Serving Time: Real-Time, Safe Motion Planning and Control for Manipulation of Unsecured Objects

Zachary Brei[1], Jonathan Michaux[2], Bohao Zhang[2], Patrick Holmes[2], Ram Vasudevan[1]

***Abstract*— A key challenge to ensuring the rapid transition of robotic systems from the industrial sector to more ubiquitous applications is the development of algorithms that can guarantee safe operation while in close proximity to humans. Motion planning and control methods, for instance, must be able to certify safety while operating in real-time in arbitrary environments and in the presence of model uncertainty. This paper proposes Wrench Analysis for Inertial Transport using Reachability (WAITR), a certifiably safe motion planning and control framework for serial link manipulators that manipulate unsecured objects in arbitrary environments. WAITR uses reachability analysis to construct over-approximations of the contact wrench applied to unsecured objects, which captures uncertainty in the manipulator dynamics, the object dynamics, and contact parameters such as the coefficient of friction. An optimization problem formulation is presented that can be solved in real-time to generate provably-safe motions for manipulating the unsecured objects. This paper illustrates that WAITR outperforms state of the art methods in a variety of simulation experiments and demonstrates its performance in the real-world.**

I. Introduction

A key challenge of non-prehensile robotic manipulation is safe trajectory planning for manipulation of supported objects. The lack of force/form closure in non-prehensile manipulation tasks means that an incorrectly applied wrench can result in damage to the unsecured objects or the environment. This challenge is compounded when the non-prehensile manipulation task must be performed in environments that may only be known at run-time as this requires real-time generation of motion plans. Because the model of the object being manipulated may not be known perfectly, these methods must be able to account for model uncertainty as well. Therefore, robotic manipulators need to be capable of finding and robustly executing provably safe trajectories in real-time.

To safely manipulate supported objects, the relative motion between the objects and supporting surface needs to be constrained. To generate such a motion, previous methods formulate optimal control problems which solve for a time-optimal trajectory along a pre-specified, discretized geometric path that is assumed to be collision-free [1]–[4]. Relative motion is partially or fully constrained through a friction cone constraint [2], as well as a lift and tipping constraint [1] and a rotational friction cone constraint [3], [4]. Note that relative motion is fully constrained only if all six degrees of freedom of the object are constrained. Other methods also incorporate optimization of the geometric path into the time-optimal trajectory following task [3], [5]–[7].

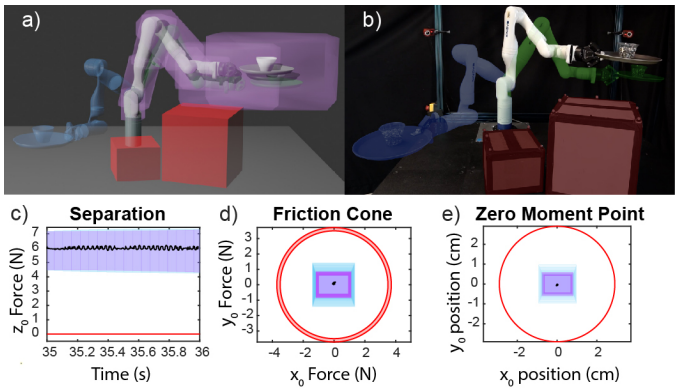

Fig. 1. This paper considers the problem of safe motion planning for manipulation of unsecured objects with uncertain dynamics such as manipulating an unsecured cup filled with an uncertain mass around randomly placed obstacles (red) such that the cup does not move relative to the tray supporting it. WAITR operates in receding-horizon fashion, moving from a start configuration (blue) to a global goal (green) by repeatedly generating new motion plans in real-time. In each motion planning iteration, WAITR calculates a reachable set (blue and purple) for the contact wrench between the manipulator and the object as well as a Forward Reachable Set (FRS) for the whole manipulator system for a continuum of possible motion plans. The FRS is shown in purple in a) for a single planning iteration. WAITR solves a constrained trajectory optimization problem to find a collision-free motion in this FRS that does not result in relative motion while making progress towards an intermediate waypoint (grey) and the global goal. Parts c)-e) show the contact constraints enforced during a hardware experiment for a single planning iteration.

Unfortunately, these methods are unable to generate a provably safe continuous-time trajectory in real-time while also accounting for modeling error. In particular, during implementation, the optimization problems are only able to represent the collision avoidance constraint at discrete time-instances. Finer discretizations result in trajectories that are more likely to be safe and dynamically feasible, but have higher computation times. More troublingly, time-optimal trajectories commonly result in the robot operating near the constraint boundaries. If not dealt with robustly, then uncertainty in the properties of the manipulator or object being manipulated could result in the *executed* time-optimal trajectory being unsafe. Uncertainty in both parameter estimation and execution error was considered in [8], however, the contact and dynamics constraints are still enforced at discrete time instances.

The contributions of this paper are two-fold. First, we develop a framework that uses polynomial zonotopes to represent the reachability of wrenches exerted throughout a serial chain manipulator, including contact wrenches applied to manipulated objects. Second, we formulate a trajectory

[1]Mechanical Engineering, University of Michigan, Ann Arbor, MI ⟨breizach, ramv⟩@umich.edu.

[2]Robotics, University of Michigan, Ann Arbor, MI ⟨jmichaux, jimzhang, pdholmes, ramv ⟩@umich.edu.

Operation	Computation
$[z] \to \mathbf{z}$ (Interval Conversion) (A8)	Exact
$\mathbf{P}_1 \oplus \mathbf{P}_2$ (PZ Minkowski Sum) (A10)	Exact
$\mathbf{P}_1\mathbf{P}_2$ (PZ Multiplication) (A12)	Exact
$\mathbf{P}_1 \otimes \mathbf{P}_2$ (PZ Cross Product) (A13)	Exact
$\texttt{slice}(\mathbf{P}, x_j, \sigma)$ (A14)	Exact
$\texttt{inf}(\mathbf{P})$ (A16) and $\texttt{sup}(\mathbf{P})$ (A15)	Overapproximative
$f(\mathbf{P}_1) \subseteq \mathbf{P}_2$ (Taylor expansion) A(23)	Overapproximative

TABLE I
Summary of polynomial zonotope operations.

optimization problem that can be tractably solved in real-time with continuous-time safety guarantees for preventing relative motion of unsecured objects. As illustrated in Fig. 1, this framework is implemented in a receding horizon fashion and can robustly handle uncertainty in both the manipulator and object parameters. Note this framework extends ARMOUR [9], which provides collision-free and dynamically feasible trajectories in real-time that account for tracking error due to modeling uncertainty in the manipulator, but is unable to make guarantees about the wrenches applied during motion. To make this distinction clear, WAITR is tested in simulation and on hardware and compared against ARMOUR.

Next, we briefly summarize the structure of this paper. Sec. II presents relevant notation and mathematical objects. Sec. III describes the robot dynamics, the contact model and constraints, and presents a continuous-time optimization problem that ensures safe manipulation of unsecured objects. Sec. IV summarizes a passivity-based robust controller [9], then describes the generation of polynomial zonotope overapproximations of the manipulator's trajectory and contact wrench and how that is used to form a tractable implementation of the continuous-time optimization problem which has continuous-time safety guarantees. Section V details the simulation and hardware experiments.

II. Preliminaries

This section describes our notation conventions, several set representations, and operations on these set representation. These operations are summarized for convenience in Tab. I. This paper uses a letter preceding an equation number, e.g. (A12), to refer to equations provided in supplementary appendices which can be found at https://roahmlab.github.io/waitr-dev/.

