Long-Horizon Torque-Limited Planning through Contact using Discrete Search and Continuous Optimization

Ramkumar Natarajan[1], Garrison L.H. Johnston[2], Nabil Simaan[2], Maxim Likhachev[1] and Howie Choset[1]

***Abstract*— By bracing against the environment robots can expand their reachable workspace that would otherwise be inaccessible due to exceeding actuator torque limits and as well as accomplish tasks beyond their design specifications. As such, it is desirable to interact with the environment to explore new possibilities to complete a task. However, motion planning for complex contact-rich tasks requires reasoning through the permutations of different possible contact modes and the bracing locations that grow exponentially with the number of contact points and links in the robot. To address this combinatorial problem, we developed INSAT [1] that interleaves graph search to explore the manipulator joint configuration space with incremental trajectory optimizations seeded by neighborhood solutions to find a dynamically feasible trajectory through contact. In this paper, we present recent additions to the INSAT algorithm that improve its runtime performance. In particular, we propose Lazy INSAT with reduced optimization rejection that systematically procrastinates its calls to trajectory optimization while reusing feasible solutions that violate boundary constraints. The algorithm is evaluated on a heavy payload transportation task in simulation and on physical hardware. In simulation, we show that Lazy INSAT is able to discover solutions for tasks that cannot be accomplished within its design limits and without interacting with the environment. In comparison to executing the same trajectory without environment support, we show that the utilization of bracing contacts reduces the overall torque required to execute the trajectory.**

I. Introduction

Collaborative robots can reduce the physiological burden for humans performing physically demanding tasks. These robots can assist humans by manipulating heavy payloads. Such robots need large torque actuators and massive links to support its own weight along with the payload. However, such operational requirements compromise the safety of collaboration with the human worker at close proximity. As a result, we are faced with a manipulation planning problem where the planner should minimize the manipulator joint torques and accelerations while respecting task manipulation requirements and avoiding obstacles. To overcome the conflicting requirements of safe collaboration, it has been shown that the robot can brace against the environment to reduce the overall effort required to manipulate heavy objects [2]. The physical constraints imposed by the environment can be transformed into opportunities that can be exploited to enable efficient manipulation that expends low energy, increases accuracy [3], [4], and reduces compliance [5].

[1] The authors are with The Robotics Institute at Carnegie Mellon University, Pittsburgh PA 15213. email: {rnataraj, maxim, choset}@cs.cmu.edu

[2] The authors are with the department of Mechanical Engineering at Vanderbilt University. email: {garrison.l.johnston, nabil.simaan}@vanderbilt.edu

This work was supported by NSF awards #1734461 and #1734460, ARL grant W911NF-18-2-0218 and by Vanderbilt and Carnegie Mellon internal university funds.

Fig. 1: Kinova Gen3 manipulator transferring and flipping (from blue side to orange side facing up) a long heavy wooden plank across a partitioned space by bracing against the environment for support.

Recently, we developed a motion planning algorithm for manipulation that automatically discovers and exploits bracing locations along the entire trajectory to achieve a desired contact-rich task [1]. Consequently, our torque-limited manipulation planning algorithm can opportunistically make/break/sustain contact with the environment to reach deep inside a confined space with insufficient actuator torques or carry a heavy payload beyond the manipulator's capability. The algorithm presented here evolved from [1] in the following ways:

- Based on the observation of the complexity of different re-wiring operations in INSAT, we propose a lazy version of INSAT that delays evaluating long-horizon trajectory optimizations without sacrificing any of the properties of INSAT.
- We extend INSAT to utilize a discrete seed path that is not dynamically feasible. This seed path is also generated within the algorithm and dramatically increases the runtime speed of INSAT.

