limitations. If we consider an optimization based IK solver instead, we can phrase the problem as follows: "find the optimal grasping location, subject to the corresponding IK solution being optimal". This informal description is essentially a bilevel problem, and its solution is both optimal in terms of the grasp, and achievable in terms of the joint angles. A more precise description is provided in section III.

We cast the problem of simultaneous grasp and motion planning using a *trilevel* optimization approach: the *top* layer finds optimal grasping locations, which are expressed in the local coordinate frame of the object being manipulated. The *mid*-layer finds optimal robot poses for the pick and place configurations using inverse kinematics. The *bottom* layer finds optimal, collision-free motion trajectories that connect these IK configurations. We describe the details of our formulation in section IV, and we emphasize that a solution to our problem is optimal in terms of both the grasping location and the overall motion. In other words, solutions represent the optimal grasp for the optimal motion given a specific assembly task. Having formulated the problem in such a way, we turn to the development of an efficient solution strategy. To this end, we leverage sensitivity analysis to compute first and second total derivatives of the top layer, which allows us to employ an efficient numerical solver based on Newton's method.

We demonstrate the performance of our framework on a set of pre-sequenced assembly tasks with wooden blocks, using a UR5 and a dual-armed YuMi® IRB 14000 with parallel grippers. We allow only longitudinal grasping similar to [2], but without any sliding or other in-hand corrections. This setup highlights the necessity for joint optimization of grasps and motion trajectories: without optimizing grasping locations, many assembly problems are infeasible.

II. Related work

Grasping and manipulation of objects has been investigated since the dawn of robotics, and is still one of the most active fields in robotic research. Due to the vastness of this topic, we mention only the most closely related and relevant papers. Recent reviews [3], [4], [5], [6], [7] discuss the various applications, challenges and algorithms, and the different aspects of grasping, such as sensing and control, optimization and machine learning. Different grasping strategies for different scenarios are discussed in [8].

Much of the research effort is aimed at solving the problem known as *grasp and motion planning*. Generally, the approach is to sample the *grasp space* and strategically pick grasps that are likely to be viable. If motion planning fails given the grasp, then a new grasp is picked until the motion planning is successful. This approach is similar to the task and motion planning (TAMP) [9], which generally suggests to treat motion planning as a black-box, and focus on task planning instead. This way, any motion planner can be used, as long as it can report whether it was successful or not. Examples of this approach appear in the work of Kragic and colleagues, e.g. ([10], [11]), as well as Huh et al. [12], which are based on sampling, or Ghalamzan et al. [13], which is learning-based. Another related stream of research led by Wan, Harada and colleagues focuses on object reorientation via regrasping, which we also target. see for example [14], [15], [16]. In addition, our model also supports two-hand regrasping, a capability useful for handing-over objects between arms, or to help one arm reorient an object using the other arm. This application was also discussed for example in [17], which proposed a sampling based approach similar to the previously mentioned techniques. Toussaint et al. [18], [19] considers task and motion planning, where motion planning includes grasping as well. The method builds on the concept of Logic-Geometric Programming [20], which uses conditional constraints based on a symbolic action sequence, that are optimized within the same framework. In contrast to the above mentioned work, we consider only the grasp and motion planning problem from a local, gradient-based optimization perspective. We treat the problem as a unified optimization problem, and propose a new optimization algorithm for it, which can potentially be used as the TAMP "black-box" solver. We mention that [18] and previous work use a general, custom built solver [21] that implements a standard Gauss-Newton method, with constraints handled using an improved Augmented Lagrangian method. Our solver on the other hand, is a novel solver dedicated to general multi-level optimization problems, which we apply to grasp and motion planning.

More related to our approach, [22] too formulates the problem as a multilevel optimization where the low level problem is motion planning. However, their top level is discrete, and does not leverage the differentiability of motion planning. We demonstrate that grasping locations *can* be efficiently optimized as continuous optimization parameters. Other work that consider multi-level optimization include [23] and [24]. In these cases, the goal is to solve a top level problem, subject to the solution being in the optimal set of the lower level problem. The underlying assumption is that the bottom level problem has an optimal set with high dimensionality due to redundancy, which is determined by the next level problem. The optimal set of the bottom level is locally approximated by a linear subspace (possibly with linear inequalities), from which a smaller subspace is selected based the next level problem. The process starts from the bottom, and iteratively produces finer and finer optimal sets, until it reaches the top level. In contrast, our approach does not assume there is a redundancy, but finds an optimal solution for the lower levels given the current candidate for the higher levels. Additionally, it informs the higher level, via the chain rule, how a change in the high level candidate will affect the low level solution.

