Proceedings Paper:
Chen, L, Figueredo, LFC orcid.org/0000-0002-0759-3000 and Dogar, M orcid.org/0000-0002-6896-5461 (2020) Manipulation Planning Using Environmental Contacts to Keep Objects Stable under External Forces. In: Proceedings of 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids). 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), 15-17 Oct 2019, Toronto, ON, Canada. IEEE . ISBN 978-1-5386-7630-1

https://doi.org/10.1109/Humanoids43949.2019.9034998

Manipulation Planning Using Environmental Contacts to Keep Objects Stable under External Forces

Lipeng Chen, Luis F. C. Figueredo, Mehmet Dogar

***Abstract*— This paper addresses the problem of sequential manipulation planning to keep an object stable under changing external forces. Particularly, we focus on using object-environment contacts. We present a planning algorithm which can generate robot configurations and motions to intelligently use object-environment, as well as object-robot, contacts, to keep an object stable under forceful operations such as drilling and cutting. Given a sequence of external forces, the planner minimizes the number of different configurations used to keep the object stable. An important computational bottleneck in this algorithm is due to the static stability analysis of a large number of configurations. We propose a containment relationship between configurations, to prune the stability checking process.**

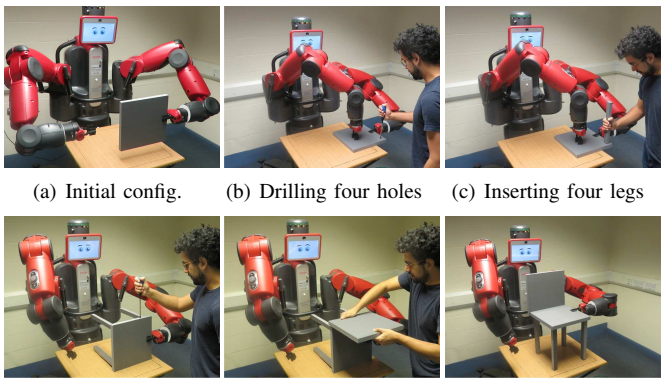
I. Introduction

We propose an approach to address manipulation planning of keeping an object stable under a sequence of changing external forces. Particularly, we focus on using object-environment contacts, in addition to using robot grasps. Take the example in Fig. 1, where a human and a robot collaborate to assemble a chair. Throughout the assembly task, the human applies changing forces on the chair pieces, due to a sequence of drilling and inserting operations. The robot is supposed to assist the human by keeping the chair assembly stable as these forceful operations are applied. The robot itself cannot to hold the chair assembly stable against all the different and large external forces. Nevertheless, by exploiting object contacts with the environment (e.g. the table surface in Fig. 1), as well as object contacts with the robot (e.g. grasps, or griper pressing as shown in Fig. 1(b)-1(c)), the robot succeeds in stabilizing the object under the sequential changing forces.

The problem of regrasp planning with environmental support/contact has been well explored before [1], [2], [3], [4], [5], while planning to resist changing external forces applied onto an object using environment contacts remains to be studied, to the best of our knowledge. In our previous work [6], we describe a robot grasping an object to resist external forces but *without* using the object contacts with the environment, which is our focus in this current work. Motivated by the potential of exploiting environmental contacts, for example in a human-robot collaborative task as shown in Fig. 1, we propose a novel planning approach that enables the robot to use both the environment's and robot's stabilization capabilities in manipulating an object under changing external forces and integrate them with intelligent motion planning.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grants agreement No. 746143 and 795714, and from the UK Engineering and Physical Sciences Research Council under grant EP/P019560/1.

Authors are with School of Computing, University of Leeds, Leeds, UK, {sclc, l.figueredo, m.r.dogar}@leeds.ac.uk



(a) Initial config. (b) Drilling four holes (c) Inserting four legs

(d) Driling four holes (e) Inserting chair back (f) Task complete

Fig. 1. The robot holds the chair pieces stable under sequential force-demanding operations in an assembly task, by exploiting object contacts with both environment (e.g. a table surface) and robot (pressing or grasping).

To achieve robust and efficient manipulation, our planner addresses three main challenges. First, the planner identifies appropriate object contacts with the environment and/or the robot, relying on which the object can stay stable under forceful operations. This requires the planner to determine *where* to move and position the object w.r.t. the environment and robot. For example, at the configuration in Fig. 1(b), the object-table contact and object-robot contact (the gripper pressing) can together stabilize the chair piece under the drilling forces. Second, to improve manipulation efficiency, the planner reasons over a large number of available contact configurations for an optimal solution, which requires minimal number of contact adjustments and thus robot motions during manipulation. For example, in Fig. 1, in total, there are only two different configurations (Fig. 1(b) and 1(d)) and thus one contact change during assembling. This requires the planner to decide on *when* to change the object pose w.r.t. the environment and robot as the operations are applied. Finally, for each change in the object contacts, the planner needs to decide on *how* to move the object to implement the change, e.g. from the one in Fig. 1(c) to the one in Fig. 1(d).

To address the challenges outlined above, we introduce a hierarchical planner. The planner first searches a sequence of contact configurations (the robot configuration and object pose in the environment) which can stabilize the object under external forces. The planner minimizes the number of different configurations in the sequence, so that the robot can move the object as minimally as possible throughout the task. Then, it connects these configurations with motion trajectories. We describe details of the planner in Sec. III.

A key source of computational cost of the planner comes from stability checking: the planner samples a large set of candidate configurations, with a wide variety of object-environment and object-robot contacts. Each such configuration needs to be checked for stability against possible external forces. Given frictional constraints at the contact points, such a stability checking takes the form of a constrained optimization problem, a computationally expensive process.

Therefore, as another contribution, we propose a novel strategy to efficiently perform stability checking of a large number of configurations. We introduce a concept of *containment* among contact configurations. Given two contact configuration, based on their object contacts with the environment and robot, we say one configuration *contains* the other, if the stability of the latter against an external force implies the stability of the former. For instance, a bimanual grasp *contains* its both individual grasps. By exploiting this concept, the planner can easily compare the stabilization capabilities among different contact configurations and thus quickly prune the list of candidates. We provide details of the containment-based stability checking in Sec. IV.

We show the performance of our planner and the containment-based stability checking in Sec. V. We show our planner can produce effective manipulation plans with reduced number of configuration changes in comparison with baseline planners. We present the containment-based stability checking brings in almost an order of magnitude improvement in planning speed compared with a naive approach.