- Using risk-sensitive cost function transformation and other numerical techniques, we are able to simplify the trajectory optimization when compared to [1].
- One of the biggest limitations of trajectory optimization methods is their inability to work reliably for long-horizon problems. However, in many cases, their solutions are feasible but do not satisfy the given boundary conditions. This work proposes a way to reuse such *partial* solutions by introducing additional nodes to the low-D graph.

The key idea behind our framework is (a) to identify a low-D manifold, (b) perform a search over a grid-based graph that discretizes this manifold, and (c) while searching the graph, utilize contact-implicit trajectory optimization to compute the cost of partial solutions found by the search. As a result, the search over the lower-dimensional graph decides what optimizations to run and with what seeds, while the cost of the solution from the trajectory optimization drives the search in the lower-dimensional graph until a dynamically feasible trajectory from start to goal is found. The flowchart in Fig. 2 gives an overview of the algorithm.

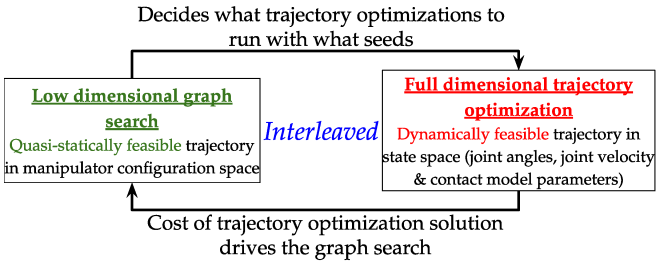

Fig. 2: A schematic of the working principle of INSAT

II. Long-Horizon Torque-Limited Planning w/ Contact

Let the planning state of a N DoF robot be comprised of joint angles and joint velocities $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}] \in \mathcal{X} \subseteq \mathbb{R}^{2N}$. The manipulator is controlled by bounded joint torque inputs $\mathbf{u} \in \mathbb{R}^N$. Consider an invertible many-to-one mapping $\lambda : \mathcal{X} \longrightarrow \mathcal{X}_L$ that projects a full-D state $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}] \in \mathcal{X}$ into a low-D state $\mathbf{x}_L = [\mathbf{q}] \in \mathcal{X}_L$. So $\mathbf{x}_L = \lambda(\mathbf{x})$, $\mathcal{X}$ and $\mathcal{X}_L \subseteq \mathbb{R}^N$ are full-D and low-D spaces. Then $\lambda^{-1} : \mathcal{X}_L \longrightarrow \mathcal{X}$ is an one-to-many inverse mapping of λ that lifts a low-D state $\mathbf{x}_L \in \mathcal{X}_L$ to any possible full-D state $\mathbf{x} \in \mathcal{X}$. So $\mathbf{x} = \lambda^{-1}(\mathbf{x}_L)$. The time ($t$) parameterized full-D trajectory from the subspace $\lambda^{-1}(\mathbf{x}'_L)$ to the subspace $\lambda^{-1}(\mathbf{x}''_L)$ is denoted as $\boldsymbol{\phi}_{\mathbf{x}'\mathbf{x}''}(t)$. For trajectories, the argument t is dropped for brevity.

A. Tunable Smooth Contact Models

A dynamically feasible control trajectory for our application might be non-smooth as the robot has to make/break/sustain contact with the environment. To optimize for such a trajectory using a gradient-based solver, we use two tunable smooth contact models *viz.* (1) virtual contact normal force model and (2) virtual contact friction force, introduced in our previous work [1]. This enables the opposing virtual friction to counteract the static friction from the physics engine and automatically discover establishing and sliding contact between the objects.