The n-dimensional real numbers are $\mathbb{R}^n$, natural numbers $\mathbb{N}$, the unit circle is $\mathbb{S}^1$, and the set of 3×3 orthogonal matrices is $\mathrm{SO}(3)$. Subscripts may index elements of a vector or a set. Let U, $V \subset \mathbb{R}^n$. For a point $u \in U$, $\{u\} \subset U$ is the set containing only u. The *power set* of U is $\mathcal{P}(U)$. The *Minkowski Sum* is $U \oplus V$ the *Minkowski Difference* is $U \ominus V = U \oplus (-V)$. For vectors $a, b \in \mathbb{R}^3$, we write the cross product $a \times b$. If $n = 3$, the *set-based cross product* is defined as $U \otimes V = \{u \times v \mid u \in U, v \in V\}$. If $\{U_i \subset \mathbb{R}^n\}_{i=1}^m$ then let $\bigtimes_{i=1}^n U_i$ denote the Cartesian product of the U_i's. Let $X \subset \mathbb{R}^{n\times n}$ be a set of square matrices. Then, *set-based multiplication* is defined as $XV = \{Av, \mid A \in X, v \in V \subseteq \mathbb{R}^n\}$. Let *0* (resp. *1*) denote a matrix of zeros (resp. ones) of appropriate size, and let I_n be the $n \times n$ identity matrix.

We use intervals to describe uncertain manipulator parameters [10]. An *n-dimensional interval* is a set $[x] = \{y \in \mathbb{R}^n \mid \underline{x}_i \leq y_i \leq \bar{x}_i,\ \forall i = 1, \ldots, n\}$. When the bounds are important, we denote an interval $[x]$ by $[\underline{x}, \bar{x}]$, where $\underline{x}$ and $\bar{x}$ are the infimum and supremum, respectively, of $[x]$. Let $\mathbb{IR}^n$ be the set of all real-valued n-dimensional interval vectors.

Because multidimensional intervals can only describe hyperrectangles, they may be overly conservative when outer approximating other shapes. To build a tighter outer-approximative representation of multidimensional sets, we use zonotopes and polynomial zonotopes. Here, we provide necessary definitions of zonotopes and polynomial zonotopes as well as a few important operations, with more operations summarized in Tab. I. App. A [1] rigorously defines the operations given in Tab. I.

To define zonotopes, we introduce an indeterminate vector $x \in [-1, 1]^{n_g}$ and exponents $\alpha_i \in \mathbb{N}^{n_g}$. Letting $\alpha_1 = [1, 0, \ldots, 0]$, $\alpha_2 = [0, 1, 0, \ldots, 0]$, $\ldots$, $\alpha_{n_g} = [0, 0, \ldots, 0, 1]$, we note that $x^{\alpha_1} = x_1, \ldots, x^{\alpha_{n_g}} = x_{n_g}$ where the exponentiation is applied element-wise. A *zonotope* $Z \subset \mathbb{R}^n$ is a convex, centrally-symmetric polytope defined by a *center* $g_0 \in \mathbb{R}^n$ and *generators* $g_i \in \mathbb{R}^n$ as $Z = \{z \in \mathbb{R}^n \mid z = \sum_{i=0}^{n_g} g_i x^{\alpha_i}, x \in [-1, 1]^{n_g}\}$, where there are $n_g \in \mathbb{N}$ generators.

Note we have written Z as the set of points produced by the polynomial $p(x) = \sum_{i=0}^{n_g} g_i x^{\alpha_i}$ over the domain $x \in [-1, 1]^{n_g}$. A polynomial zonotope is the more general case where exponents $\alpha_i \in \mathbb{N}^{n_g}$ can be arbitrary, which includes zonotopes as a subset. When we need to emphasize the generators and exponents of a polynomial zonotope, we write $\mathbf{P} = \mathcal{PZ}(g_i, \alpha_i, x)$. Note we exclusively use bold symbols to denote polynomial zonotopes.

We use polynomial zonotopes $\mathbf{P}$ to represent a set of possible positions of a robot arm operating near an obstacle. It may be beneficial to know whether a particular choice of $\mathbf{P}$'s indeterminates yields a subset of positions that could collide with the obstacle. To this end, we introduce the operation of "slicing" a polynomial zonotope $\mathbf{P} = \mathcal{PZ}(g_i, \alpha_i, x)$ by evaluating an element of the indeterminate x. Given the j^{th} indeterminate x_j and a value $\sigma \in [-1, 1]$, let $\texttt{slice}(\mathbf{P}, x_j, \sigma)$ denote the slicing operation which yields a subset of $\mathbf{P}$ by plugging σ into the specified element x_j and is formally defined in (A14). It is possible to efficiently generate upper and lower bounds on the values of a polynomial zonotope through overapproximation. In particular, we define the $\texttt{sup}$ and $\texttt{inf}$ operations which return these upper and lower bounds, respectively in (A15) and (A16). Note that for any $z \in \mathbf{P}$, $\texttt{sup}(\mathbf{P}) \geq z$ and $\texttt{inf}(\mathbf{P}) \leq z$, where the inequalities are taken element-wise.

III. Wrench and Online Optimization

This section introduces an extended arm model for the manipulator-tray-object system and a contact model between the tray and object. To simplify exposition, this paper primarily focuses on ensuring that the manipulator operates without causing any relative motion between the serving tray and the objects on it. Section V describes how to apply ARMOUR [9] to ensure that the robot does not collide with obstacles while satisfying input and joint limit constraints.

[1] Supplementary Appendices found at https://roahmlab.github.io/waitr-dev/

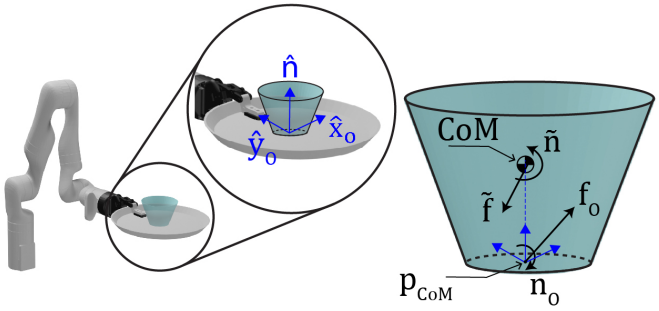

Fig. 2. Free body diagram of a manipulated object. p_{CoM} is the projection of the COM onto the contact plane and $\hat{n}$ is the contact normal vector and is chosen to be the z-axis of the contact frame.

A. Manipulator Model

Given an n_q-dimensional serial robotic manipulator with configuration space Q and a compact time interval $T \subset \mathbb{R}$ we define a trajectory for the configuration as $q : T \to Q \subset \mathbb{R}^{n_q}$. The velocity of this configuration trajectory is $\dot{q} : T \to \mathbb{R}^{n_q}$. We make the following assumptions about the robot model:

Assumption 1. *The robot operates in a three dimensional workspace, $W \subset \mathbb{R}^3$. The robot is composed of revolute joints, where the j^{th} joint actuates the robot's j^{th} link. The robot has encoders that allow it to measure its joint positions and velocities. The robot is fully actuated, where the robot's input $u : T \to \mathbb{R}^{n_q}$.*

We make the one-joint-per-link assumption with no loss of generality because joints with multiple degrees of freedom (*e.g.*, spherical joints) may be represented using links with zero length. This work can be extended to robots with prismatic joints by straightforward extensions to the RNEA algorithms presented in [9]. Finally, let $N_q = \{1, \ldots, n_q\}$.