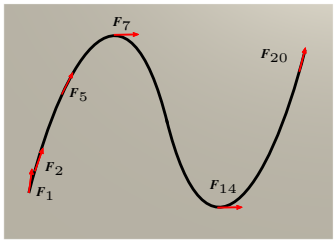
A. Previous Work

The idea of exploiting structures in the environment via physical contacts can be traced in the earliest approaches for compliant fine motion generation [7]. Evidence of complex interaction between multi-step manipulation planning problems and a shared environment is also found in [1], [2], which addressed structures in the shared environment as obstacles to be avoided while manipulating target objects.

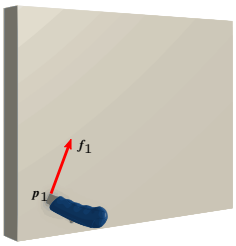
Recently, planners have been proposed to use structures in the shared environment as extra supports for various manipulation tasks like grasping [8], [9] regrasping [3], [5], prehensile [10], [11] and non-prehensile manipulation [12]. A similar multi-step planning problem is studied by Bretl [13], to produce contact modes for a spider robot climbing a wall. We use a similar hierarchical approach, but focus on the problem where there are changing external forces applied onto the manipulated objects.

Our novel contributions in this work include:

1) A planning framework that exploits both the object-environment contacts and the object-robot contacts to resist changing external forces with a minimal number of configuration changes—ensuring a more smooth and efficient manipulation;
2) The formulation of the containment relations among different contact configurations to perform efficient static stability checking;
3) Real robot and simulated experiments illustrating the effectiveness and efficiency of the planning framework.



A forceful task example: $\{\boldsymbol{F}_i\}_{i=1}^{20}$



A forceful operation example: $\boldsymbol{F}_1$

Fig. 2. A forceful task consists of a sequence of forceful operations.

II. Problem Formulation

We are interested in building a planner that enables a robot to efficiently manipulate an object under the application of sequential or changing forceful operations, e.g. drilling, cutting and peg inserting.

We refer to a forceful *task* as a complete sequence of operations to be applied onto an object, for example, the chair assembly task in Fig. 1. We give another example in Fig. 2, where the task is cutting a wave-shaped curve on a board. We refer to a forceful *operation* as a single step of a forceful task. For example, in Fig. 1, each involved assembling action (e.g. drilling) is a separate forceful operation. For tasks consisting of a continuous application of forces over time, we divide into a sequence of operations via discretization, e.g. the 20 discrete cutting operations in Fig. 2.

We model a forceful operation $\boldsymbol{F}$ as a 6D *generalized* force (force/torque) $\boldsymbol{f}$ w.r.t. a tool frame and its application pose $\boldsymbol{p}$ w.r.t. the target object (Fig. 2-Right), and thus denote a forceful task as:

$$\{\boldsymbol{F}_i\}_{i=1}^{m} = \{(\boldsymbol{f}_i, \boldsymbol{p}_i)\}_{i=1}^{m} \tag{1}$$

where m indicates the number of involved operations.

We formulate the problem by explicitly exploiting the object contacts with both the environment and robot. Specifically, herein we assume the robot can contact an object by *pressing* (e.g. the robot's right hand in Fig. 1(b) and 1(c)) or *grasping* (e.g. the robot's left hand in Fig. 1), while the environment includes rigid structures which allow an object to be in contact with, e.g. the table surface in Fig. 1. During a forceful task, the environmental and robot contacts together provide supporting wrenches to manipulate and stabilize the object under forceful operations.

Throughout this work, we assume the forceful task and the geometric model of the system are given. Other physical parameters, including object mass, centre of gravity and friction coefficients are specified. We denote the composite system configuration as $\boldsymbol{q} = (\boldsymbol{q}_\mathrm{r}, \boldsymbol{q}_\mathrm{o})$, where $\boldsymbol{q}_\mathrm{r} \in \mathbb{R}^{n_\mathrm{r}}$ denotes configuration of robot manipulators (n_r is the total DoFs of manipulators) and $\boldsymbol{q}_\mathrm{o} \in \mathbb{SE}(3)$ denotes object pose. A system configuration $\boldsymbol{q}$ specifies a set of environmental and/or robot contacts onto the target object, by which the environment and robot generate reaction forces capable of moving and stabilizing the object against gravity and external forces.

A. Overview of Problem

As illustrated in Fig. 3, the robot is supposed to stabilize an object under a sequence of forceful operations using object contacts with the environment and the robot grippers.

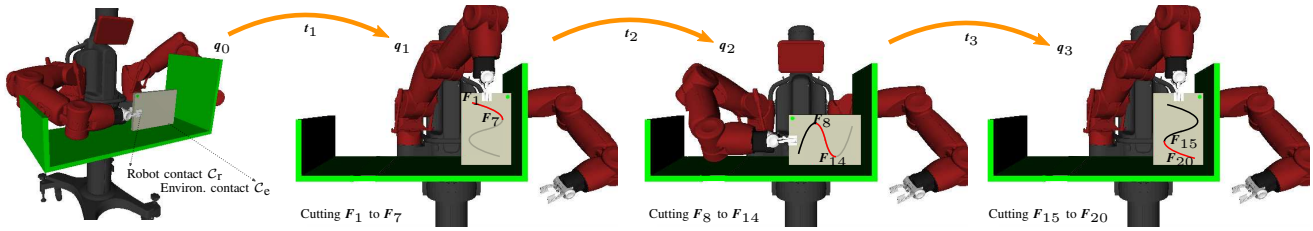


Fig. 3. The planner generates a minimum sequence of system configurations $\{\boldsymbol{q}_j\}_{j=1}^{3}$ to keep the object stable under $\{\boldsymbol{F}_i\}_{i=1}^{20}$ and a sequence of trajectories $\{\boldsymbol{t}_j\}_{j=1}^{3}$ to move the object to go through the planned configurations.

Given a *single* forceful operation $\boldsymbol{F}$, finding a system configuration $\boldsymbol{q}$ that can keep the object stable requires 1) first finding a kinematically valid configuration $\boldsymbol{q}$; 2) then checking if the environment and robot can provide sufficient resultant wrenches via physical contacts to keep the object stable. We refer to this problem as *stability checking* and explain in detail how we address it in Sec. IV-A.

Then, given *a sequence of* forceful operations $\{\boldsymbol{F}_i\}_{i=1}^{m}$, i.e. a complete task, the most straightforward solution would be to find one feasible configuration $\boldsymbol{q}$ for a given operation $\boldsymbol{F}_i$ and hold such configuration until it is no longer stable for a new incoming one $\boldsymbol{F}_{i^+}$, where $i^+ > i$. However, such a strategy, would require the robot and object to use a large sequence of different system configurations and accordingly configuration transfers during manipulation. Herein, we refer to such a configuration transfer as a *configuration change*.