Algorithm 1 Trajectory Optimization Routine within INSAT

procedure GenerateTrajectory($\mathbf{x}_L$, $\mathbf{x}'_L$)
  for $\mathbf{x}''_L \in Ancestors(\mathbf{x}_L)$ **do** ▷ From $\mathbf{x}^S_L$ to $\mathbf{x}_L$
    if $\mathbf{x}''_L = \mathbf{x}^S_L$ **then**
      $\lambda^{-1}(\mathbf{x}''_L) = \mathbf{x}^S$
    else if $\mathbf{x}'_L = \mathbf{x}^G_L$ **then**
      $\lambda^{-1}(\mathbf{x}'_L) = \mathbf{x}^G$
    $\phi_{\mathbf{x}''\mathbf{x}'} = O(\lambda^{-1}(\mathbf{x}''_L), \lambda^{-1}(\mathbf{x}'_L))$ ▷ Eq. 1
    if $\phi_{\mathbf{x}''\mathbf{x}'}.IsCollisionFree()$ **then**
      $\phi_{\mathbf{x}^S\mathbf{x}'} = O_w(\phi_{\mathbf{x}^S\mathbf{x}''}, \phi_{\mathbf{x}''\mathbf{x}'})$ ▷ Eq. 1 with warm-start
      return $\phi_{\mathbf{x}^S\mathbf{x}'}$
  return NULL w/ discrete ∞ cost

Algorithm 2 INSAT with Lazy Edge Evaluation and Reduced Optimization Rejection

procedure Key($\mathbf{x}_L$)
  return $g(\mathbf{x}_L) + \epsilon * h(\mathbf{x}_L)$
procedure Main($\mathbf{x}^S, \mathbf{x}^G$)
  $\mathbf{x}^S_L = \lambda(\mathbf{x}^S)$; $\forall \mathbf{x}_L, g(\mathbf{x}_L) = \infty$; $g(\mathbf{x}^S_L) = 0$
  $\theta_{\mathbf{x}^S\mathbf{x}^G}$ = RRTConnect($\mathbf{x}^S_L, \mathbf{x}^G_L$)
  $\forall \mathbf{x}^S_L \in \theta_{\mathbf{x}^S\mathbf{x}^G}$; Insert $\mathbf{x}_L$ in OPEN with Key($\mathbf{x}_L$)
  while Key($\mathbf{x}^G_L$) $= \infty$ **do** ▷ $\mathbf{x}^G_L = \lambda(\mathbf{x}^G)$
    $\mathbf{x}_L$ = OPEN.$pop()$
    $\phi_{\mathbf{x}^S\mathbf{x}}$ = GenerateTrajectory($\mathbf{x}_L$, $\mathbf{x}'_L$) ▷ Sec. II-C
    if $J_{total}(\phi_{\mathbf{x}^S\mathbf{x}}) + h(\mathbf{x}_L) >$ OPEN.$min()$ ▷ Eq. 2
      Re-insert $\mathbf{x}_L$ in OPEN with $\phi_{\mathbf{x}^S\mathbf{x}}$ and Key($\mathbf{x}_L$)
      continue
    if $\phi_{\mathbf{x}^S\mathbf{x}}(T) \neq \mathbf{x}_L$
      Insert $\lambda(\phi_{\mathbf{x}^S\mathbf{x}}(T))$ in OPEN with $\phi_{\mathbf{x}^S\mathbf{x}}$ and Key($\lambda(\phi_{\mathbf{x}^S\mathbf{x}}(T))$)
    $g(\mathbf{x}_L) = J_{total}(\phi_{\mathbf{x}^S\mathbf{x}})$
    $\mathbf{X}'_L = \{Succ(\mathbf{x}_L) \cup \{\mathbf{y}_L \in \theta_{\mathbf{x}^S\mathbf{x}^G} \mid (\mathbf{x}_L, \mathbf{y}_L) \in (\mathcal{X} \setminus \mathcal{X}^{obs}) \cup \mathcal{X}^S\}\}$
    for $\mathbf{x}''_L \in \mathbf{X}'_L$ **do**
      $\mathbf{x}'_L = SoftCopy(\mathbf{x}''_L)$ ▷ Allow state revisiting
      if $\mathbf{x}'_L \in$ CLOSED **then**
        $\mathbf{x}'_L = DeepCopy(\mathbf{x}''_L)$;
      if $g(\mathbf{x}_L) + c_L(\mathbf{x}_L, \mathbf{x}'_L) < g(\mathbf{x}'_L)$ **then**
        $g(\mathbf{x}'_L) = g(\mathbf{x}_L) + c_L(\mathbf{x}_L, \mathbf{x}'_L)$
        Insert $\mathbf{x}'_L$ in OPEN with Key($\mathbf{x}'_L$)