As stated, we assume that the sequencing is predetermined, and target pick-and-place tasks which are typical during assembly operations. We note however that our framework can run several tasks concurrently. As we exclusively deal with wooden blocks, we can restrict the grasp space to longitudinal grasping, which assumes that the grippers only come in contact with the sides of the block. This space was utilized in several recent publications, for various in-hand manipulation tasks. For example, [2] used the dynamics of the arm and careful friction control to slide the object in the gripper. Similarly, [25] uses friction to *pivot* the object. Careful motion planning in the same space was used in [26] to solve the so-called

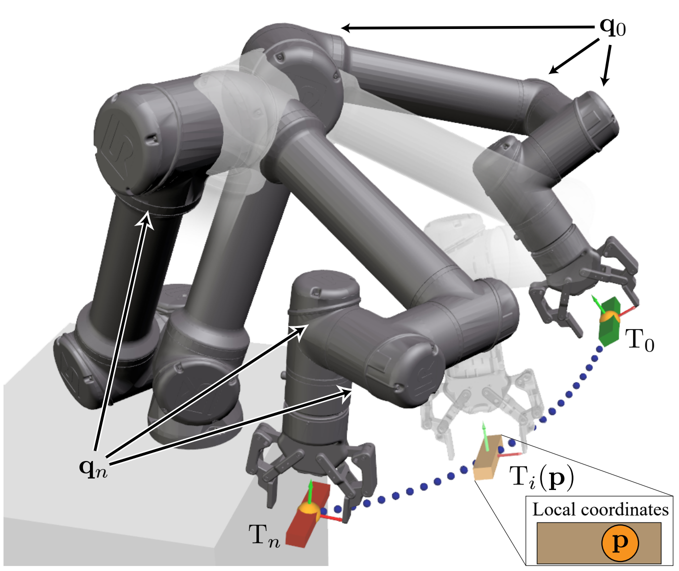


Fig. 2. System overview using a UR5. Given a pick and a place location of the block, our system simultaneously computes the optimal grasp location, the kinematics, and the motion.

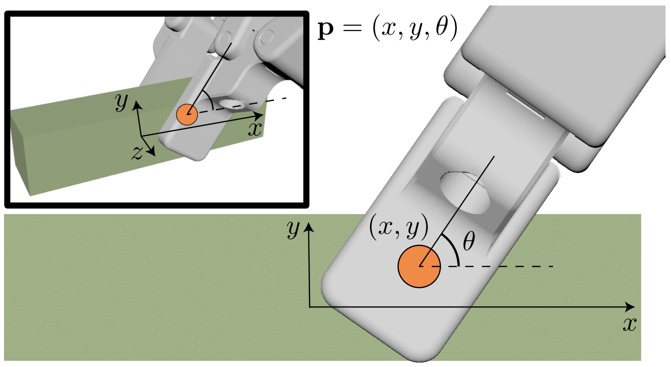


Fig. 3. We use a longitudinal grasp model, which can be parameterized by the x and y coordinates of the contact point, and the angle θ of the end effector.

”Shallow-Depth Insertion”. In contrast to these methods, we only consider the challenge of precise gripping and do not allow in hand manipulation.

III. Overview

Our goal in this paper is to demonstrate a complete solution for pre-scheduled (i.e. predefined pick up and drop off timings) assembly planning problems. To this end, we start describing our method on a simple pick-and-place task, as generalizing to more complex tasks is rather straightforward. Thus, given an initial and final pose of a block, we jointly search for the optimal grasp and the optimal motion. We show an overview of our optimization framework in Fig 2. Assuming the motion trajectory comprises of $n+1$ time steps, let T_0 and T_n be the initial and final transformations that take the block from its local frame to the world frame. The position and orientation of the gripper (i.e. the grasp location) as it makes contact with the block are expressed in the block's local coordinate frame. Since the grasping is longitudinal, it is enough to describe the grasping configuration in terms of the x and y coordinates of the point on the contact side of the block and the orientation θ of the gripper (See Fig. 3). We let $\mathbf{p} = (x, y, \theta)$. The robot is represented by its kinematic chain and its joint angles $\mathbf{q} = (q_1, \dots, q_m) \in \mathbb{R}^m$. We denote the joint angles at time step i by $\mathbf{q}_i$ and the joint limits by box constraints $\mathbf{q}_{\text{min}} \le \mathbf{q}_i \le \mathbf{q}_{\text{max}}$. Thus, given grasping parameters $\mathbf{p}$ and a transformation T from the local coordinate frame of the block to world coordinates, we can solve the inverse kinematics problem to find $\mathbf{q}$. We do so using an optimization approach. More precisely, we denote the forward kinematics function that returns the world pose of the end effector by $\mathcal{K}(\mathbf{q})$, and solve the following optimization problem:

$$\min_{\mathbf{q}} \quad \|\mathcal{K}(\mathbf{q}) - \mathrm{T}(\mathbf{p})\|^2 \tag{2a}$$

$$\text{s.t.} \quad \mathbf{q}_{\text{min}} \le \mathbf{q} \le \mathbf{q}_{\text{max}}. \tag{2b}$$

This problem is the basis of the mid-level optimization scheme we discuss in the following section. As described in the introduction, we note again that if $\mathbf{p}$ is a parameter given by an independent grasp planner, then successfully solving (2) (i.e. reaching a value of 0 for the objective), would complete the picture and provide the joint angles. A more inclusive, bilevel formulation is the following:

$$\min_{\mathbf{p},\mathbf{q}} \quad \mathcal{O}_{\text{grasp}}(\mathbf{p}, \mathbf{q}) \tag{3a}$$

$$\text{s.t.} \quad \mathbf{q} = \arg\min_{\mathbf{q}^*} \quad \|\mathcal{K}(\mathbf{q}^*) - \mathrm{T}(\mathbf{p})\|^2 \tag{3b}$$

$$\text{s.t.} \quad \mathbf{q}_{\text{min}} \le \mathbf{q}^* \le \mathbf{q}_{\text{max}}, \tag{3c}$$

where (3b),(3c) are the same as the optimization problem in (2). We can think of this optimization as a function $\mathbf{q}(\mathbf{p})$ that returns the joint angles for a specific grasping location. If this function was known explicitly, we could have substituted $\mathbf{q}$ for $\mathbf{q}(\mathbf{p})$ in (3), and write the equivalent unconstrained problem

$$\min_{\mathbf{p}} \quad \mathcal{O}_{\text{grasp}}(\mathbf{p}, \mathbf{q}(\mathbf{p})).$$