Alternatively, to avoid frequent task interruptions due to configuration changes, the robot can reason about the system configurations to make the utmost of one configuration $\boldsymbol{q}$ against a larger sequence of forceful operations, which in turn would reduce the number of configuration changes. This, *configuration change minimization*, imposes an additional but practically necessary requirement for efficient and smooth manipulation. In this work, we explicitly address this as a main objective, building a planning framework that minimizes the sequence of system configurations while satisfying the sequential forceful constraints. In Sec. III, we introduce a planner that generates such efficient sequences based on the assessment of a set of stable system configurations, while in Sec. IV, we present how this assessment can be performed efficiently by exploiting their *contact-containment* relations.

B. Definition of Sequence of Stable Configurations

We say a system configuration $\boldsymbol{q}$ is *stable against* a sequence of k forceful operations $\{\boldsymbol{F}_i\}_{i=1}^{k}$, if the environmental and robot contacts at the configuration $\boldsymbol{q}$ are able to provide sufficient wrenches to keep the object stable under any forceful operation in $\{\boldsymbol{F}_i\}_{i=1}^{k}$.

Further, we say that a sequence of system configurations $\{\boldsymbol{q}_j\}_{j=1}^{n}$ is stable against a sequence of forceful operations $\{\boldsymbol{F}_i\}_{i=1}^{m}$, if the configurations in $\{\boldsymbol{q}_j\}_{j=1}^{n}$ cover all operations in $\{\boldsymbol{F}_i\}_{i=1}^{m}$ in order, i.e., if $\boldsymbol{q}_1$ is stable against operations $\{\boldsymbol{F}_1, \boldsymbol{F}_2, ..., \boldsymbol{F}_k\}$, and $\boldsymbol{q}_2$ is stable against operations $\{\boldsymbol{F}_{k+1}, \boldsymbol{F}_{k+2}, ..., \boldsymbol{F}_l\}$, and so on, until $\boldsymbol{q}_n$ is stable against operations $\{\boldsymbol{F}_{w+1}, \boldsymbol{F}_{w+2}, ..., \boldsymbol{F}_m\}$, where $1 \leq k \leq l \leq ... \leq w \leq m$. For example, in Fig. 3, the three system configurations $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3\}$ cover 20 cutting operation: $\boldsymbol{q}_1$ is stable against the cutting operations $\boldsymbol{F}_1$ to $\boldsymbol{F}_7$; $\boldsymbol{q}_2$ is stable against the cutting operations $\boldsymbol{F}_8$ to $\boldsymbol{F}_{14}$; $\boldsymbol{q}_3$ is stable against the cutting operations $\boldsymbol{F}_{15}$ to $\boldsymbol{F}_{20}$. In this sense, configuration change minimization can be achieved by finding a minimal sequence of configurations $\{\boldsymbol{q}_j\}_{j=1}^{n}$ stable against $\{\boldsymbol{F}_i\}_{i=1}^{m}$.

In addition to minimizing the number of configuration changes, the robot also needs to move the object to go through the planned configurations in $\{\boldsymbol{q}_j\}_{j=1}^{n}$ in order, using collision-free and stable trajectories $\{\boldsymbol{t}_j\}_{j=1}^{n}$. Specifically, each trajectory $\boldsymbol{t}_j$ moves the system from $\boldsymbol{q}_{j-1}$ to $\boldsymbol{q}_j$ ($\boldsymbol{q}_0$ is the initial system configuration), which corresponds to a constrained motion planning. For example, in Fig. 3, the three orange lines illustrate such trajectories $\{\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3\}$.

In this context, given a sequence of forceful operations $\{\boldsymbol{F}_i\}_{i=1}^{m}$ and an initial system configuration $\boldsymbol{q}_0$, our problem can be stated as finding a minimum sequence of system/contact configurations $\{\boldsymbol{q}_j\}_{j=1}^{n}$ and its connecting trajectories $\{\boldsymbol{t}_j\}_{j=1}^{n}$ to move and stabilize the object under the application of sequential forceful operations $\{\boldsymbol{F}_i\}_{i=1}^{m}$.

III. Planning Approach

This section presents details of our planning framework.

A. Manipulation Planning Using Operation Graph

Our planner takes a hierarchical framework. At the high-level it builds and searches what we call an *operation graph* for a minimal sequence of configurations $\{\boldsymbol{q}_j\}_{j=1}^{n}$ that are stable against $\{\boldsymbol{F}_i\}_{i=1}^{m}$. It then generates motion trajectories $\{\boldsymbol{t}_j\}_{j=1}^{n}$ to connect these planned configurations. We provide pseudo-code of the planner in Alg. 1.

1) Building the Operation Graph: The planner starts by building a directed acyclic weighted graph, referred to as the operation graph, using checked-stable configurations from among a set of sampled candidate configurations Q_s.

Specifically, as illustrated in Fig. 4, in an operation graph, the i_{th} column corresponds to the i_{th} operation $\boldsymbol{F}_i$, while all nodes in the i_{th} column represent a subset of sampled configurations in Q_s, which are checked stable against $\boldsymbol{F}_i$ (We present how this check is performed in Sec. IV-A).

We further define a link between every two nodes in neighbouring columns of the graph and weight the link with a weighting scheme associated with the number of configuration changes.

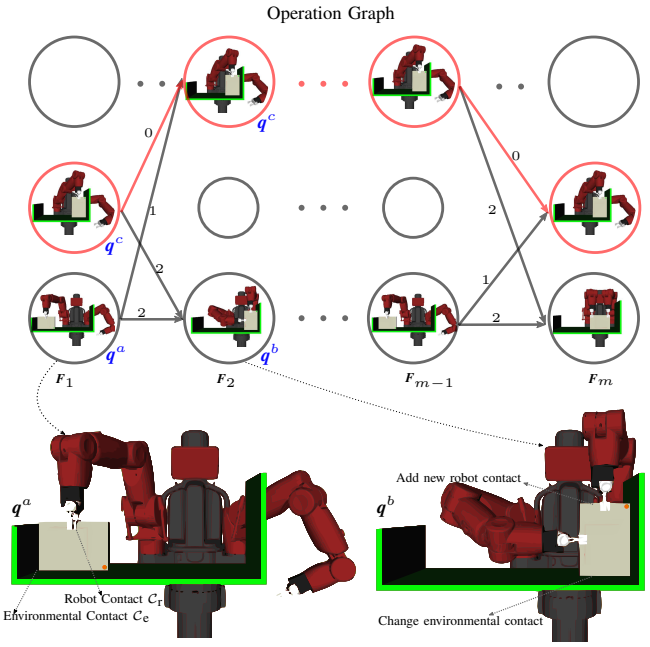


Fig. 4. We build an operation graph to search for efficient solutions.