B. Low-Dimensional Graph Search

The low-D space $\mathcal{X}_L$ (N-D) comprised of the joint angles $\mathbf{q}$. We build the low-D graph $\mathcal{G}_L$ by discretizing the contact inclusive joint configuration space of the manipulator. Each edge in the graph corresponds to unit joint movement by a known distance (Fig. 3). Every newly generated node is checked to not violate joint angle and torque limits by calculating the gravity compensation before adding to the graph. So for an N DoF manipulator, the branching factor of the graph is $2N$ (unit joint movement in either direction satisfying joint angle and static joint torque limits). The graph search can be sped up using a heuristic $h(\mathbf{x}_L) = \left\|\mathbf{x}_L - \mathbf{x}^G_L\right\|$, an underestimate on the cost-to-goal of the optimal trajectory. In this low-D graph contact configurations are permitted and generated as described in [1].

C. Trajectory Optimization for Planning through Contact

The mathematical program that we are trying to solve within the graph search in INSAT is given by Eq. 1. Here $\mathbf{k}, \mathbf{b}$ are the stiffness and damping parameters of the virtual contact normal force model and $\boldsymbol{\mu}$ is the virtual friction coefficient of the virtual contact friction model (see [1] for details). In [1], we used Successive Convexification (SCvx) [6] which is similar to indirect methods like iLQR, with the difference that SCvx computes the backward pass updates by maintaining a trust region. The trust region is in turn grown or shrunk depending on the quality of the update. This means that the single update step in SCvx in the best case is the same as iLQR and will be slower than iLQR in the cases when the solution is rejected and the trust region has to be modified.

$$\min_{\mathbf{u}[.],\mathbf{k},\mathbf{b},\mu} \quad l(\mathbf{x},\mathbf{u}) = \left\|\mathbf{x}_L[N] - \mathbf{x}_L^G\right\| + \sum_{i=0}^{N-1}(\|\mathbf{u}[i]\| + \|\dot{\mathbf{x}}[i]\|) + \sum_{n=1}^{N_\Gamma}\|\mathbf{k}\| + \|\mathbf{b}\| + \|\boldsymbol{\mu}\| \tag{1a}$$

$$\text{s.t.} \quad \mathbf{x}[0] = \mathbf{x}^0; \mathbf{x}[i+1] = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i) \tag{1b}$$

$$|\dot{\mathbf{x}}(t)| \leq \dot{\mathbf{x}}_{\text{lim}}; |\ddot{\mathbf{x}}(t)| \leq \ddot{\mathbf{x}}_{\text{lim}}; |\mathbf{u}(t)| \leq \mathbf{u}_{\text{lim}} \tag{1c}$$

Using a parallel line search over the step size to find the best improvement, constrained backward pass [7] that enforces action limits, adaptive regularization and an exponential scalar transformation corresponding to the classical risk-sensitive control framework [8] such as Eq. 2 we are able to obtain faster convergence and better performance using iLQR over the trust region based SCvx.

$$J_{total} = \xi(l(\mathbf{x},\mathbf{u}), R) = \frac{e^{R.l(\mathbf{x},\mathbf{u})} - 1}{R} \tag{2}$$