Unfortunately, there does not exist a closed form solution in general. Nevertheless, to solve the optimization problem above using Newton's method, we only need to be able to compute the first and second total derivative of $\mathcal{O}_{\text{grasp}}(\mathbf{p}, \mathbf{q}(\mathbf{p}))$, which turns out to be rather simple using *sensitivity analysis*. We elaborate on this in section IV-B.

Before we move on, we mention that our implementation handles joint limit constraints using barrier functions which are inspired by interior point methods. This allows us to employ unconstrained optimization methods which generally converge faster. For example, instead of the constraint $q_i > q_{i,\text{min}}$, we add a barrier function $B(q_i - q_{i,\text{min}})$ to the objective, where $B(t)$ quickly increases as t approaches zero. Thus, in the following we omit constraints like (2b), since they will be expressed as part of the objective function. More details appear in section IV-C.

IV. A MULTILEVEL FORMULATION

In this section we describe the optimization problem, first without going into the details of every objective. We then explain our solution for general multilevel problems, based on sensitivity analysis. Finally, we provide the details of our formulation.

A. The optimization problem

As noted, a convenient approach to jointly optimize grasps and motions is via multilevel optimization. There are countless ways of splitting the problem into levels. The specific approach we choose reflects the notion that boundary conditions should be handled on higher levels, and that given a set of boundary condition, "filling-in" the missing parts is simpler. Thus, in our case, the boundary conditions of the motion $\mathbf{q}_i, i = 0, \dots, n$, are the pick and place *joint angles* given by $\mathbf{q}_0$ and $\mathbf{q}_n$. These in turn are determined by *their* "boundary conditions" e.g. desired end effector poses, which are none other than the grasping points $\mathrm{T}_0(\mathbf{p})$ and $\mathrm{T}_n(\mathbf{p})$. Finally, the *local* grasping point itself, $\mathbf{p}$, is optimized in the top level.

More formally, we define the prototype for our problem using this notation:

$$\min_{\mathbf{p},\mathbf{q}_{0n},\mathbf{Q}} \quad \mathcal{O}_{\text{grasp}}(\mathbf{p}, \mathbf{q}_{0n}, \mathbf{Q}) \tag{4a}$$

$$\text{s.t.} \quad \mathbf{q}_{0n} = \arg\min_{\mathbf{q}^*_{0n},\mathbf{Q}^*} \mathcal{O}_{\text{pose}}(\mathbf{p}, \mathbf{q}^*_{0n}, \mathbf{Q}^*) \tag{4b}$$

$$\text{s.t.} \quad \mathbf{Q} = \arg\min_{\mathbf{Q}^*} \mathcal{O}_{\text{motion}}(\mathbf{p}, \mathbf{q}_{0n}, \mathbf{Q}^*), \tag{4c}$$

where we stack together the pick-and-place joint angles $\mathbf{q}_0$ and $\mathbf{q}_n$ and denote them by $\mathbf{q}_{0n}$, and denote the rest of the $\mathbf{q}_i$'s for $i = 1, \dots, n-1$ by $\mathbf{Q}$. We explain how to solve this problem in the following.

B. Optimization

The basis for our approach to solve (4) dates back to at least [27], and appears under many different names, such as the adjoint method or sensitivity analysis [28]. Originally employed for optimization with equality constraints, the goal is to compute the derivative of the top level in terms of the low-level variables only. This is done using the chain-rule and the implicit function theorem.

The derivation for the bilevel case is common and can be found in literature. Here we derive the extension to trilevel optimization. We begin by writing (4) in a more compact and symmetric form:

$$\mathbf{x}_3 = \arg\min_{\mathbf{x}^*_1,\mathbf{x}^*_2,\mathbf{x}^*_3} \quad \mathcal{O}_3(\mathbf{x}^*_1, \mathbf{x}^*_2, \mathbf{x}^*_3) \tag{5a}$$

$$\text{s.t.} \quad \mathbf{x}_2 = \arg\min_{\mathbf{x}^*_2,\mathbf{x}^*_1} \mathcal{O}_2(\mathbf{x}^*_1, \mathbf{x}^*_2, \mathbf{x}_3) \tag{5b}$$

$$\text{s.t.} \quad \mathbf{x}_1 = \arg\min_{\mathbf{x}^*_1} \mathcal{O}_1(\mathbf{x}^*_1, \mathbf{x}_2, \mathbf{x}_3), \tag{5c}$$

The bottom two layers are essentially a bilevel problem. In the continuous setting, the constraint (5c) can be equivalently posed as

$$\mathbf{g}_1(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) := \frac{d}{d\mathbf{x}_1}\mathcal{O}_1(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = 0.$$