2) Weighting Links in Operation Graph: As illustrated in Fig 4, rather than weighting the link of two different configurations (e.g., from $\boldsymbol{q}^a$ to $\boldsymbol{q}^b$) as 1, here we adopt a more detailed weighting scheme, computing the weight as the number of the total changes in both environmental and robot contacts. For example, in Fig. 4, from the node $\boldsymbol{q}^a$ in the first column to the node $\boldsymbol{q}^b$ in the second column, we say the number of configuration changes is 2, including one robot contact change (one extra gripper is added onto the object) and one environmental contact change (the contact region changes). However, the weight between the node $\boldsymbol{q}^c$ in the first column and the node $\boldsymbol{q}^c$ in the second column is set to be 0, since they simply represent the same configuration.

3) Searching the Operation Graph: At this point, using the operation graph, finding a minimal sequence of system configurations $\{\boldsymbol{q}_j\}_{j=1}^n$ stable against $\{\boldsymbol{F}_i\}_{i=1}^m$ is reformulated as a graph search problem. The expected output is a path that starts from one node in the leftmost column for operation $\boldsymbol{F}_1$ and ends with a node in the rightmost column for operation $\boldsymbol{F}_m$, with the smallest total weight. By searching the graph, e.g., using the Dijkstra's algorithm, the planner can easily get optimal solutions, i.e. efficient manipulation plans with a minimal number of configuration changes. In Fig. 4, the red path illustrates such a solution.

The procedure PlanStableSequence in Alg. 1 provides the pseudo-code of planning using the operation graph. The procedure BuildOperationGraph builds the operation graph (line 2) as outlined above. The procedure GraphSearch (line 3) searches the operation graph to generate a candidate configuration sequence $\{\boldsymbol{q}_j\}_{j=1}^n$. Then for every two subsequent pair of configurations in the sequence (line 5 - 9), the procedure PlanConfigChange attempts to plan motions via general motion planners, e.g. RRT-based planners, to implement the configuration changes.

4) Finding Stable Configurations for An Operation: As described above, building an operation graph requires the planner to check and find a subset of stable configurations for each forceful operation in $\{\boldsymbol{F}_i\}_{i=1}^m$, from a set of sampled candidate configurations Q_s. This is achieved by FindStableConfigs in Alg. 1 (line 3 in BuildOperationGraph).

Algorithm 1 Manipulation Planning Using Operation Graph

PlanStableSequence $\left(\{\boldsymbol{F}_i\}_{i=1}^m, \boldsymbol{q}_0\right)$:
1: $Q_s \leftarrow$ Generate a set of candidate system configurations
2: $\mathcal{G}_O \leftarrow$ BuildOperationGraph $\left(\{\boldsymbol{F}_i\}_{i=1}^m, Q_s\right)$
3: $\{\boldsymbol{q}_j\}_{j=1}^n \leftarrow$ GraphSearch $(\mathcal{G}_O)$
4: $\{\boldsymbol{q}_j\}_{j=0}^n \leftarrow$ Add $\boldsymbol{q}_0$ to the beginning of $\{\boldsymbol{q}_j\}_{j=1}^n$
5: **for** each subsequent $\boldsymbol{q}_j$ and $\boldsymbol{q}_{j+1}$ in $\{\boldsymbol{q}_j\}_{j=0}^n$ **do**
6: $\quad \boldsymbol{t}_{j+1} \leftarrow$ PlanConfigChange$(\boldsymbol{q}_j, \boldsymbol{q}_{j+1})$
7: $\quad$ **if** PlanConfigChange failed **then**
8: $\quad\quad$ Remove failing edge from graph $\mathcal{G}_O$
9: $\quad\quad$ Go to line 3
10: **return** $(\{\boldsymbol{q}_j\}_{j=1}^n, \{\boldsymbol{t}_j\}_{j=1}^n)$

BuildOperationGraph $\left(\{\boldsymbol{F}_i\}_{i=1}^m, Q_s\right)$:
1: $\mathcal{G}_O \leftarrow \emptyset$
2: **for** each forceful operation $\boldsymbol{F}_i$ in $\{\boldsymbol{F}_i\}_{i=1}^m$ **do**
3: $\quad S \leftarrow$ FindStableConfigs$(\boldsymbol{F}_i, Q_s)$
4: $\quad$ **if** $i = 1$ **then**
5: $\quad\quad$ Add S into $\mathcal{G}_O$ as the first column
6: $\quad$ **else**
7: $\quad\quad$ **for** each configuration $\boldsymbol{q}'$ in previous column of $\mathcal{G}_O$ **do**
8: $\quad\quad\quad$ **for** each configuration $\boldsymbol{q}''$ in S **do**
9: $\quad\quad\quad\quad w \leftarrow$ ComputeWeight $(\boldsymbol{q}', \boldsymbol{q}'')$
10: $\quad\quad\quad\quad$ Create a link from $\boldsymbol{q}'$ to $\boldsymbol{q}''$ with a weight w
11: **return** $\mathcal{G}_O$

The planner starts from sampling a set of candidate configurations Q_s, which acts as a representative to the high-dimensional composite configuration space (line 1 in PlanStableSequence). The set Q_s includes a variety of configurations with different object-environment and object-robot contacts. Given an object and an environment model, the problem of contact generation with the environment has been extensively studied in the literature based on geometric computation [14], [15], learning [16] and kinematics simulators [17], [5]. Likewise, robot-object contacts can be computed via general grasp planners, e.g. Miller and Allen [18]. Such techniques lie outside the scope of this work. Our planner is, in fact, agnostic to the contact generation strategy and thus can take any existing method in the literature for this step. In Sec. V, we explain how we generated such sets of candidate configurations for our experimental studies.

Note that, to find efficient manipulation plans that minimize configuration changes, the planner needs a large set of candidate configurations Q_s, which, however, makes the procedure FindStableConfigs and as a result the procedure BuildOperationGraph computationally expensive. This is because, given an operation $\boldsymbol{F}$, the procedure requires the planner to perform a separate stability checking for each sampled configuration $\boldsymbol{q} \in Q_s$, while the stability checking of a single configuration itself is already a computationally expensive process (as explained in Sec. IV-A). Therefore, after presenting a naive approach in Sec. IV-B, we present, in Sec. IV-C, a containment-based strategy to implement the procedure FindStableConfigs efficiently.