1) Tray and Object Model: As illustrated in Fig. 2, the system considered in this paper is a robotic manipulator that grasps a flat surfaced tray with an object resting on it. The contact normal to the tray is denoted by the unit vector $\hat{n} \in \mathbb{R}^3$. We make the following assumption:

Assumption 2. *The manipulator maintains force closure on the tray for all time. The object has a circular contact area with radius r. The friction between the tray and the object is described by linear Coulomb Friction where the coefficient of static friction is μ_s.*

As a result, the tray can be treated as part of the link which maintains force closure on it, which we choose to be the n_q^{th} link for convenience. The approach described in this paper can be generalized to ensure that a force closure condition was satisfied throughout the motion, but in the interest of simplicity, we leave that extension for future work.

2) Kinematics and Dynamics: Next, we introduce the kinematics and dynamics of the system. Suppose there exists a fixed inertial reference frame, which we call the *world* frame, and a *base* frame, which we define as 0^{th} frame, that indicates the origin of the robot's kinematic chain. The j^{th} reference frame $\{\hat{x}_j, \hat{y}_j, \hat{z}_j\}$ is attached to the robot's j^{th} revolute joint, and $\hat{z}_j = [0,0,1]^\top$ corresponds to the j^{th} joint's axis of rotation for $j \in N_q$. For a configuration at a particular time, $q(t)$, the position and orientation of frame j with respect to frame $j-1$ can be expressed using homogeneous transformations [11, Ch. 2] with a a configuration-dependent rotation matrix, $R_j^{j-1}(q_j(t))$, and a fixed translation vector from frame $j-1$ to frame j, p_j^{j-1}.

The robot is composed of n_q rigid links with inertial parameters given by the vector $\Delta_{arm} = (m_1, \ldots, m_{n_q}, c_{x,1}, \ldots, c_{z,n_q}, I_{xx,1}, \ldots, I_{zz,n_q})^\top$, where m_j, $c_j = (c_{x,j}, c_{y,j}, c_{z,j})^\top$, and $(I_{xx,j}, \ldots, I_{zz,j})^\top$ represent the mass, center of mass (CoM), and inertia tensor of the j^{th} link, respectively. The properties for n_q^{th} link are comprised of the gripping link and tray properties. The object is a rigid body with inertial parameters given by the vector $\Delta_o = (m_o, c_{x,o}, \ldots, c_{z,o}, I_{xx,o}, \ldots, I_{zz,o})^\top$, where m_o, $c_o = (c_{x,o}, c_{y,o}, c_{z,o})^\top$, and $(I_{xx,o}, \ldots, I_{zz,o})^\top$ represent the mass, center of mass, and inertia tensor of the object, respectively.

The arm dynamics are represented by the standard manipulator equations [11, (10.13)]

$$\begin{aligned} M(q(t), \Delta_{arm})\ddot{q}(t) + C(q(t), \dot{q}(t), \Delta_{arm})\dot{q}(t) + G(q(t), \Delta_{arm}) \\ = u(t) + J^\top(q(t))\mathrm{w}_c(q(t)) \end{aligned} \tag{1}$$

where $M(q(t), \Delta_{arm}) \in \mathbb{R}^{n_q \times n_q}$ is the positive definite inertia matrix, $C(q(t), \dot{q}(t), \Delta_{arm}) \in \mathbb{R}^{n_q \times n_q}$ is the Coriolis matrix, $G(q(t), \Delta_{arm}) \in \mathbb{R}^{n_q}$ is the gravity vector, $u(t) \in \mathbb{R}^{n_q}$ is the input torque, $J(q(t)) \in \mathbb{R}^{6 \times n_q}$ is the manipulator Jacobian, and $\mathrm{w}_c(q(t)) \in \mathbb{R}^6$ is the wrench applied by the object onto the manipulator, all at time t.

Let $\mathrm{w}_o = -\mathrm{w}_c$ be the wrench that the manipulator applies to the object. We refer to w_o as the contact wrench and define it as $\mathrm{w}_o(q(t)) = \begin{bmatrix} \mathrm{f}_o(q(t))^\top & \mathrm{n}_o(q(t))^\top \end{bmatrix}^\top$ where $\mathrm{f}_o(q(t)) \in \mathbb{R}^3$ is the contact force and $\mathrm{n}_o(q(t)) \in \mathbb{R}^3$ is the contact moment [12, (5.99)]. If there are no other external wrenches applied to the object, then the object at time t is manipulated solely by the contact wrench. The object dynamics can be represented using standard Newton-Euler equations for a rigid body.

While following a trajectory, if the object moves with respect to the tray during operation, we say that the object is experiencing *relative motion*. **This paper's goal is to devise a planning strategy to avoid relative motion** when beginning from some initial condition while trying to reach a user-specified goal configuration or end effector position. Importantly, when there is no relative motion, the object can be treated as a rigid body attached to the tray. However, instead of treating it as part of the last link similar to the tray, we instead define a *fixed joint* between the object and tray. We define the location of the fixed joint to be at p_{CoM}, as shown in Fig. 2. The reference frame for the fixed joint has the z-axis $\hat{z}_o$ chosen to be aligned with the contact normal $\hat{n}$. We choose the y-axis $\hat{y}_o$ to be oriented towards the plane of the supporting surface aligned with joint n_q and Throughout the remainder of the paper, we refer to this manipulator, tray, and object system as the *extended arm*. We also let $N_e = \{1, \ldots, n_q + 1\}$.

There are several benefits to this modeling choice. The first is that the dynamics of the entire system, including the object, can be calculated using the forward pass of a modified Recursive Newton-Euler Algorithm (RNEA) which is described in App. B[1]. The second is that the backward pass of RNEA calculates the wrench exerted through the fixed joint by the tray on the object, which due to the choice of joint location, corresponds to the contact wrench applied to the object.

For the extended arm, the object inertial parameters in Δ_o are inserted into the n_{q+1}^{th} positions in the manipulator inertial parameter vector Δ_{arm} to form the extended arm inertial parameter vector, Δ. We make the following assumption about the extended arm's inertial parameters:

Assumption 3. *The model structure (i.e., number of joints, sizes of links, etc.) of the robot is known, but its true inertial parameters Δ are unknown. The uncertainty in each inertial parameter is known and given by the interval vector, $[\Delta] = ([m_1], \ldots, [m_o], [c_{x,1}], \ldots, [c_{z,o}], [I_{xx,1}], \ldots [I_{zz,o}])^\top$. The true parameters lie in this interval, i.e., $\Delta \in [\Delta]$. All elements of this interval vector have bounded elements and any inertia tensor drawn from $[I_j]$ must be positive semidefinite.*

Before continuing, we make one final observation. **Note that the state of the robot depends on the inertia parameter vector.** We leave out this dependence for convenience unless we want to emphasize the importance of this dependence in which case we write $q(t;\Delta)$.

B. Trajectory Parameterization and Online Control

WAITR performs planning in a receding horizon fashion. We assume without loss of generality that the control input and trajectory of a planning iteration begin at time $t = 0$ and end at a fixed time t_f. To ensure persistent real-time operation, we require that WAITR identifies a new trajectory parameter within a fixed planning time of t_p seconds, where $t_p < t_\mathrm{f}$. Thus WAITR has limited planning time and must select a new trajectory parameter before completing its tracking of the previously identified desired trajectory.