With that, [29] shows how to compute the sensitivities $\frac{\mathrm{d}\mathbf{x}_1}{\mathrm{d}\mathbf{x}_2}$ and the total derivative of $\mathcal{O}_2$ w.r.t. $\mathbf{x}_1$. In our notation,

$$\frac{\mathrm{d}\mathbf{x}_1}{\mathrm{d}\mathbf{x}_2} = -\left(\frac{\partial \mathbf{g}_1}{\partial \mathbf{x}_1}\right)^{-1} \frac{\partial \mathbf{g}_1}{\partial \mathbf{x}_2}, \tag{6}$$

$$\frac{\mathrm{d}\mathcal{O}_2}{\mathrm{d}\mathbf{x}_2} = \frac{\partial \mathcal{O}_2}{\partial \mathbf{x}_1}\frac{\mathrm{d}\mathbf{x}_1}{\mathrm{d}\mathbf{x}_2} + \frac{\partial \mathcal{O}_2}{\partial \mathbf{x}_2}. \tag{7}$$

To compute $\frac{\mathrm{d}\mathcal{O}_3}{\mathrm{d}\mathbf{x}_3}$, we apply a similar procedure: We first observe that

$$\mathbf{g}_2(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) := \frac{d}{d\mathbf{x}_2}\mathcal{O}_2(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = 0.$$

Taking the total derivative w.r.t $\mathbf{x}_3$ we get,

$$\frac{\mathrm{d}\mathbf{g}_2}{\mathrm{d}\mathbf{x}_3} = \frac{\partial \mathbf{g}_2}{\partial \mathbf{x}_3} + \frac{\partial \mathbf{g}_2}{\partial \mathbf{x}_2}\frac{\mathrm{d}\mathbf{x}_2}{\mathrm{d}\mathbf{x}_3} + \frac{\partial \mathbf{g}_2}{\partial \mathbf{x}_1}\frac{\mathrm{d}\mathbf{x}_1}{\mathrm{d}\mathbf{x}_2}\frac{\mathrm{d}\mathbf{x}_2}{\mathrm{d}\mathbf{x}_3} = 0,$$

from which we get

$$\frac{\mathrm{d}\mathbf{x}_2}{\mathrm{d}\mathbf{x}_3} = -\left(\frac{\partial \mathbf{g}_2}{\partial \mathbf{x}_2} + \frac{\partial \mathbf{g}_2}{\partial \mathbf{x}_1}\frac{\mathrm{d}\mathbf{x}_1}{\mathrm{d}\mathbf{x}_2}\right)^{-1} \frac{\partial \mathbf{g}_2}{\partial \mathbf{x}_3}. \tag{8}$$

Then, all that is left to do is compute $\frac{\mathrm{d}\mathcal{O}_3}{\mathrm{d}\mathbf{x}_3}$ using the chain-rule and substitute (6) and (8):

$$\frac{\mathrm{d}\mathcal{O}_3}{\mathrm{d}\mathbf{x}_3} = \frac{\partial \mathcal{O}_3}{\partial \mathbf{x}_3} + \frac{\partial \mathcal{O}_3}{\partial \mathbf{x}_2}\frac{\mathrm{d}\mathbf{x}_2}{\mathrm{d}\mathbf{x}_3} + \frac{\partial \mathcal{O}_3}{\partial \mathbf{x}_1}\frac{\mathrm{d}\mathbf{x}_1}{\mathrm{d}\mathbf{x}_2}\frac{\mathrm{d}\mathbf{x}_2}{\mathrm{d}\mathbf{x}_3}.$$

Next, we compute the second derivative $\frac{\mathrm{d}^2\mathcal{O}_3}{\mathrm{d}\mathbf{x}_3^2}$. We again follow [29] and only compute a positive definite approximation that does not include tensor terms. This is beneficial since on one hand, it guarantees a descent direction and on the other hand, the evaluation of the approximation is cheaper. The approximation, in our notation, is

$$\begin{aligned}\frac{\mathrm{d}^2\mathcal{O}_2}{\mathrm{d}\mathbf{x}_2^2} \approx& \mathbf{S}_1^T \frac{\partial^2\mathcal{O}_2}{\partial \mathbf{x}_1^2}\mathbf{S}_1 + \mathbf{S}_1^T\frac{\partial^2\mathcal{O}_2}{\partial \mathbf{x}_1\partial \mathbf{x}_2} + \\ &+ \left(\frac{\partial^2\mathcal{O}_2}{\partial \mathbf{x}_1\partial \mathbf{x}_2}\right)^T \mathbf{S}_1 + \frac{\partial^2\mathcal{O}_2}{\partial \mathbf{x}_2^2},\end{aligned}$$

where $\mathbf{S}_1 := \frac{dx_1}{dx_2}$, and by re-using this term and using the same strategy we obtain,