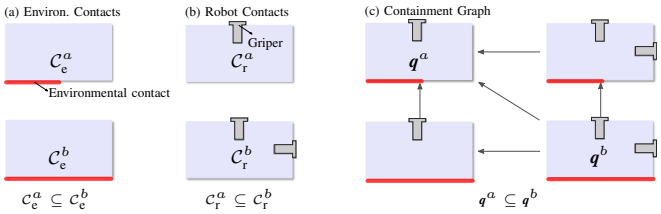


Fig. 5. We build a containment graph over all system configurations in Q_s to represent their containments. The red segments illustrate environmental contacts with the object.

IV. Stability Checking

In this section, we formulate the stability checking of a system configuration $\boldsymbol{q}$ against a forceful operation $\boldsymbol{F}$ as an optimization problem. Then, we present an approach that can efficiently find all configurations in Q_s that are stable against $\boldsymbol{F}$, i.e. the procedure FindStableConfigs in Alg. 1.

A. Stability Checking for A Single Forceful Operation

Stability checking refers to the problem of assessing if a system configuration $\boldsymbol{q}$ is able to provide sufficient wrenches to stabilize an object under a forceful operation $\boldsymbol{F}$.

As illustrated in Fig 5, under the assumption of rigid bodies, both environmental and robot contacts specify contact regions on the object surface (denoted as $\mathcal{C}_e$ and $\mathcal{C}_r$ respectively), which can be further represented as a set of point contacts. At each contact point, the contact force $\boldsymbol{f}_c$ applied to the object is constrained within a generalized friction cone $\boldsymbol{f}_c \in FC(\boldsymbol{n}, \mu)$ characterized by a contact normal $\boldsymbol{n}$ and a friction coefficient μ.

Given a forceful operation $\boldsymbol{F}$=($\boldsymbol{f}$, $\boldsymbol{p}$), and a configuration $\boldsymbol{q}$ consisting of n_c point contacts in total (including the uniformly discretized surfaces of both the environmental contacts and robot contacts), stability checking requires finding a distribution of contact forces $\boldsymbol{h} = [\boldsymbol{f}_{c,1}^{\mathsf{T}}, \boldsymbol{f}_{c,2}^{\mathsf{T}}, ..., \boldsymbol{f}_{c,n_c}^{\mathsf{T}}]^{\mathsf{T}}$, such that:

1) The resultant wrench provided by $\boldsymbol{h}$ is able to keep the object stable, i.e. in static equilibrium, against the external force $\boldsymbol{f}$ by the operation $\boldsymbol{F}$;
2) The contact forces $\boldsymbol{h}$ lie within the composite friction cone H=$FC_1 \times ... \times FC_{n_c}$.

Therefore, similar to [19], [20], [21], we formulate the problem of stability checking as a constrained optimization problem, particularly a quadratic programming:

$$\min_{\boldsymbol{h} \in \mathrm{H}} \quad \boldsymbol{h}^{\mathrm{T}}\boldsymbol{h} \tag{2a}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{h} + \boldsymbol{h}_{mg} + R(\boldsymbol{p})\boldsymbol{f} = 0 \tag{2b}$$

$$\boldsymbol{h} \in \mathrm{H} \tag{2c}$$

where $\boldsymbol{G}$=$[\boldsymbol{G}_1, ..., \boldsymbol{G}_{n_p}]$ is a $6 \times 6n_c$ matrix mapping the contact wrenches to a resultant wrench onto the object. $\boldsymbol{h}_{mg}$ is the wrench incurred by object gravity. $R(\boldsymbol{p})$ transforms the external wrench $\boldsymbol{f}$ from the tool frame to a common frame.

For a candidate configuration $\boldsymbol{q}$ and an operation $\boldsymbol{F}$, if we can find a solution $\boldsymbol{h}$ satisfying the constraints in Eq. 2, we say that the configuration $\boldsymbol{q}$ is able to stabilize the object against the operation $\boldsymbol{F}$. Otherwise, we say that the configuration $\boldsymbol{q}$ cannot be stable against the operation $\boldsymbol{F}$. The constraints Eq. 2(b)-2(c) ensure that there exists a distribution of contact forces $\boldsymbol{h}$ that can keep the object stable under the operation $\boldsymbol{F}$, while optimizing the objective in Eq. 2(a) ensures $\boldsymbol{h}$ to be close to the actual force distribution [21].

Algorithm 2 Naive Stability Checking

FindStableConfigs($\boldsymbol{F}, Q_s$):
1: $S = \emptyset$
2: **for** each configuration $\boldsymbol{q}$ in Q_s **do**
3: Solve the constrained optimization problem in Eq. 2
4: **if** Eq. 2 has a solution **then**
5: Add $\boldsymbol{q}$ to S
6: **return** S

In the following sections, we present two approaches to implement the procedure FindStableConfigs in Alg. 1, which performs stability checkings over a large set of sampled configurations in Q_s.

B. Naive Stability Checking of Q_s

Eq. 2 enables the planner to check whether a configuration $\boldsymbol{q}$ is stable against a forceful operation $\boldsymbol{F}$. Accordingly, given a forceful operation $\boldsymbol{F}$, the planner can implement the procedure FindStableConfigs by simply performing a separate stability checking via optimizing in Eq. 2, for each candidate configuration in Q_s.

We call this implementation the *naive stability checking*. Alg. 2 provides the pseudo-code of this implementation.

Nonetheless, as aforementioned, due to the large set Q_s, this implementation can be computationally expensive, and therefore degrade the planing efficiency, as we show with experimental results in Sec. V-B.

C. Containment-Based Stability Checking of Q_s

Rather than performing excessive stability checkings by solving optimization problems in Eq. 2 over all configurations in Q_s, we propose a *containment-based* approach to implement the procedure FindStableConfigs efficiently. The algorithm relies on the *containment relationship* among different configurations to quickly eliminate redundant stability checkings. Alg. 3 presents the pseudo-code of this strategy.