In each planning iteration, WAITR chooses a desired trajectory to be followed by the arm. These trajectories are chosen from a continuum of trajectories, with each uniquely determined by a *trajectory parameter* $k \in K$ and are written as $q_d(t;k)$ or $q_d(t;k,\Delta)$ when we want to emphasize the dependence on the inertia parameter vector. The set $K \subset \mathbb{R}^{n_k}$, $n_k \in \mathbb{N}$, is compact and represents a user-designed continuum of trajectories. In general, K can be designed to include trajectories designed for a wide variety of tasks and robot morphologies [13]–[15]. Finally, we assume that $\ddot{q}_d(\cdot;k)$ is a Lipschitz continuously differentiable function and at $\dot{q}_d(t_\mathrm{f};k) = \ddot{q}_d(t_\mathrm{f};k) = 0$.

A user can track a particular desired trajectory by applying some feedback controller. As a result we write $q(\cdot;k)$ for a trajectory of the system at time k. Note WAITR uses a specific type of feedback controller that is described in Sec. IV-A. Because the feedback controller can be a function of the state and the desired trajectory and its derivatives, we define a *total feedback trajectory*, q_A, as:

$$q_A(t;k) = \begin{bmatrix} q(t;k) & \dot{q}(t;k) & q_d(t;k) & \dot{q}_d(t;k) & \ddot{q}_d(t;k) \end{bmatrix}. \quad (2)$$

C. Contact Constraints

To ensure that the object does not experience relative motion, we need to compute the wrench applied on the object during motion. This contact wrench is applied by the manipulator through the supporting surface, and an equivalent contact wrench w_o can be written with respect to point p_CoM. The contact wrench is a function of the robot's configuration trajectory and its velocity and acceleration. Because there are no external wrenches applied to the object, the object at time t is manipulated solely by the following contact wrench w_o. Then, the motion of the object can be constrained in all six degrees of freedom by using components of the contact wrench w_o. Using the following lemma, whose proof can be found in App. C[1], one can ensure no relative motion occurs:

Lemma 4. *Let the* vertical separation *be defined as:*

$$h_{sep}(\mathrm{w}_o(q_A(t;k))) := -f_{o,z}(q_A(t;k)) \quad (3)$$

Let the linear slip *be defined as:*

$$\begin{aligned} h_{slip}(\mathrm{w}_o(q_A(t;k))) := (f_{o,x}(q_A(t;k)))^2 + \\ + (f_{o,y}(q_A(t;k)))^2 - (\mu_s f_{o,z}(q_A(t;k)))^2 \end{aligned} \quad (4)$$

Let the tip *of the object be defined as:*

$$\begin{aligned} h_{tip}(\mathrm{w}_o(q_A(t;k))) := (\hat{n} \times n_o(q_A(t;k)))^2 + \\ - r^2(\hat{n} \cdot f_o(q_A(t;k)))^2 \end{aligned} \quad (5)$$

If (3),(4)*, and* (5) *are less than or equal to zero for all $t \in T$, then the object does not experience relative motion while the robot is moving along q.*

D. Online Trajectory Optimization

Then by using this parameterization and the constraints described in Sec. III-C, one could compute a trajectory to manipulate an unsecured object with zero relative motion by solving the following problem:

$$\min_{k \in K} \quad \texttt{cost}(k) \quad (6)$$

$$h_{sep}(\mathrm{w}_o(q_A(t;k,\Delta))) \leq 0 \qquad \forall t \in T, \Delta \in [\Delta] \quad (7)$$

$$h_{slip}(\mathrm{w}_o(q_A(t;k,\Delta))) \leq 0 \qquad \forall t \in T, \Delta \in [\Delta] \quad (8)$$

$$h_{tip}(\mathrm{w}_o(q_A(t;k,\Delta))) \leq 0 \qquad \forall t \in T, \Delta \in [\Delta]. \quad (9)$$

Note that the total trajectory in this instance is a function of the parameterized trajectory and the true inertia parameters of the extended arm. Because these parameters may not be known, we require that the constraints be satisfied for all possible inertia parameters. Implementing a real-time optimization algorithm to solve this problem is challenging for several reasons. First, the dynamics of the robot are nonlinear and constructing an explicit solution to them is intractable. Second, the constraints must be satisfied for all time t in an uncountable set T.

[1]Supplementary Appendices found at https://roahmlab.github.io/waitr-dev/

IV. Planning Algorithm Formulation

The key technical idea of this work is forming polynomial zonotope overapproximations of the desired trajectories and the contact wrench. This enables us to generate polynomial zonotope overapproximations of all constraints in the optimization problem (6)-(9) at the start of the planning iteration, then use these constraint overapproximations to perform optimization within t_p. Then, we can use the robust passivity-based controller proposed in [9, Sec. VII] to bound the tracking error of the extended arm. This section begins by summarizing the important properties of the aforementioned robust passivity-based controller. Subsequently, it describes how to represent the aforementioned constraints conservatively using polynomial zonotopes. This section concludes by describing how to transform the optimization problem (6)-(9) into an implementable version whose solutions can be be followed by the robot without experiencing relative motion while satisfying input constraints.

A. Robust Passivity-Based Controller

We begin by restating Cor. 12 from [9]:

Lemma 5. *Suppose there exists a $\sigma_m > 0$ such that $0 < \sigma_m \leq \lambda_m(M(q(t),\Delta))$ for all $q(t) \in Q$ and $\Delta \in [\Delta]$. Let $e(t;k) := q_d(t;k) - q(t;k)$. Let $e(0) = \dot{e}(0) = 0$, and let $V_M > 0$ and $K_m \geq 0$ be user-specified constants. Define*

$$\varepsilon_p := \frac{1}{K_r}\sqrt{\frac{2V_M}{\sigma_m}} \quad \text{and} \quad \varepsilon_v := 2\sqrt{\frac{2V_M}{\sigma_m}}. \tag{10}$$

Then there exists a feedback controller that when applied to the extended arm dynamics (1) ensure that $|e_j(t)| \leq \varepsilon_p$ and $|\dot{e}_j(t)| \leq \varepsilon_v$ for all $t \in T$ and $j \in N_e$.

We make two important remarks about the feedback controller. First, we compute σ_m numerically and verify that it is greater than zero during experiments. Second, note that one can compute the feedback controller by applying a variant of the Recursive Newton Euler Algorithm (RNEA) [9, Sec. VII-B]. To bound the error for all time rather than just over $t \in T$ one can apply the previous lemma and [9, Rem. 13] which we summarize here for convenience:

Remark 6. *Suppose that the initial condition of the desired trajectory in the first planning iteration matches the actual state of the robot, and that the initial condition of all subsequent desired trajectories match the state of the previous desired trajectory at $t = t_p$. Then, one can satisfy the bounds described by Lem. 5.*

Note that to apply Rem. 6, one only needs to know the final state of the previous *desired trajectory* rather than the final state of the robot's *actual trajectory*.

B. Polynomial Zonotope Overapproximation

This subsection describes how WAITR uses Lem. 5 to overapproximate parameterized desired trajectories to conservatively account for continuous time operation and tracking error within our optimization framework. The desired trajectories $q_d(t;k)$ are functions of only time t and the trajectory parameter k. As illustrated in Fig. 3, our approach creates polynomial zonotope versions of T and K, which are then plugged into the formula for $q_d(t;k)$ to create polynomial zonotope overapproximations which are then extended to the joint trajectory, the force trajectory, and subsequently the tipping constraint for the manipulation problem.