D. INSAT: INterleaved Search with Trajectory Optimization

1) Lazy INSAT: A comprehensive exposition of INSAT is presented in [1]. Here we present INSAT with Lazy edge evaluation (magenta lines) using the pseudocode in Alg. 2. We had a crucial observation that when using indirect trajectory optimization techniques, warm-starting large problems is significantly faster than solving from scratch but is slower than solving small problems from scratch. Here the small problems correspond to the incremental optimizations we perform to get the trajectory from the state being expanded to the successor and warm-starting corresponds to finding the full trajectory to the successor from the start (Fig. 3). In other words, although the warm-starting step in INSAT was making a significant improvement to runtime and the convergence of long-horizon problems, the longer the horizon is, the higher the total runtime of the optimization is. Thus, it would be beneficial to postpone as many warm-starting and incremental optimizations as possible. To this end, we propose a lazy version of INSAT (Alg. 2), that maintains a lower bound on the solution of trajectory optimization as a pseudo cost to sort the priority queue. The true cost is computed only when the node is picked for expansion rather than when it was generated as in [1]. The true cost will increase the g-value of the node picked for expansion and hence this node need not be at the top of the OPEN list anymore. In that case, the node is inserted back into the OPEN list and the next state at the top is picked. The state picked for expansion is expanded only when its true cost is also the lowest cost among the costs of the node in OPEN.

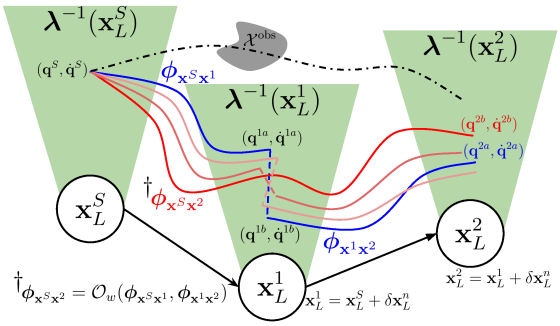

Fig. 3: Illustration of low-D graph, full-D subspaces of low-D states, trajectory optimization $O(.)$ and warm-started trajectory optimization $O_w(.)$ and iterating over low-D ancestors (line 2).

2) Reduced Rejection of Optimized Trajectory: The graph search in INSAT operates on a discrete low-D space. As a result, the optimizer is required to solve a boundary value problem that exactly connects the cell centers of discrete cells. However, in practice, some relaxations are allowed on these stiff constraints using penalty methods. Though this improves the behavior of the optimization, there are many instances where the optimization converges to a feasible trajectory but does not satisfy the boundary condition. Since we use indirect methods and rely on rollouts to generate a dynamically feasible trajectory, the starting point condition is satisfied by definition. We reuse the feasible but non-convergent trajectories by introducing or updating a new/existing node in the low-D graph based on the terminal point of the reused trajectory (orange lines).

3) Seed Low-D search with RRT-Connect: Bidirectional sampling-based planners like RRT-Connect can be very fast for high dimensional manipulation problems at the cost of poor solution quality and dynamic infeasibility. We use the RRT-Connect solution to seed the low-D discrete graph search by inserting the OPEN priority queue with the nodes on the RRT-Connect solution at the start of the algorithm (red lines). These nodes from the RRT-Connect serve as visibility nodes to escape local minima yet preserve bounded suboptimality guarantees of the graph search.

III. Experiments and Results

A. Simulation Experiments

We evaluated Lazy INSAT for the task of transporting and flipping a long wooden plank (Fig. 5) across a partition. Here the flipped goal configuration of the payload is shown with a lower alpha. The different colors of the faces of the payload demonstrate the object being flipped when moved across the partition. We evaluated the planner across three different heights of the partition and several different combinations of weight of the payload. The one shown in Fig. 5 is one of the most difficult scenarios where the robot has to transport and flip a 6kg payload well outside its capable limits. Further note that the weight of the payload is concentrated far away