1) Containment Relationship: We define the containment relationship between two different system configurations over their capability of resisting external wrenches, i.e. contact wrench space [22], [23]. A system configuration $\boldsymbol{q}$ specifies the geometric relationships among the object, robot and environment, while its corresponding contact region(s) qualitatively indicates its capability of resisting external forces that can be applied on the object. For example in Fig. 5[1], since the environmental contact region $\mathcal{C}_e^b$ contains $\mathcal{C}_e^a$ and the robot contact $\mathcal{C}_r^b$ contains $\mathcal{C}_r^a$, one can say any forceful operation resistible to $\boldsymbol{q}^a$ is also resistible to $\boldsymbol{q}^b$. That is, $\boldsymbol{q}^b$ *contains* $\boldsymbol{q}^a$ in terms of the capability of resisting possible external wrenches. In this context, we define the containment relationship (denoted with $\subseteq$) among different contact configurations as:

Definition 1. *Let $\boldsymbol{q}^a$ and $\boldsymbol{q}^b$ be two system configurations, we say $\boldsymbol{q}^b$ contains $\boldsymbol{q}^a$ ($\boldsymbol{q}^a \subseteq \boldsymbol{q}^b$), iff*

[1]Fig. 5 is illustrated in 2D for clarity of presentation. In our implementation, contact regions and containment are represented in 3D.

- $\mathcal{C}_e^a \subseteq \mathcal{C}_e^b$ *and*
- $\mathcal{C}_r^a \subseteq \mathcal{C}_r^b$

Note that the containment among the environmental contact regions holds only for the same surface contacts, as the friction coefficients may be different for different materials.

2) Containment Graph: We then build an directed acyclic containment graph over all sampled configurations in Q_s to represent their containments (line 2 in Alg. 3).

Specifically, as illustrated in Fig. 5(c), each node in the graph represents a candidate configuration $\boldsymbol{q}$ in Q_s. For every two nodes, if there exists a containment between them as stated in Def. 1, they are connected in the graph with a directed path, e.g. in Fig. 5(c), the node $\boldsymbol{q}^a$ is contained and thus becomes a successor to the other three nodes.

3) Stability Checking Using Containment Graph: Based on the characteristics of the containment graph, we introduce two properties to simplify the procedure FindStableConfigs.

Property 1. *Given a forceful operation $\boldsymbol{F}$ and a containment graph $\mathcal{T}$, if a node $\boldsymbol{q}$ is stable against $\boldsymbol{F}$, then all its predecessors in $\mathcal{T}$ are stable against the operation $\boldsymbol{F}$.*

The property can be easily proved by Def. 1. We call this property the '*activating*' property, as using this property, if a configuration $\boldsymbol{q}$ is checked to be stable against a forceful operation $\boldsymbol{F}$, all its predecessor configurations can be directly 'activated' as feasible without solving optimization problems in Eq. 2 (line 6-9 in Alg. 3). For example, if $\boldsymbol{q}^a$ in Fig. 5(c) is stable against an operation $\boldsymbol{F}$, then all other nodes in the graph can be directly regarded as stable against $\boldsymbol{F}$.

Property 2. *Given a forceful operation $\boldsymbol{F}$ and a containment graph $\mathcal{T}$, if a node $\boldsymbol{q}$ is not stable against $\boldsymbol{F}$, then all its successors in $\mathcal{T}$ are not stable against the operation $\boldsymbol{F}$.*

Similarly, the property can be easily proved by Def. 1. We call this property the '*blocking*' property, as using it, if a configuration $\boldsymbol{q}$ is checked as not stable against a forceful operation $\boldsymbol{F}$, all its successor configurations can be directly 'blocked' (line 10-12 in Alg. 3) from searching. For instance in Fig. 5(c), if $\boldsymbol{q}^b$ is not able to keep the object against an operation $\boldsymbol{F}$, then all other nodes in the graph can be directly regarded as infeasible without additional checking.

By exploiting the activating property and blocking property, the planner needs not to solve a constrained optimization problem for each configuration in Q_s separately, thus reducing computational complexity of the procedure FindStableConfigs greatly. We proposed experiments to verify the effectiveness of using the containment graph in Sec. V-B.

Algorithm 3 Containment-Based Stability Checking

```
FindStableConfigs(F, Q_s):
1:  S ← ∅
2:  T ← Build a containment graph as Fig. 5
3:  while T is not empty do
4:      q ← Randomly pick a configuration in T
5:      Solve the constrained optimization problem in Eq. 2
6:      if Eq. 2 has a solution then
7:          S ← Add q and its predecessors in T into S
8:          T ← Remove q and its predecessors from T
9:          go to line 3
10:     else
11:         T ← Remove q and its successors from T
12:         Go to line 3
13: return S
```

V. Implementation and Results

This section presents a variety of simulated and real robot experiments, using a Baxter robot in both cases, to validate and quantitatively assess the performance of our planner.

Experimental Settings: We implemented the planner in OpenRAVE [24] with the flexible collision library (FCL) [17] for collision check and contact detection. We used Scipy.optimize library for the optimization based stability checking (Eq. 2), NetworkX [25] for graph construction and search, and BiRRT [26] as the motion planner for the procedure PlanConfigChange in Alg. 1.

We implemented the planner on three types of forceful operations. To obtain a more realistic representation, we implemented these operations multiple times on a foam board used as the target object, and measured the operation forces via a force/torque sensor (FT150 from Robotiq). For each type of operations, we took a maximum operation force:

- A *drilling* operation yields a maximum $20\,N$ force;
- A *cutting* operation yields a maximum $45\,N$ force;
- An *inserting* operation yields a maximum $16\,N$ force.

Generating a Set of Candidate Configurations Q_s: We sampled three sets of candidate system configurations (line 1 of PlanStableSequence in Alg. 1) and fed them to the planner.

To obtain such sets with higher sample variety, we evenly discretized and generated a set of contact regions $\mathcal{C}_\mathrm{e}$ on the object surface with a fixed step size for environmental contacts, and discretized the object surface as a set of contact points $\mathcal{C}_\mathrm{r}$ for robot contacts. A combination $(\mathcal{C}_\mathrm{e}, \mathcal{C}_\mathrm{r})$ of such an environmental and robot contact regions defines a contact profile a system configuration $\boldsymbol{q}$ may have. Then, to map the contact regions $(\mathcal{C}_\mathrm{e}, \mathcal{C}_\mathrm{r})$ into a fully-assigned system configuration $\boldsymbol{q}$, we evenly discretized the structure surfaces in the environment into a set of placement positions. At each position, we checked whether there exists a kinematically valid configuration $\boldsymbol{q}$ meeting the contact profile $(\mathcal{C}_\mathrm{e}, \mathcal{C}_\mathrm{r})$.

In this manner, we sampled three sets of candidate configurations with different set size ($|Q_\mathrm{s}|$=144, 560, 1172) for following experimental studies.