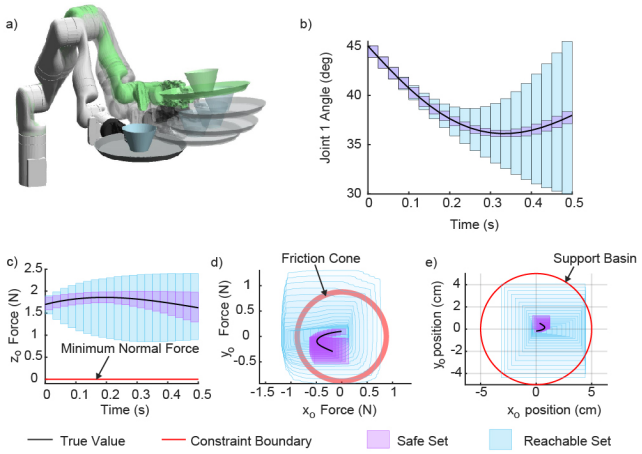

Fig. 3. An illustration of the polynomial zonotope overapproximations of the configuration, force, and constraint trajectories. The motion of the robot is shown in intermediate configurations the top of the figure. Blue indicates unsliced reachable sets, purple indicates sliced reachable sets and black lines are the nominal values. Red indicates a constraint boundary.

1) Time Horizon and Trajectory Parameter PZs: We first create polynomial zonotopes representing the time horizon T. Choose a timestep Δt so that $n_t := \frac{T}{\Delta t} \in \mathbb{N}$. Let $N_t := \{1,\ldots,n_t\}$. Divide the compact time horizon $T \subset \mathbb{R}$ into n_t time subintervals. We represent the i^{th} time subinterval corresponding to $t \in [(i-1)\Delta t, i\Delta t]$ as a polynomial zonotope $\mathbf{T_i} = \left\{ t \mid t = \frac{(i-1)+i}{2}\Delta t + \frac{1}{2}\Delta t x_{t_i},\ x_{t_i} \in [-1,1] \right\}$. with indeterminate $x_{t_i} \in [-1,1]$.

In this work, we choose $K = \bigtimes_{j=1}^{n_q} K_j$, where each K_j is a compact one-dimensional interval. For simplicity, let each $K_j = [-1,1]$. We represent the interval K_j as a polynomial zonotope $\mathbf{K_j} = x_{k_j}$ where $x_{k_j} \in [-1,1]$ is an indeterminate. Let $x_k \in [-1,1]^{n_q}$ denote the vector of indeterminates where the j^{th} component of x_k is x_{k_j}. With this choice of K_j, any particular k_j directly yields $k_j = \texttt{slice}(\mathbf{K_j}, x_{k_j}, k_j)$ (see (A14)[2]).

2) Trajectory PZs: Recall that $\mathbf{T_i}$ and $\mathbf{K_j}$ have indeterminates x_{t_i} and x_{k_j}, respectively. Because the desired trajectories only depend on t and k, the polynomial zonotopes $\mathbf{q_{d,j}}(\mathbf{T_i};\mathbf{K})$, $\dot{\mathbf{q}}_{\mathbf{d,j}}(\mathbf{T_i};\mathbf{K})$ and $\ddot{\mathbf{q}}_{\mathbf{d,j}}(\mathbf{T_i};\mathbf{K})$ depend only on the indeterminates x_{t_i} and x_k. By plugging in a given k for x_k via the `slice` operation, we obtain a polynomial zonotope where x_{t_i} is the only remaining indeterminate. Because we perform this particular slicing operation repeatedly throughout this document, if we are given a polynomial zonotope, $\mathbf{q_{d,j}}(\mathbf{T_i};\mathbf{K})$, we use the shorthand $\mathbf{q_{d,j}}(\mathbf{T_i};k) = \texttt{slice}(\mathbf{q_{d,j}}(\mathbf{T_i};\mathbf{K}), x_k, k)$. Because of our previous observation, $q_{d,j}(t;k) \in \mathbf{q_{d,j}}(\mathbf{T_i};k)$ for all $t \in \mathbf{T_i}$.

Next, we use Lem. 5 to generate polynomial zonotopes that overapproximate any trajectory that is followed by the robot.

[2]Supplementary Appendices found at https://roahmlab.github.io/waitr-dev/

Lemma 7. *Let* $\boldsymbol{\varepsilon}_{\mathbf{p,j}} = \varepsilon_{p,j} x_{e_{p,j}}$ *and* $\boldsymbol{\varepsilon}_{\mathbf{v,j}} = \varepsilon_v x_{e_{v,j}}$*, with indeterminates* $x_{e_{p,j}} \in [-1,1]$ *and* $x_{e_{v,j}} \in [-1,1]$*. Then, let*

$$\mathbf{q_j}(\mathbf{T_i};\mathbf{K}) := \mathbf{q_{d,j}}(\mathbf{T_i};\mathbf{K}) \oplus \boldsymbol{\varepsilon}_{\mathbf{p,j}} \tag{11}$$

$$\dot{\mathbf{q}}_{\mathbf{j}}(\mathbf{T_i};\mathbf{K}) := \dot{\mathbf{q}}_{\mathbf{d,j}}(\mathbf{T_i};\mathbf{K}) \oplus \boldsymbol{\varepsilon}_{\mathbf{v,j}} \tag{12}$$

for each $i \in N_t$*. The polynomial zonotopes* $\mathbf{q_j}(\mathbf{T_i};\mathbf{K})$ *and* $\dot{\mathbf{q}}_{\mathbf{j}}(\mathbf{T_i};\mathbf{K})$ *overapproximate the set of all joint trajectories that can be executed by the robot, i.e., for each* $k \in \mathbf{K}$ *and* $j \in N_e$*,* $q_j(t;k) \in \mathbf{q_j}(\mathbf{T_i};k)$ *and* $\dot{q}_j(t;k) \in \dot{\mathbf{q}}_{\mathbf{j}}(\mathbf{T_i};k), \forall t \in \mathbf{T_i}$*.*

Moving forward, let $\mathbf{q_d}(\mathbf{T_i};\mathbf{K}) = \bigtimes_{j=1}^{n_q} \mathbf{q_{d,j}}(\mathbf{T_i};\mathbf{K})$ and $\mathbf{q}(\mathbf{T_i};\mathbf{K}) = \bigtimes_{j=1}^{n_q} \mathbf{q_j}(\mathbf{T_i};\mathbf{K})$ We similarly define $\dot{\mathbf{q}}_{\mathbf{d}}(\mathbf{T_i};\mathbf{K})$, $\dot{\mathbf{q}}(\mathbf{T_i};\mathbf{K})$, $\ddot{\mathbf{q}}_{\mathbf{d}}(\mathbf{T_i};\mathbf{K})$, and $\ddot{\mathbf{q}}(\mathbf{T_i};\mathbf{K})$. Furthermore, let $\mathbf{E_p} = \bigtimes_{j=1}^{n_q} \boldsymbol{\varepsilon}_{\mathbf{p,j}}$ and $\mathbf{E_v} = \bigtimes_{j=1}^{n_q} \boldsymbol{\varepsilon}_{\mathbf{v,j}}$.