$$\begin{aligned}\frac{\mathrm{d}^2\mathcal{O}_3}{\mathrm{d}\mathbf{x}_3^2} \approx& \mathbf{S}_2^T \frac{\mathrm{d}^2\mathcal{O}_2}{\mathrm{d}\mathbf{x}_2^2}\mathbf{S}_2 + \mathbf{S}_2^T\frac{\partial^2\mathcal{O}_3}{\partial \mathbf{x}_2\partial \mathbf{x}_3} + \\ &+ \left(\frac{\partial^2\mathcal{O}_3}{\partial \mathbf{x}_2\partial \mathbf{x}_3}\right)^T \mathbf{S}_2 + \mathbf{S}_3^T\frac{\partial^2\mathcal{O}_3}{\partial \mathbf{x}_1\partial \mathbf{x}_3} + \\ &+ \left(\frac{\partial^2\mathcal{O}_3}{\partial \mathbf{x}_1\partial \mathbf{x}_3}\right)^T \mathbf{S}_3 + \frac{\partial^2\mathcal{O}_3}{\partial \mathbf{x}_3^2},\end{aligned}$$

where $\mathbf{S}_2 := \frac{dx_2}{dx_3}$ and $\mathbf{S}_3 = \frac{dx_1}{dx_2}\frac{\mathrm{d}x_2}{\mathrm{d}x_3}$.

With both $\frac{\mathrm{d}^2\mathcal{O}_3}{\mathrm{d}\mathbf{x}_3^2}$ and $\frac{\mathrm{d}\mathcal{O}_3}{\mathrm{d}\mathbf{x}_3}$ we can use Newton's method to compute a search direction for $\mathbf{x}_3$ and optimize $\mathcal{O}_3$. One important issue to note is that this approach assumes and requires that $\mathbf{g}_1 = \mathbf{g}_2 = 0$. Therefore, before we can use $\frac{\mathrm{d}\mathcal{O}_2}{\mathrm{d}\mathbf{x}_2}$, we must optimize $\mathcal{O}_1$, and before we can use $\frac{\mathrm{d}\mathcal{O}_3}{\mathrm{d}\mathbf{x}_3}$, we must optimize $\mathcal{O}_2$. We optimize those using Newton's method as well.

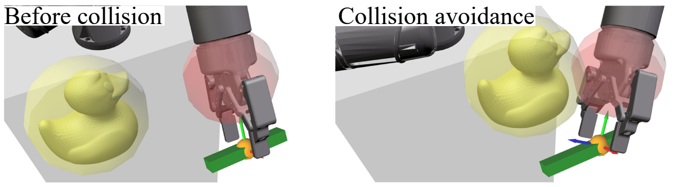


Fig. 4. Using our system, the user can interactively manipulate the obstacle, and the grip will automatically change accordingly. We use collision spheres as proxies for the objects, as shown. Please refer to the accompanying video to view the real-time interaction.

C. *Objectives and constraints*

We discuss the particular choices we made for the different objectives in the different levels, starting from the low-level: the motion optimization.

The motion level. This level assumes that $\mathbf{q}_0$ and $\mathbf{q}_n$ are given, and is required to find an interpolating motion that is optimally smooth, collision-free, and respect joint limits. For smoothness, we require the acceleration of the joint angles to be minimized. This is obtained by the objective

$$\mathcal{O}_{\text{smooth}}(\mathbf{q}_0, \mathbf{Q}, \mathbf{q}_n) = \sum \|\mathbf{q}_{i-1} - 2\mathbf{q}_i + \mathbf{q}_{i+1}\|^2.$$

Additionally, we require the end effectors to approach the pick and place poses slowly in order to account for the time necessary to open and close the grippers, i.e. we would like the joint velocity to be minimized at the ends of the trajectory:

$$\mathcal{O}_{\text{slow}}(\mathbf{q}_0, \mathbf{Q}, \mathbf{q}_n) = \|\mathbf{q}_1 - \mathbf{q}_0\|^2 + \|\mathbf{q}_{n-1} - \mathbf{q}_n\|^2.$$

As mentioned, we replace the joint limit by a soft barrier function. To this end, we define unilateral quadratic functions

$$B_c^+(t) = \begin{cases} 0 & t \leq c \\ (t-c)^2 & t > c \end{cases} \quad B_c^-(t) = \begin{cases} (t-c)^2 & t \leq c \\ 0 & t > c \end{cases}.$$

and for each joint angle q_{ij} at timestep i we add the corresponding term:

$$\mathcal{O}_{\text{limits},i}(q_{ij}) = B^+_{q_{\max,i}}(q_{ij}) + B^-_{q_{\min,i}}(q_{ij}).$$

To avoid collisions, we use the same collision model as in [30], where the robot arms, obstacles and blocks are approximated by spherical proxies (see Fig. 4 for example). We denote this objective by $\mathcal{O}_{collision}(\mathbf{q})$ and apply it to each $\mathbf{q}_i$. Finally, we define $\mathcal{O}_{\text{motion}}$ to be a weighted sum of the above-mentioned objectives.

The IK level. In this level we are given the grasp poses $\mathbf{p}$ as input, and the objective is to find the appropriate joint angles. This is the same as the objective in (2a), applied to $\mathbf{q}_0$ and $\mathbf{q}_n$, that is,

$$\mathcal{O}_{\text{IK}}(\mathbf{q}_0, \mathbf{q}_n, \mathbf{p}) = \|\mathcal{K}(\mathbf{q}_0) - \text{T}_0(\mathbf{p})\|^2 + \|\mathcal{K}(\mathbf{q}_n) - \text{T}_n(\mathbf{p})\|^2.$$

In addition, we apply the same joint limit penalty as above to $\mathbf{q}_0$ and $\mathbf{q}_n$.