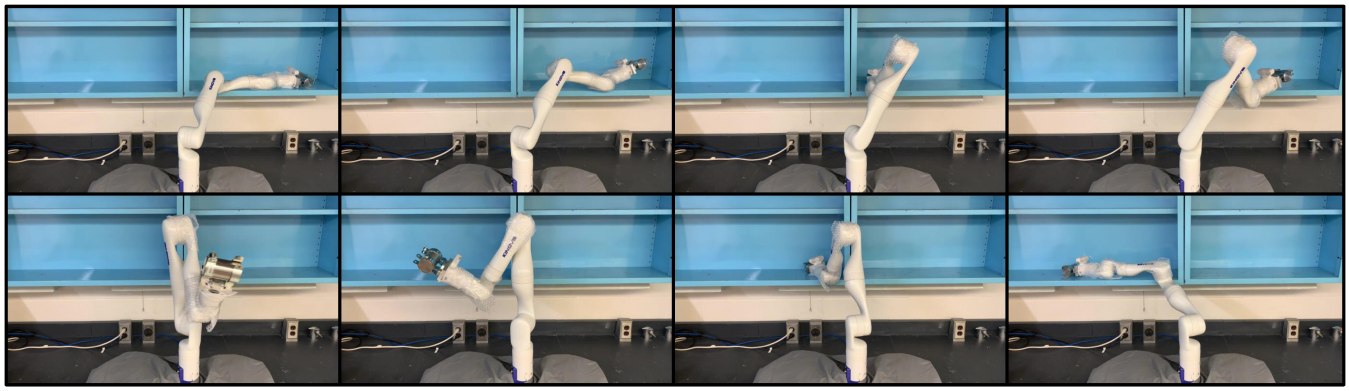

Fig. 4: Film strip showing a 7 DoF Kinova Gen3 robot utilizing bracing contacts to transfer a 2.5 kg payload between two cabinets using minimal torque. The arm slides the payload all the way to the center by bracing with its wrist before lifting on its own. The payload is then placed at the proximal end of the target shelf and pushed by bracing its forearm.

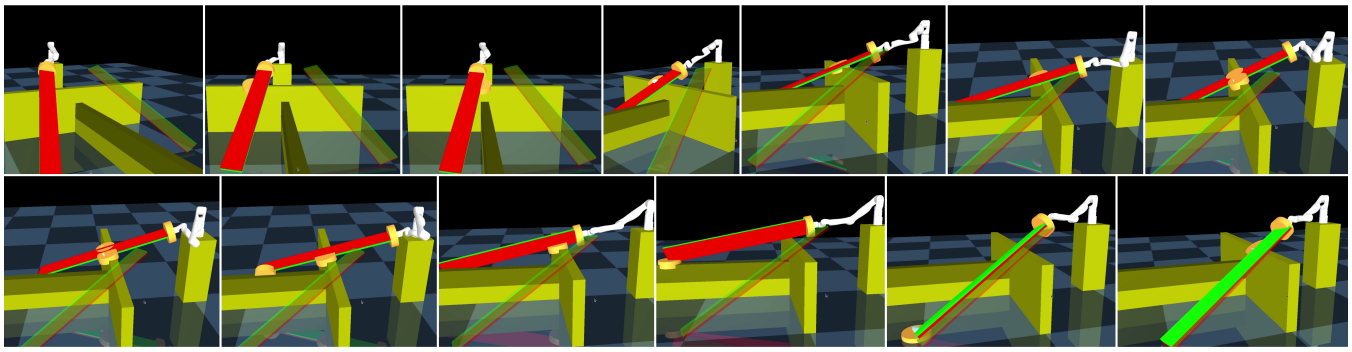

Fig. 5: Film strip showing Kinova Gen3 arm transporting and flipping an overweight payload of 6kg. The maximum manufacturer payload limit is 4kg at configurations close to the base of the robot. The robot cannot lift the payload at any instant and has to find its way to goal via a tiny constrained manifold of its entire workspace. Lazy INSAT finds the solution by leveraging contact.

from the tool tip because of the length of the payload making the effective payload much more than 6kg. INSAT discovers a 24s long-horizon trajectory to keep the payload in contact throughout the task as it is the only option. The payload cannot be lifted by the robot in any configuration with lower momentum.