A. Analysis of Minimizing Configuration Changes

We first assessed the performance of our planner in minimizing the number of configuration changes (consequently, task interruptions) along four different forceful tasks:

- Task 1: A rectangular cutting task consisting of 20 continuous cuttings as shown in Fig. 6. The environment has a flat surface in the front of the robot;
- Task 2: A stool fabricating task involving cutting four legs (discretized as twenty cuttings), drilling four holes and inserting four legs (thus a sequence of 28 forceful operations in total) as shown in Fig. 7. The environment has an L-shaped structure;
- Task 3: A chair assembly task consisting of four hole-drillings, four peg-insertings, four leg-insertings, four

TABLE I

NUMBERS OF CONFIG. CHANGES FOR EACH TASK BY THE BASELINE AND PROPOSED PLANNER WITH THREE SIZES OF SAMPLE SETS.

Method	$\|Q_s\|$	# of Configuration Changes			
		Task 1	Task 2	Task 3	Task 4
Baseline	144	23	19	20	27
Proposed	144	4	10	7	9
	560	3	4	3	3
	1172	3	1	1	2

hole-drillings and then four peg-insertings (thus a sequence of 20 forceful operations in total) as shown in Fig. 1. The environment has a flat supporting surface;

- Task 4: A wave cutting task discretized into 20 cuttings as shown in Fig. 3. The environment has a ⊔-shaped supporting structure.

To the best of our knowledge, there exists no planner in the literature directly capable of solving such complex tasks. Existing strategies would, in the best scenario, need to plan each forceful operation individually, neglecting the sequential property that defines a task. We define such a scenario as the *baseline* planner. For each operation in a task, the baseline planner iterates over the available candidate configurations in Q_s until it finds the first stable one.

Table I summarizes the results of configuration changes by the baseline planner and our proposed planner. As shown in the table, compared with the baseline planner, our planner reduces the number of configuration changes dramatically. Specifically, with $|Q_s|$=144, for Task 1 (the rectangular cutting task), the baseline planner finds a solution with 23 configuration changes and therefore almost generates a new configuration for each involved operation (20 operations in total). Our planner generates a more efficient solution (Fig. 6-Left, Solution A), which involves only 4 different system configurations and 4 configuration changes in total. Similarly, Fig. 7 shows a solution generated by our planner for Task 2, which involves only 1 configuration change. Fig. 1 and 3 show an efficient solution for Task 3 and 4 respectively.

It is also notable that as we increase the number of sampled configurations in Q_s, i.e. the set size $|Q_s|$, the planner may come up with better solutions, that is, manipulation plans with a further reduced number of configuration changes, as shown in Table I for all tasks. To better illustrate the difference, take for instance Fig. 6-Right which shows a different solution for Task 1 generated by our planner but with $|Q_s|$=1172. As shown, when the set size $|Q_s|$ increased, the planner came up with a more efficient solution: in Solution A (real robot experiments), the planner requires a regrasping (from right arm in $\boldsymbol{q}_2$ to left arm in $\boldsymbol{q}_3$) whereas in Solution B (from the simulator)[2] the robot is capable to perform the task only with the right hand.

B. Analysis of Planning Efficiency

We further verified the performance of our planner in terms of time efficiency. More specifically, we compared

[2]Note both scenarios were implemented in simulation and in the real robot, yet they are presented separately in Fig. 6 to aid the discussion.

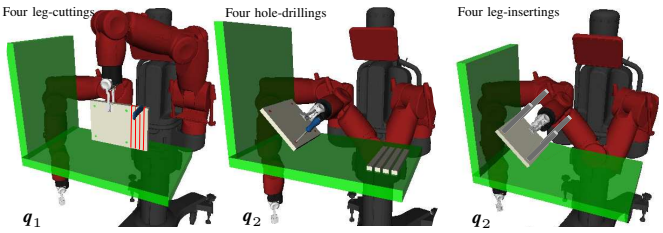


Fig. 7. A manipulation plan for the stool fabricating task (Task 2) involving 28 operations. The solution contains 1 configuration change (environmental contact from $\boldsymbol{q}_1$ to $\boldsymbol{q}_2$).

our planner (Alg. 1) using the naive stability checking in Alg. 2 (Plan-N, for brevity) with the (same) planner but using the containment-based strategy in Alg. 3 (Plan-Cont, for brevity)[3]. Note that the two planners differ in the strategy of implementing the procedure FindStableConfigs in Alg. 1, i.e. building the operation graph which is the most computationally complex procedure in Alg. 1, as it involves perform stability checkings over all forceful operations and candidate configurations.

Taking the same four tasks as previous, Table II summarizes the average time of building the operation graph over 50 runs for each task, with the *total planning time* listed in parentheses. As shown, the containment-based strategy (bold in Table II) increases the planning efficiency significantly. For example, for Task 2 with $|Q_s| = 1172$, it takes about 2087.8s for the Plan-N to build the operation graph compared to 135.8s by the containment-based planner (an improvement of about 15×). Similar analysis can also be made for the other three tasks and configuration sets.

Table II also highlights the time cost of building the operation graph in the total planning time (in parentheses). As shown in the Table, with lower values of $|Q_s|$, i.e. smaller configuration sets, the planner would generate solutions with more configuration changes, and therefore more motion planning iterations would be required. This results in a larger difference between the time of building the operation graph and the total planning time.

It is also important to mention that the containment-based planner requires extra construction of the containment graph as described in Sec. IV-C. Nonetheless, this is a low computational complexity task as illustrated in Table III, which shows the average time of building the containment graph over 50 runs for each task. As shown, the time of building the containment graph increases proportionally to $|Q_s|$, yet it still represents less than 1% of the building time required for the operation graph in Table II and thus negligible in the overall planning. For example, for Task 2 with $|Q_s| = 1172$, it takes only 1.05s for the planner to build the containment graph, but 135.8 s to build the operation graph and 138.2 s for total planning.

Based on above analysis, we can conclude that compared with performing naive stability checkings (Eq. 2) for all operations and sampled configurations, building the containment graph can greatly improve planning efficiency.

[3]In Sec. V-A, the proposed planner refers to Plan-Cont, yet both strategies could have been used as they return the same plans and configuration changes.