The grasp level. The main objective of this level is to find the grasping parameters $\mathbf{p}$. The only requirement that must be fulfilled is that $\mathbf{p}$ lies within the block. To enforce this, we use a penalty similar to (IV-C), namely,

$$\mathcal{O}_{\text{grasp}}(p) = B^+_{b_x}(x) + B^-_{-b_x}(x) + B^+_{b_y}(y) + B^-_{-b_y}(y).$$

where we recall that $\mathbf{p} = (x, y, \theta)$ and the block's dimensions on the grasped side are $[-b_x, b_x] \times [-b_y, b_y]$. In addition, we add several of the lower level objectives to the high level as well. These are the collision avoidance, smoothness and IK terms. This is a very important step, as we illustrate with this example: The mid-level optimization solves for an IK solution given grasp parameters $\mathbf{p}$. As we employ an optimization-based approach, unreachable configurations also correspond to a minimum of $\mathcal{O}_{\text{IK}}$ – this means the gradient of this objective with respect to $\mathbf{q}_0$ and $\mathbf{q}_n$ is 0, but the objective does not reach a value of 0. Consequently, at the top level, the gradient of the overall objective captures the way in which the IK solution changes with respect to $\mathbf{p}$, and how this change affects $\mathcal{O}_{\text{IK}}$ in turn. This enables $\mathbf{p}$ to be adjusted such that the IK solution becomes feasible. Similarly, this formulation enables us to compute analytic derivatives that capture the way in which changes to $\mathbf{p}$ affect the smoothness and collision avoidance terms, and which are defined on the motion trajectory generated by the low level.

We leverage the formulation derived in the previous section to find optimal grasping configurations for the robot.

D. *Beyond pick-and-place*

So far we have described the method for single, independent pick-and-place tasks. However, it is easily generalizable to more complex tasks. The two types of tasks we highlight in this paper are concurrent pick-and-place tasks, and object handovers. Concurrent pick-and-place tasks for a dual-armed robot requires each arm to perform the task without collisions. Our system allows the user to specify the time step where the pick task and the place task occur for each arm, and will compute an optimal grasping and collision free motion automatically. For object hand-over, we only require the two arms to be able to *grip* the object at the same time, without collisions. The poses of the object in space during hand-over is completely determined by the optimization process.

V. Results

We demonstrate the efficacy of our framework in several scenarios. We refer the reader to the accompanying video for further results. The algorithm, as well as robot communication, was implemented in C++, and run on Windows 10 PC with Intel Core i7-9700. We used a UR5 or a YuMi® IRB 14000 with standard parallel grippers for simulation. To test with a physical robot, we use a YuMi, with thin cushioning attached to the grippers to avoid slippage, and blocks of dimensions 118 mm×23 mm×16 mm.

A. *Interactive planning*

In most scenarios the user is free to interactively modify pick-and-place locations, change the timing, and introduce and manipulate obstacles using a simple GUI. The optimization process runs continuously, so the user can inspect intermediate results in real-time. Once the convergence thresholds have reached, the motions can be submitted to the robot to perform. In Fig. 4 we show how the grippers react and continuously

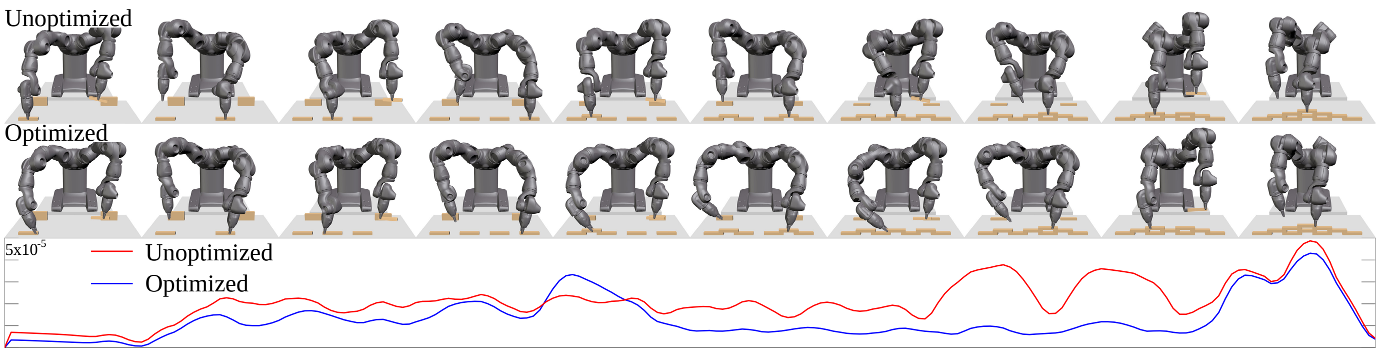


Fig. 5. Grasp optimized and unoptimized sequence for assembling a small pyramid of blocks. In both cases, the motion was optimized for low joint acceleration and velocity, but the top one had a fixed grasp location while the bottom one had it optimized as well. The graph shows the sum of squares of the joint velocities, which in the optimized case is indeed lower, and varies less.