After generating the polynomial zonotope overapproximations of joint trajectories in the previous subsection, WAITR composes these together to compute a polynomial zonotope overapproximation of the contact wrench by applying Alg. 1. This algorithm is based on RNEA, which is a tool to compute the dynamics of serial chain manipulators. Alg. 1 is a polynomial zonotope version of the RNEA, and was originally proposed in [9]. Importantly, because of the extended arm modeling choice described in Sec. III, this same algorithm now computes an overapproximation of the contact wrench. Using [9, Lem. 7] and the fact that the vertical separation, linear slip, and tip constraints are all polynomial functions, one can prove the following result:

Lemma 8. *Let* $\{\mathbf{w_j^j}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta])\}_{j\in N_e} = \texttt{PZRNEA}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta])$ *for each* $i \in N_t$*. Then the vertical separation, linear slip, and tip constraints can be overapproximated by polynomial zonotopes using* $\mathbf{w}_{\mathbf{n_q+1}}^{\mathbf{n_q+1}}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta])$*, i.e., for each* $k \in \mathbf{K}$

$$h_{sep}(\mathrm{w}_o(q_A(t;k,\Delta))) \in \mathbf{h_{sep}}(\mathbf{w}_{\mathbf{n_q+1}}^{\mathbf{n_q+1}}(\mathbf{q_A}(\mathbf{T_i};k),[\Delta])) \tag{13}$$

$$h_{slip}(\mathrm{w}_o(q_A(t;k,\Delta))) \in \mathbf{h_{slip}}(\mathbf{w}_{\mathbf{n_q+1}}^{\mathbf{n_q+1}}(\mathbf{q_A}(\mathbf{T_i};k),[\Delta])) \tag{14}$$

$$h_{tip}(\mathrm{w}_o(q_A(t;k,\Delta))) \in \mathbf{h_{tip}}(\mathbf{w}_{\mathbf{n_q+1}}^{\mathbf{n_q+1}}(\mathbf{q_A}(\mathbf{T_i};k),[\Delta])), \tag{15}$$

$\forall t \in \mathbf{T_i}$ *and* $\Delta \in [\Delta]$.

C. Implementable Online Optimization Problem

Using the polynomial zonotope overapproximations of the vertical separation, linear slip, and tip constraints, one can construct an representation of (6)-(9):

$$\min_{k\in\mathbf{K}} \quad \texttt{cost}(k) \tag{16}$$

$$\texttt{sup}(\mathbf{h_{sep}}(\mathbf{w}_{\mathbf{n_q+1}}^{\mathbf{n_q+1}}(\mathbf{q_A}(\mathbf{T_i};k),[\Delta])) \leq 0 \qquad \forall i \in \mathbb{N}_t \tag{17}$$

$$\texttt{sup}(\mathbf{h_{slip}}(\mathbf{w}_{\mathbf{n_q+1}}^{\mathbf{n_q+1}}(\mathbf{q_A}(\mathbf{T_i};k),[\Delta])) \leq 0 \qquad \forall i \in \mathbb{N}_t \tag{18}$$

$$\texttt{sup}(\mathbf{h_{tip}}(\mathbf{w}_{\mathbf{n_q+1}}^{\mathbf{n_q+1}}(\mathbf{q_A}(\mathbf{T_i};k),[\Delta])) \leq 0 \qquad \forall i \in \mathbb{N}_t. \tag{19}$$

Using Lem. 8 one can prove the following theorem about this optimization problem.

Theorem 9. *Any feasible solution* k *to the optimization problem described in* (16)-(19) *parameterizes a trajectory that results in no relative motion between an unsecured object and the end-effector of the robot over the time horizon* T*.*

Algorithm 1: $\{\mathbf{w_j^j}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta])\}_{j\in N_e} = \texttt{PZRNEA}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta])$

1: Let $\dot{\mathbf{q}}_{\mathbf{a}}(\mathbf{T_i};\mathbf{K}) := \mathbf{q_d}(\mathbf{T_i};\mathbf{K}) \oplus K_r \mathbf{E_p}$.
2: Let $\ddot{\mathbf{q}}_{\mathbf{a}}(\mathbf{T_i};\mathbf{K}) := \dot{\mathbf{q}}_{\mathbf{d}}(\mathbf{T_i};\mathbf{K}) \oplus K_r \mathbf{E_v}$.
3: Let $a_0^0 = \begin{bmatrix} 0 & 0 & 9.81 \end{bmatrix}^\top$.
4: **parfor** $i \in N_t$
5: **for** $j = 1 : n_q + 1$
6: $\mathbf{R_j^{j-1}}, \mathbf{p_j^{j-1}} \leftarrow \mathbf{S_j^{j-1}}(\mathbf{q_j}(\mathbf{T_i};\mathbf{K}))$
7: $\mathbf{R_{j-1}^j} \leftarrow \texttt{pzTranspose}(\mathbf{R_j^{j-1}})$
8: **end for**
9: **for** $j = 1 : n_q + 1$
10: $\omega_{\mathbf{j}}^{\mathbf{j}} \leftarrow \mathbf{R_{j-1}^j} \omega_{\mathbf{j-1}}^{\mathbf{j-1}} \oplus \dot{\mathbf{q}}_{\mathbf{j}} z_j$
11: $\omega_{\mathbf{a,j}}^{\mathbf{j}} \leftarrow \mathbf{R_{j-1}^j} \omega_{\mathbf{a,j-1}}^{\mathbf{j-1}} \oplus \dot{\mathbf{q}}_{\mathbf{a,j}} z_j$
12: $\dot{\omega}_{\mathbf{j}}^{\mathbf{j}} \leftarrow \mathbf{R_{j-1}^j} \dot{\omega}_{\mathbf{j-1}}^{\mathbf{j-1}} \oplus ((\mathbf{R_{j-1}^j} \omega_{\mathbf{a,j}}^{\mathbf{j}}) \otimes (\dot{\mathbf{q}}_{\mathbf{j}} z_j)) \oplus \ddot{\mathbf{q}}_{\mathbf{a,j}} z_j$
13: $\mathbf{a_j^j} \leftarrow (\mathbf{R_{j-1}^j} \mathbf{a_{j-1}^{j-1}}) \oplus (\dot{\omega}_{\mathbf{j}}^{\mathbf{j}} \otimes \mathbf{p_j^{j-1}}) \oplus (\omega_{\mathbf{j}}^{\mathbf{j}} \otimes (\omega_{\mathbf{a,j}}^{\mathbf{j}} \otimes \mathbf{p_j^{j-1}}))$
14: $\mathbf{a_{CoM,j}^j} \leftarrow \mathbf{a_j^j} \oplus (\dot{\omega}_{\mathbf{j}}^{\mathbf{j}} \otimes \mathbf{p_{CoM,j}^j}) \oplus (\omega_{\mathbf{j}}^{\mathbf{j}} \otimes (\omega_{\mathbf{a,j}}^{\mathbf{j}} \otimes \mathbf{p_{CoM,j}^j}))$
15: $\mathbf{F_j^j} \leftarrow \mathbf{m_i a_{CoM,j}^j}$
16: $\mathbf{N_j^j} \leftarrow \mathbf{I_j^j} \dot{\omega}_{\mathbf{j}}^{\mathbf{j}} \oplus (\omega_{\mathbf{a,j}}^{\mathbf{j}} \otimes (\mathbf{I_j^j} \omega_{\mathbf{j}}^{\mathbf{j}}))$
17: **end for**
18: Initialize $\mathbf{f}_{\mathbf{n_q+2}}^{\mathbf{n_q+2}}, \mathbf{n}_{\mathbf{n_q+2}}^{\mathbf{n_q+2}}, \mathbf{R}_{\mathbf{n_q+2}}^{\mathbf{n_q}}$
19: **for** $j = n_q + 1 : -1 : 1$
20: $\mathbf{f_j^j}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta]) \leftarrow (\mathbf{R_{j+1}^j} \mathbf{f_{j+1}^{j+1}}) \oplus \mathbf{F_j^j}$
21: $\mathbf{n_j^j}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta]) \leftarrow (\mathbf{R_{j+1}^j} \mathbf{n_{j+1}^{j+1}}) \oplus (\mathbf{p_{CoM,j}^j} \otimes \mathbf{F_j^j}) \oplus (\mathbf{p_{j+1}^j} \otimes (\mathbf{R_{j+1}^j} \mathbf{f_{j+1}^{j+1}})) \oplus \mathbf{N_j^j}$
22: $\mathbf{w_j^j}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta]) \leftarrow \begin{bmatrix} \mathbf{f_j^j}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta]) \\ \mathbf{n_j^j}(\mathbf{q_A}(\mathbf{T_i};\mathbf{K}),[\Delta]) \end{bmatrix}$
23: **end for**
24: **end parfor**

The implementation of this optimization problem is summarized in Alg. 2. Note in particular that we apply Rem. 6 and Lem. 7 to compute the polynomial zonotope overapproximations of the parameterized trajectories and the actual trajectory of the system for each time interval $i \in N_t$ in parallel in Line 2. In addition, note that one can calculate the analytical gradients of the cost function and constraints and provide them to an optimization solver. One can use polynomial differentiation of the polynomial zonotope representations of the constraints to compute the required gradients [9, Sec. IX-C]. In conjunction with ARMOUR, additional constraints can be added to the optimization problem such that the parameterized trajectory is guaranteed collision-free and can be tracked while satisfying joint and input limits [9, Sec. IX-B]. Thus, a solution to this optimization problem results in a safe manipulation trajectory.