B. Real Robot Experiments

In order to experimentally validate our method, we used the Kinova Gen 3 robot to lift a 2.5 Kg payload between adjacent cabinets as shown in Fig. 4 and to flip a 1.7kg bulky payload across the partition (same experiment as above on physical hardware) as shown in Fig. 1. In both cases, INSAT was able to accomplish the task successfully by exhibiting different behaviors. As a comparison point, the trajectories were executed in free space (i.e., without the cabinets and the partition). Table I shows the RMS of the sensed joint torques in both experiments. Note that in the flipping task, the second joint (shoulder) of the arm reaches very close to its maximum torque of 39Nm when executed in the free space. The percentage RMS torque savings in both experiments are 22% and 68% respectively. From these values, it is clear that our planner was able to utilize bracing contacts to meaningfully reduce the torque in most joints when compared to free-space motion.

Joint ID	1	2	3	4	5	6	7	Total
Free	1.93	33.28	10.14	14.06	0.97	6.29	0.20	66.86
	6.12	37.73	11.05	16.8	3.11	10.25	0.48	85.53
Bracing	4.65	23.96	6.67	11.80	1.28	5.67	0.59	54.62
	8.61	15.5	3.39	10.71	4.94	5.11	2.41	50.68
Difference	-2.72	9.32	3.46	2.26	-0.31	0.63	-0.40	12.24
	-2.48	22.22	7.65	6.09	-1.83	5.13	-1.93	34.85

TABLE I: Grayed row corresponds to payload transportation between shelves in Fig. 4 and white row corresponds to flipping the bulky wooden plank in Fig. 1. Experimental RMS torques [Nm] during (i) the braced trajectory shown in Fig. 4 and (ii) the same trajectory running in free-space (i.e. without the cabinets or partition wall) producing net savings of 12.24Nm and 34.85Nm respectively.

IV. Conclusion & Future Directions

With the Lazy edge evaluation in INSAT along with reduced optimization rejection our planning time has come down to an average of 3 minutes from >15 minutes in [1]. Our current work on the multi-threaded version of INSAT using parallelized graph search algorithms is showing around further 15x reduction in planning time. We show that planning with torque and obstacle constraints can be achieved in a way that finds bracing locations in the environment in order to make an otherwise inaccessible configuration reachable due to the torque reduction achieved by bracing. Experiments showed that the use of bracing contacts can reduce the required actuator torque for a given trajectory.

REFERENCES

[1] R. Natarajan, G. L. Johnston, N. Simaan, M. Likhachev, and H. Choset, "Torque-limited manipulation planning through contact by interleaving graph search and trajectory optimization," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8148–8154.

[2] C. Fang, N. Kashiri, G. F. Rigano, A. Ajoudani, and N. G. Tsagarakis, "Exploitation of environment support contacts for manipulation effort reduction of a robot arm," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9502–9508.

[3] R. Hollis and R. Hammer, "Real and virtual coarse-fine robot bracing strategies for precision assembly," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, pp. 767–774.

[4] G. Wang and M. Minami, "Modelling and control of hyper-redundancy mobile manipulator bracing multi-elbows for high accuracy/low-energy consumption," in *Proceedings of SICE Annual Conference 2010*. IEEE, 2010, pp. 2371–2376.

[5] G. L. Johnston, A. L. Orekhov, and N. Simaan, "Kinematic modeling and compliance modulation of redundant manipulators under bracing constraints," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4709–4716.

[6] A. Ö. Önol, P. Long, and T. Padır, "Contact-implicit trajectory optimization based on a variable smooth contact model and successive convexification," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2447–2453.

[7] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[8] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive sampling: Real-time behaviour synthesis with mujoco," 2022.