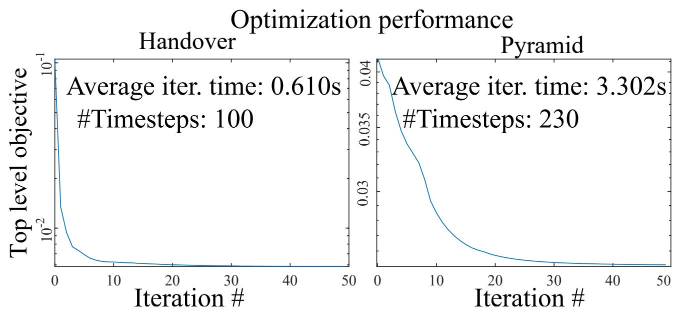


Fig. 6. Convergence plots showing the value of the top level objective per iteration, for the pyramid and the handover tasks.

change the grasping pose to avoid an obstacle controlled by the user. This and several more examples are shown in the video. In Fig. 6 we show the value of the top level objective and the progression of our solver per iteration, and the average time per iteration for the pyramid and hand-over experiments shown in Fig. 5 and 7 respectively. The graph show the typical convergence behaviour exhibited by second order optimization methods.

B. Smoothness and collisions

Fig. 5 and the video show the assembly of a small pyramid of blocks. The top sequence shows the sequence without grasp optimization, where the grip is predetermined to be in the middle of each block. As can be seen in the video, there is one collision that is unavoidable without changing the timing. Furthermore, although the motions are optimized for smoothness, without allowing the grasp to change, they cannot be as smooth as the grasp optimized motion. This is exemplified by the velocity graph in Fig. 5.

In Fig. 1 we show the assembly sequence of three side-by-side trusses, each made of two blocks leaning against each other. We note that such an assembly is only possible using two arms. The sequence shows that while the first truss can be assembled without any concerns, the second and third parts must be gripped in such a way to avoid collisions. We refer the reader to the video for the complete sequence.

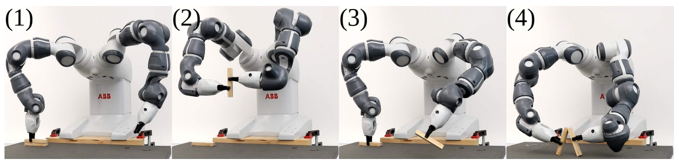


Fig. 7. Assembling a truss with a handover. (1) the blocks are positioned next to the right arm. (2) the right arm passes one block to the left arm. (3) it then picks up another block and (4) together they assemble a truss.

C. Handovers

Handovers are useful and sometimes necessary in many different settings. We demonstrate one case in Fig. 7 where the robot is tasked with making a truss, but one arm cannot reach the stack of blocks. In this case, the other arm first hands the block over to the first arm, and then they proceed to assemble the truss. Note that the handover configuration was not predetermined, but optimized using our system.

In the video we show another example where the task is to flip a block bottom-up, but only one arm can reach it. That arm cannot do it without colliding with the table. The only solution is to pick up the block, hand it over to the other arm, and grasp it in a different location. Then it can be placed back at its original location without difficulty.

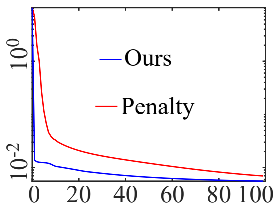
VI. Discussion

A. Alternative: Constrained optimization

We argue above that the multilevel formulation is a convenient way to describe grasp and motion planning problems. While initially multilevel problems seem like complicated constrained optimization problems, we show that they can be easily transformed into unconstrained problems, which are generally considered easier. Unconstrained optimization algorithms are also simpler to develop and are less demanding in term of optimality conditions. However, constrained optimization is a viable approach for grasp and motion planning, and we demonstrate it on a problem equivalent to (4), namely,

$$\begin{aligned} \min_{\mathbf{p},\mathbf{q}_{0n},\mathbf{Q}} \quad & \mathcal{O}_{\text{grasp}}(\mathbf{p},\mathbf{q}_{0n},\mathbf{Q}) \\ \text{s.t.} \quad & \nabla\mathcal{O}_{\text{pose}}(\mathbf{p},\mathbf{q}_{0n},\mathbf{Q}) = 0 \\ & \nabla\mathcal{O}_{\text{motion}}(\mathbf{p},\mathbf{q}_{0n},\mathbf{Q}) = 0. \end{aligned}$$

We can solve this problem using any constrained optimization algorithm. The penalty method for example is another method that converts the problem into an unconstrained problem, by squaring the constraints and adding them to the objective. We show a convergence plot comparing our method to the penalty method in the inset. In both cases we plot the objective plus the penalty terms. As shown, the behaviour is similar, but in our case, the penalty terms vanish in one iteration, resulting in the gap between the graphs. Another formulation used e.g. in [21] is to express $\mathbf{p}$ is a function of $\mathbf{Q}$ using forward kinematics.