D. WAITR's Online Operation

The online operation of WAITR is summarized in Alg. 3. Because WAITR operates in receding horizon fashion, the controller in Lem. 5 is used to track the trajectory parameter computed during the previous planning iteration on Line 7. At the same time, WAITR computes the trajectory parameter for the following planning iteration on Line 8. Recall that if a

Algorithm 2: $k^* = \texttt{Opt}(q_{d_0}, \dot{q}_{d_0}, \ddot{q}_{d_0}, \texttt{cost}, t_p, N_t, \Delta_0, [\Delta], \varepsilon_p, \varepsilon_v)$

1: **Parfor** $i = 1 : N_t$
2: $\{\mathbf{q}(\mathbf{T_i}; \mathbf{K}), \ldots, \ddot{\mathbf{q}}_\mathbf{d}(\mathbf{T_i}; \mathbf{K})\} \leftarrow \texttt{PZ}(q_{d_0}, \dot{q}_{d_0}, \ddot{q}_{d_0})$ // Sec. IV-B2
3: // object wrench reachable set using Alg. 1 //
4: $\{\mathbf{w}_\mathbf{j}^\mathbf{j}(\mathbf{q_A}(\mathbf{T_i}; \mathbf{K}), [\Delta])\}_{j \in N_e} = \texttt{PZRNEA}(\mathbf{q_A}(\mathbf{T_i}; \mathbf{K}), [\Delta])$
5: // constraints using Lem. 8 and $\mathbf{w}_{\mathbf{n_q}+\mathbf{1}}^{\mathbf{n_q}+\mathbf{1}}(\mathbf{q_A}(\mathbf{T_i}; \mathbf{K}), [\Delta])$ //
6: **Compute** $\mathbf{h_{sep}}, \mathbf{h_{slip}}$, and $\mathbf{h_{tip}}$
7: **End Parfor**
8: **Try:** $k^* \leftarrow$ solve (16) – (19)
9: **Catch:** (if $t_e > t_p$), **then** $k^* = \texttt{NaN}$ // t_e measures the amount of time since `Opt` was called //

Algorithm 3: WAITR Online Planning and Control

1: **Require:** $t_p > 0$, $N_t \in \mathbb{N}, [\Delta], \Delta_0 \in [\Delta]$, $\texttt{cost} : K \to \mathbb{R}$,, ε_p, ε_v
2: **Initialize:** $l = 0$, $t_j = 0$, and $k_l^* = \texttt{Opt}(q_{\text{start}}, 0, 0, \mathscr{O}, \texttt{cost}, t_p, N_t, \Delta_0, [\Delta], \varepsilon_p, \varepsilon_v)$
3: **If** $k_l^* = \texttt{NaN}$, **then** execute brake
4: **Loop:**
5: // Line 7 executes simultaneously with Lines 8 – 10 //
6: // Use Lem. 5 //
7: **Apply** feedback controller to robot for $t \in [t_l, t_l + t_p]$
8: $k_{l+1}^* = \texttt{Opt}(q_d(t_p; k_l^*), \dot{q}_d^j(t_p; k_l^*), \ddot{q}_d^j(t_p; k_l^*), \texttt{cost}, t_p, N_t, \Delta_0, [\Delta], \varepsilon_p, \varepsilon_v)$
9: **If** $k_{l+1}^* = \texttt{NaN}$, **then** execute brake and break loop
10: **Else** $t_{l+1} \leftarrow t_l + t_p$ and $j \leftarrow j + 1$
11: **End**

new trajectory parameter is not found before time t_p, then the robust controller tracks the braking maneuver of the previous trajectory to bring the robot to a safe stop, seen on Line 9. Further, by applying Lem. 5 and Thm. 9, one can prove that WAITR generates behavior that is dynamically feasible and results in no relative motion.

V. Experiments

This section details the implementation and testing of the WAITR framework on a Kinova Gen3 7 DOF robotic arm. WAITR is implemented in C++ and CUDA and a MEX version is used in MATLAB for simulation experiments. The code can be found at https://github.com/roahmlab/waitr-dev. Simulation experiments were run using a AMD Ryzen 9 5950x processor and an NVIDIA RTX A6000 GPU. Two hardware experiments were run using an AMD Ryzen 9 3950x processor and NVIDIA Quadro RTX 8000 GPU.

A. Trajectory Creation

This work parameterizes the reference trajectories by using a set of degree 5 Bernstein polynomials as is done in [9, Sec. IX-A]. Letting $T = [t_0, t_f]$, WLOG, the reference trajectories take the form $q_{d,j}(t; k) = \sum_{l=0}^{5} \beta_{j,l} b_{j,l}(t)$, where $\beta_{j,l} \in \mathbb{R}$ are the Bernstein Coefficients and $b_{j,l} : T \to \mathbb{R}$ are the Bernstein Basis Polynomials of degree 5 for each $l \in \{0, \ldots, 5\}$. As a result of Rem. 6, the initial position, velocity, and acceleration of a parameterized trajectory are constrained. Further, the parameterized trajectories are constrained to have zero velocity and acceleration by t_f in order to bring the robot to a stop. Therefore, one can show that five of the six Bernstein coefficients $\beta_{j,v}$ are known and only the last coefficient can be optimized over. We let the last coefficient $\beta_{j,5} = \eta_{j,1} k_j + \eta_{j,2}$ where $\eta_{j,1}$ and $\eta_{j,2} \in \mathbb{R}$ are user-specified constants. The choice of trajectory parameter k_j directly determines the coefficient $\beta_{j,5}$ and the final position of the robot.

B. Implementation Details

1) Robot Model and Environment: For simulation, the tray mass was 0.044 kg and the object mass was 0.172 kg. For the first hardware experiment, the tray mass was 0.058 kg and the object mass was 0.048 kg. For the second hardware experiment, the tray mass was 0.354 kg and the object mass was 0.592 kg. These are added to the robot model as described in Sec. III. By sampling, the minimum eigenvalue of the mass matrix was determined to be uniformly bounded from below by $\sigma_m = 8.0386$.

2) Trajectories: In all experiments, $t_p = 1.0$s and $t_f = 2.0$s. In simulation experiments, we let $\eta_{j,2} = q_{d,j_0}$ and $\eta_{j,1} = \frac{\pi}{72}$, so that after $t_f = 2.0$s the final position of any joint trajectory can differ from its initial position by up to $\pm\frac{\pi}{72}$ radians. For hardware experiments, we let $\eta_{j,2} = q_{d,j_0}$ and $\eta_{j,1} = \frac{\pi}{32}$ for $j \in 1, 2, 6$ and $\eta_{j,1} = \frac{\pi}{72}$ for $j \in 3, 4, 5, 7$.