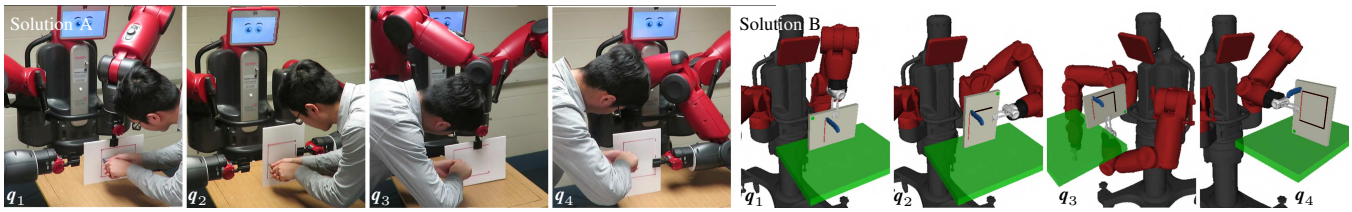


Fig. 6. Two manipulation plans for the rectangular cutting task (Task 1) consisting of 20 operations. Left: Solution A contains 4 configuration changes, with a robot regrasp (from left arm in $\boldsymbol{q}_2$ to right arm in $\boldsymbol{q}_3$). Right: Solution B contains 3 configuration changes.

TABLE II

AVERAGE TIME (S) OF BUILDING THE OPERATION GRAPH AND THE OVERALL PLANNING TIME IN PARENTHESES FOR EACH TASK OVER 50 RUNS.

$\lvert Q_s \rvert$	Task 1		Task 2		Task 3		Task 4	
	Plan-N	Plan-Cont	Plan-N	Plan-Cont	Plan-N	Plan-Cont	Plan-N	Plan-Cont
144	198.5(223.1)	**20.1**(37.7)	280.1(335.0)	**31.0**(81.9)	211.8 (246.8)	**22.9**(56.7)	201.3(256.1)	**22.2**(74.6)
560	773.2(792.0)	**69.7**(86.6)	1100.6(1121.1)	**77.8**(96.0)	821.5(837.5)	**72.4**(87.9)	799.1(820.0)	**70.8**(91.3)
1172	1551.2(1570.1)	**98.3**(115.4)	2087.8(2092.5)	**135.8**(138.2)	1611.7(1615.4)	**112.0**(115.8)	1605.4(1616.7)	**109.9**(120.0)

TABLE III

AVERAGE TIME (S) OF BUILDING THE CONTAINMENT GRAPH.

$\lvert Q_s \rvert$	Task 1	Task 2	Task 3	Task 4
144	0.04	0.06	0.04	0.07
560	0.26	0.40	0.31	0.59
1172	0.97	1.25	1.05	1.52

VI. CONCLUSION AND FUTURE WORK

We presented a manipulation planning approach to keep an object stable under sequential external forces by utilizing both environmental and robot contacts. The proposed planning strategy fills the gap in sequential manipulation planning literature, increasing the aptitude of robots to perform complex tasks efficiently. Indeed, the proposed solution not only reasons about possible object-environment and object-robot contacts during planning, but also addresses the dimensionality and combinatorial explosion for the stability checking procedure in an efficient manner through the introduction of a new concept of wrench based containment relationship between configurations.

REFERENCES

[1] T. Lozano-Pérez, J. Jones, E. Mazer, P. O'Donnell, W. Grimson, P. Tournassoud, and A. Lanusse, "Handey: A robot system that recognizes, plans, and manipulates," 1987.

[2] P. Tournassoud, T. Lozano-Pérez, and E. Mazer, "Regrasping," 1987.

[3] W. Wan and K. Harada, "Developing and comparing single-arm and dual-arm regrasp," *RA-L*, 2016.

[4] N. Chavan-Dafle and A. Rodriguez, "Sampling-based planning of in-hand manipulation with external pushes," in *ISRR*, 2017.

[5] J. Ma, W. Wan, K. Harada, Q. Zhu, and H. Liu, "Regrasp planning using stable object poses supported by complex structures," *IEEE Transactions on Cognitive and Developmental Systems*, 2018.

[6] L. Chen, L. F. Figueredo, and M. Dogar, "Manipulation planning under changing external forces," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3503–3510.

[7] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *The International Journal of Robotics Research*, vol. 3, no. 1, pp. 3–24, 1984.

[8] C. Eppner, R. Deimel, J. Alvarez-Ruiz, M. Maertens, and O. Brock, "Exploitation of environmental constraints in human and robotic grasping," *IJRR*, vol. 34, no. 7, 2015.

[9] T. Nishimura, K. Mizushima, Y. Suzuki, T. Tsuji, and T. Watanabe, "Variable-grasping-mode underactuated soft gripper with environmental contact-based operation," *IEEE Rob & Autom Lett*, vol. 2, 2017.

[10] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," 2004.

[11] P. Lertkultanon and Q.-C. Pham, "A certified-complete bimanual manipulation planner," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 1355–1368, 2018.

[12] X. Ji and J. Xiao, "Planning motions compliant to complex contact states," *IJRR*, vol. 20, no. 6, 2001.

[13] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," 2006.

[14] J. Xiao and X. Ji, "Automatic generation of high-level contact state space," *IJRR*, vol. 20, no. 7, 2001.

[15] T. Lefebvre, "Contact modelling, parameter identification and task planning for autonomous compliant motion using elementary contacts," *PhD thesis, KU Leuven, Department of Mechanical Engineering*, 2003.

[16] T. J. Debus, P. E. Dupont, and R. D. Howe, "Contact state estimation using multiple model estimation and hidden markov models," *The International Journal of Robotics Research*, vol. 23, no. 4-5, 2004.

[17] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012.

[18] A. T. Miller and P. K. Allen, "Graspit!: A versatile simulator for grasp analysis," in *in Proc. of the ASME Dyn Syst & Contr Div*, 2000.

[19] G. Lee, T. Lozano-Pérez, and L. P. Kaelbling, "Hierarchical planning for multi-contact non-prehensile manipulation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 264–271.

[20] M. Murooka, R. Ueda, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, "Global planning of whole-body manipulation by humanoid robot based on transition graph of object motion and contact switching," *Advanced Robotics*, vol. 31, no. 6, pp. 322–340, 2017.

[21] M. Y. Wang and D. M. Pelinescu, "Contact force prediction and force closure analysis of a fixtured rigid workpiece with friction," *Journal of manufacturing science and engineering*, vol. 125, no. 2, 2003.

[22] C. Borst, M. Fischer, and G. Hirzinger, "Grasp planning: How to choose a suitable task wrench space," in *ICRA*, 2004.

[23] K. Hertkorn, M. A. Roa, C. Preusche, C. Borst, and G. Hirzinger, "Identification of contact formations: Resolving ambiguous force torque information," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3278–3284.

[24] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, vol. 79, 2008.

[25] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[26] J. J. Kuffner Jr and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, 2000.