B. limitations and future work

While our approach is rather flexible, we currently demonstrated it only for the simple longitudinal grasp space. There are other potential parameterizable grasp spaces for trivial objects, such as spherical or cylindrical objects, but the ultimate challenge would be to formulate a differentiable grasping model of arbitrary objects. One approach we explore is the use of an implicit representation of the object, e.g. a *distance function*, to derive a differentiable model. Similarly, our model only supports parallel grippers, but a follow up work would be to generalize it to other, more dexterous grippers, such as robotic hands or soft grippers. The smooth optimization framework we are employing can only find locally optimal motions. This is a general limitation of gradient-based methods, which can be alleviated using a sampling, multi-start strategy. In this paper, thanks to the simplicity of the model, we did not experience cases of clearly bad local minima. However, as the model becomes more complicated when generalized to other grippers and geometry, we expect bad local minima to become more prevalent. In terms of assemblies, we only treated the grasp and motion optimization part. However, a significant part of the problem is sequencing. Sequencing in general has a more discrete nature to it, which makes it challenging to integrate into our differentiable formulation. Some research avenues would be to attempt to solve a mixed-integer optimization problem, or to integrate the task and motion optimization problem into our framework, similar to the attempt in [22].

References

[1] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, April 2018.

[2] A. Sintov, O. Tslil, and S. Frenkel, "Longitudinal Regrasping of Elongated Objects," *Robotica*, pp. 1–13, May 2019.

[3] M. Graña, M. Alonso, and A. Izaguirre, "A panoramic survey on grasping research trends and topics," *Cybernetics and Systems*, vol. 50, no. 1, pp. 40–57, Jan. 2019.

[4] O. Kroemer, S. Niekum, and G. Konidaris, "A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms," *arXiv:1907.03146 [cs]*, July 2019, arXiv: 1907.03146.

[5] Y. Li, Q. Lei, C. Cheng, G. Zhang, W. Wang, and Z. Xu, "A review: machine learning on robotic grasping," in *Eleventh International Conference on Machine Vision (ICMV 2018)*, vol. 11041. International Society for Optics and Photonics, Mar. 2019, p. 110412U.

[6] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, 2019.

[7] G. Du, K. Wang, and S. Lian, "Vision-based Robotic Grasping from Object Localization, Pose Estimation, Grasp Detection to Motion Planning: A Review," *arXiv:1905.06658 [cs]*, May 2019.

[8] V. Babin and C. Gosselin, "A grasp manipulation selection chart to pick-up objects lying on hard surfaces," p. 5, May 2019.

[9] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *Int. J. Rob. Res.*, vol. 32, no. 9-10, pp. 1194–1227, Aug. 2013.

[10] J. A. Haustein, K. Hang, and D. Kragic, "Integrating motion and hierarchical fingertip grasp planning," in *2017 IEEE International Conference on Robotics and Automation*, May 2017, pp. 3439–3446.

[11] K. Hang, M. Li, J. A. Stork, Y. Bekiroglu, F. T. Pokorny, A. Billard, and D. Kragic, "Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation," *IEEE Trans. Robotics*, vol. 32, no. 4, pp. 960–972, 2016.

[12] J. Huh, B. Lee, and D. D. Lee, "Constrained sampling-based planning for grasping and manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 223–230.

[13] A. M. Ghalamzan E., N. Mavrakis, M. S. Kopicki, R. Stolkin, and A. Leonardis, "Task-relevant grasp selection: A joint solution to planning grasps and manipulative motion trajectories," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14*, 2016, pp. 907–914.

[14] W. Wan, H. Igawa, K. Harada, H. Onda, K. Nagata, and N. Yamanobe, "A regrasp planning component for object reorientation," *Autonomous Robots*, vol. 43, no. 5, pp. 1101–1115, June 2019.

[15] R. Moriyama, W. Wan, and K. Harada, "Dual-arm Assembly Planning Considering Gravitational Constraints," *arXiv:1903.00646*, Mar. 2019.

[16] W. Wan, K. Harada, and F. Kanehiro, "Planning grasps for assembly tasks," *CoRR*, vol. abs/1903.01631, 2019.

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

[18] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 6231–6235.

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

[20] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, p. 1930–1936.

[21] M. Toussaint, "Newton methods for k-order markov constrained motion problems," *CoRR*, vol. abs/1407.0414, 2014.

[22] C. Zhang and J. A. Shah, "Co-optimizing task and motion planning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4750–4756.

[23] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[24] O. Kanoun, F. Lamiraux, and P. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, Aug 2011.

[25] S. Cruciani and C. Smith, "Integrating path planning and pivoting," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 6601–6608.

[26] C. H. Kim and J. Seo, "Shallow-Depth Insertion: Peg in Shallow Hole Through Robotic In-Hand Manipulation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 383–390, Apr. 2019.

[27] J.-F. M. Barthelemy and J. Sobieszczanski-Sobieski, "Optimum sensitivity derivatives of objective functions in nonlinear programming," *AIAA Journal*, vol. 21, pp. 913–915, June 1983.

[28] R. H. Jackson and G. P. Mccormick, "Second-order sensitivity analysis in factorable programming: Theory and applications," *Math. Program.*, vol. 41, no. 1-3, pp. 1–27, May 1988.

[29] S. Zimmermann, R. Poranne, and S. Coros, "Optimal control via second order sensitivity analysis," *arXiv preprint arXiv:1905.08534*, 2019.

[30] S. Duenser, J. M. Bern, R. Poranne, and S. Coros, "Interactive Robotic Manipulation of Elastic Objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 3476–3481.