3) Tracking Error Bound: We use the same controller as presented in [9, Sec. VII], with a different K_r, V_M and σ_m, which are reported in App. D[3].

4) High-level Planners: We use a cost function that minimizes the distance between $q(t_p; k)$ and an intermediate waypoint. Simulation experiments are run with two different high level planners. The first is a straight line high-level planner (SL-HLP), which calculates a waypoint along a straight line between the start and goal in configuration space without checking for collision. The second is a graph-based high level planner (GB-HLP), which is constructed using robot configurations with a flat end-effector pose. Before operation, the configurations that are in collision with obstacles are removed from the graph. Note that the high-level planner does not need safety guarantees because WAITR's safety guarantees are independent of the high-level planner.

5) Comparison Framework: We compare WAITR to a previous method ARMOUR, which does not contain contact wrench constraints, by running both methods on identical simulation worlds with the same high level planner.

6) Trajectory Optimization Implementation: The WAITR C++ framework uses IPOPT [16] to solve the trajectory optimization problem during each planning iteration. Analytic gradients/subgradients of the cost function and constraints are provided for the optimization solver to speed computation.

C. Simulation Experiments

1) Simulation Setup: The simulation experiment consisted of testing WAITR with two different high-level planners in 100 trials where each was initialized with a random feasible start and goal state. Each trial had 10 box shaped obstacles randomly placed in the workspace and randomly scaled side lengths that were allowed to vary between 0.010 to 0.050 m.

Method	Goals Reached	Safe Stops	Violations
WAITR + SL-HLP	44	56	**0**
WAITR + GB-HLP	**91**	**9**	**0**
ARMOUR [9] + GB-HLP	64	30	6

TABLE II
Results on 100 random simulation experiments.

2) Results: The results of the 100 random trials are presented in Tab. II, with example videos available at https://roahmlab.github.io/waitr-dev/. WAITR robustly handles infeasible waypoints from the SL-HLP while still ensuring safety, at the cost of not always making it to the goal. When using the GB-HLP, which provides feasible waypoints, WAITR reaches the goal 91 times and safely stops 9 times. In contrast, even though ARMOUR uses the GB-HLP, and is given feasible waypoints, it reaches the goal only 64 times, safely stops 30 times, and violates the contact constraints 6 times. This shows that it is not adequate to have a high-level planner that can provide feasible waypoints.

D. Hardware Results

1) Setup: The first experiment compares WAITR and ARMOUR on a Kinova Gen3 robot arm. The tray surface is covered with Duct tape to adjust the coefficient of friction, which was experimentally measured to be $\mu = 0.36$. Both frameworks used the GB-HLP, and had the same random noise added to the last three joints in the waypoints selected. This randomly adjusted the orientation of the tray in each waypoint such that it was not flat. Two obstacles were placed randomly in the workspace but at the same location when running each approach. This ensures that the only difference between the experiments for the two frameworks is the constraints that they are enforcing. The second experiment demonstrates the ability of WAITR to operate under the presence of modeling uncertainty. The object for this experiment was a plastic cup filled with metal nuts. The coefficient of friction between the plastic cup and the tray was measured to be $\mu = 0.60$. WAITR was run in three consecutive trials which each had random placed obstacles. Before the last trial, the mass of the cup was adjusted by removing 0.028 kg, or 4.73% of the total mass of the unsecured object. This also adjusted the inertia by 4.73%. WAITR was set to account for 5% variation in the mass and inertia of the object, and the model used by the WAITR framework was not adjusted to reflect the changed object parameters.

2) Results: Videos of the hardware experiments are available at https://roahmlab.github.io/waitr-dev/. The first hardware experiment shows WAITR successfully manipulating an unsecured object while navigating around two randomly placed obstacles. ARMOUR also successfully navigates around the obstacles, but fails to manipulate the unsecured object, resulting in the object slipping and falling off the tray. The second trial demonstrates that WAITR can successfully operate in the presence of uncertainty in the inertial parameters of the object being manipulated. All of the hardware experiments were run in real-time.

[3]Supplementary Appendices found at https://roahmlab.github.io/waitr-dev/

VI. Conclusion

This paper describes WAITR, a real-time provably safe planning and control framework for collision-free non-prehensile manipulation capable of robustly dealing with uncertainty in both the manipulator's and target object's inertial parameters. There are several directions of future work for the WAITR framework. The first is enabling WAITR to handle changes in the contact state of the objects being manipulated, such as allowing manipulated objects to slip on the tray. The second is to form constraints that can guarantee safe force closure on objects throughout a desired trajectory.

References

[1] F. Debrouwere, W. Van Loock, G. Pipeleers, M. Diehl, J. Swevers, and J. De Schutter, "Convex time-optimal robot path following with Cartesian acceleration and inertial force and torque constraints," *undefined*, vol. 227, no. 10, pp. 724–732, 2013.

[2] G. Csorvási, Á. Nagy, and I. Vajk, "Near Time-Optimal Path Tracking Method for Waiter Motion Problem," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4929–4934, Jul. 2017.

[3] H. Gattringer and A. Müller, "Reliable Time-Optimal Point to Point Handling of Unmounted Objects with Industrial Robots," pp. 210–219, 2022.

[4] H. Gattringer, A. Mueller, M. Oberherber, and D. Kaserer, "Time-optimal robotic manipulation on a predefined path of loosely placed objects: Modeling and experiment," *Mechatronics*, vol. 84, p. 102 753, Jun. 2022.

[5] F. G. Flores and A. Kecskeméthy, "MMS 14 - Time-Optimal Path Planning for the General Waiter Motion Problem,"

[6] J. Ichnowski, Y. Avigal, Y. Liu, and K. Goldberg, "Gomp-fit: Grasp-optimized motion planning for fast inertial transport," *arXiv preprint arXiv:2110.15326*, 2021.

[7] Y. Avigal, J. Ichnowski, M. Y. Cao, and K. Goldberg, "Gomp-st: Grasp optimized motion planning for suction transport," in *International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2023, pp. 488–505.

[8] J. Luo and K. Hauser, "Robust trajectory optimization under frictional contact with iterative learning," *Autonomous Robots*, vol. 41, no. 6, pp. 1447–1461, Aug. 2017.

[9] J. Michaux, P. Holmes, B. Zhang, *et al.*, "Can't touch this: Real-time, safe motion planning and control for manipulators under uncertainty," *arXiv preprint*, 2023.

[10] T. Hickey, Q. Ju, and M. H. Van Emden, "Interval arithmetic: From principles to implementation," *Journal of the ACM (JACM)*, vol. 48, no. 5, pp. 1038–1068, 2001.

[11] M. Spong, S. Hutchinson, and M. Vidyasagar, "Robot modeling and control," 2005.

[12] J. J. Craig, P. Prentice, and P. P. Hall, "Introduction to Robotics Mechanics and Control Third Edition," 2005.

[13] P. Holmes, S. Kousik, B. Zhang, *et al.*, " Reachable Sets for Safe, Real-Time Manipulator Trajectory Design," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, Jul. 2020.

[14] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *The International Journal of Robotics Research*, vol. 39, no. 12, pp. 1419–1469, 2020.

[15] S. Kousik, P. Holmes, and R. Vasudevan, "Safe, aggressive quadrotor flight via reachability-based trajectory design," in *Dynamic Systems and Control Conference*, American Society of Mechanical Engineers, vol. 59162, 2019, V003T19A010.

[16] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006.