Fast Nonprehensile Object Transportation on a Mobile Manipulator

Adam Heins and Angela P. Schoellig

***Abstract*— We consider a nonprehensile manipulation task in which a mobile manipulator must balance objects on its end effector without grasping them—known as the *waiter's problem*—while moving to a desired location. In contrast to existing approaches, our focus is on fast online planning in response to new and changing environments. Our main contribution is a whole-body constrained model predictive controller (MPC) for a mobile manipulator that balances objects and avoids collisions. Furthermore, we propose planning using the minimum statically-feasible friction coefficients, which provides robustness to frictional uncertainty and other force disturbances while also substantially reducing the compute time required to update the MPC policy. Notably, we demonstrate a projectile avoidance task in which our mobile manipulator avoids a thrown ball while balancing a tall bottle.**

I. Introduction

We consider the nonprehensile object transportation task known as the *waiter's problem* [1], which requires the robot to transport objects from one location to another while keeping them balanced on a tray at the end effector (EE), like a restaurant waiter. *Nonprehensile* manipulation [2] refers to the case when the manipulated objects are subject only to unilateral constraints [3] and thus retain some degrees of freedom (DOFs); that is, they are not fully grasped. In contrast to *prehensile* manipulation, a nonprehensile approach allows the robot to carry many objects at once with a simple, non-articulated EE (e.g., a tray; see Fig. 1). Furthermore, a nonprehensile approach skips the potentially slow grasping and ungrasping processes, and can handle small or delicate objects which cannot be adequately grasped [4]. Beyond food service, efficient object transportation is useful across many industries, such as warehouse fulfilment and manufacturing.

Specifically, we address the waiter's problem using a wheeled mobile manipulator. Mobile manipulators are capable of performing a wide variety of tasks due to the combination of the large workspace of a mobile base and the manipulation capabilities of robotic arms. We are particularly interested in having the mobile manipulator move and react *quickly*, whether to avoid obstacles or simply for efficiency. However, a challenge of *mobile* manipulation is that moving across the ground causes vibration at the EE, which requires our object balancing strategy to be robust to such disturbances. We assume that the geometry, inertial properties, and initial poses of the objects are known, but we do not assume that feedback of the objects' poses is available online. A full version of this paper is available in [5].

The authors are with the Learning Systems and Robotics Lab (www.learnsyslab.org) at the Technical University of Munich, Germany, and the University of Toronto Institute for Aerospace Studies, Canada. They are also affiliated with the University of Toronto Robotics Institute, the Munich Institute of Robotics and Machine Intelligence (MIRMI), and the Vector Institute for Artificial Intelligence. E-mail: adam.heins@robotics.utias.utoronto.ca, angela.schoellig@tum.de

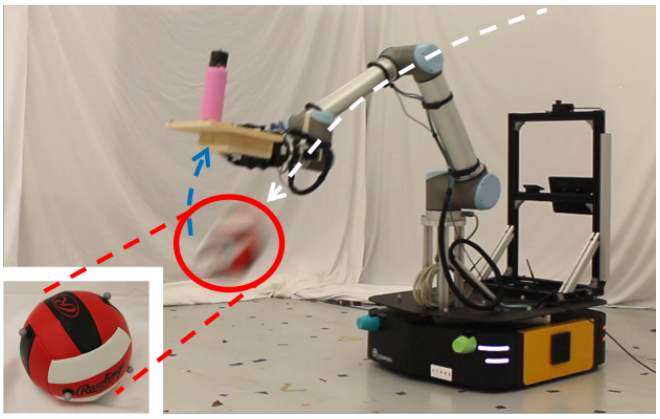

Fig. 1: Our mobile manipulator balancing a pink bottle while avoiding a thrown volleyball (ball circled in red with approximate trajectory in white; approximate end effector trajectory in blue). The controller has less than $0.75\,\mathrm{s}$ between first observing the ball and a potential collision. A video of our experiments is available at **`http://tiny.cc/keep-it-upright`**.

A. Contributions

We propose the first whole-body model predictive controller (MPC) for a mobile manipulator solving the waiter's problem. Furthermore, the controller uses the minimum statically-feasible friction coefficients, which provides robustness to frictional uncertainty, vibration, and other real-world disturbances. When the minimum statically-feasible friction coefficients are *zero*, we show that the MPC problem can be solved more efficiently. We also demonstrate experiments on a real velocity-controlled mobile manipulator balancing up to seven objects; balancing an assembly of stacked objects; and avoiding a thrown volleyball (see Fig. 1). Our code is available at `https://github.com/utiasDSL/upright`.

B. Related Work

Prior examples of robots directly inspired by waiters in a restaurant include [6]–[8], but these are mobile robots without manipulator arms. In contrast, a mobile *manipulator* has additional DOFs that provide redundancy and a larger workspace, at the cost of requiring a larger and more computationally-demanding control problem. The waiter's problem has also been addressed using offline motion planning [1], [4], [9], [10], but these approaches cannot react quickly to changes in the environment. In [11]–[13], a reactive controller automatically regulates the commanded motion to ensure the object remains balanced. One of the only works to use a full mobile manipulator (a quadruped) for the waiter's problem is [13], but it is demonstrated only in simulation and does not consider dynamic obstacles. To our knowledge, the only physical mobile manipulator experiments for the waiter's problem have been performed on a humanoid in [14], but they focus on the detection and rejection of disturbances to the object's stability rather than fast object transportation.

Finally, like us, some recent works use MPC to address the waiter's problem. In [15], a dual-arm approach is proposed in which a time-optimal trajectory is planned and MPC is used to compute the applied wrench required to realize the object's trajectory. Another MPC approach is described in [16], which is designed to track a manipulator's joint-space reference trajectory. In contrast, our MPC approach optimizes the joint-space trajectory online while considering task-space objectives and constraints, which allows us to respond quickly to changes in the environment like dynamic obstacles, and we also show how reducing the friction coefficients in the controller constraints can provide robustness and computational savings.

II. Methodology

In this section we present the system model and controller.

A. System Model

1) Robot Model: We consider a velocity-controlled mobile manipulator with state $\boldsymbol{x} = [\boldsymbol{q}^T, \boldsymbol{v}^T, \dot{\boldsymbol{v}}^T]^T$, where $\boldsymbol{q}$ is the generalized position, which includes the pose of the mobile base and the arm's joint angles, and $\boldsymbol{v}$ is the generalized velocity. We include acceleration in the state and take the input $\boldsymbol{u}$ to be jerk, which ensures a continuous acceleration profile [16]. The input is double-integrated to obtain the velocity commands sent to the actual robot. We require only a kinematic model, which we represent generically as $\dot{\boldsymbol{x}} = \boldsymbol{a}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}$, with $\boldsymbol{a}(\boldsymbol{x}) \in \mathbb{R}^{\dim(\boldsymbol{x})}$ and $\boldsymbol{B}(\boldsymbol{x}) \in \mathbb{R}^{\dim(\boldsymbol{x})\times\dim(\boldsymbol{u})}$.

2) Object Model: We model each object $\mathcal{O}$ as a rigid body subject to the Newton-Euler equations

$$\boldsymbol{w}_{\mathrm{C}} + \boldsymbol{w}_{\mathrm{GI}} = \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{w}_{\mathrm{C}}$ is the total contact wrench and $\boldsymbol{w}_{\mathrm{GI}}$ is the gravito-inertial wrench, expressed in the body frame as

$$\boldsymbol{w}_{\mathrm{GI}} \triangleq \begin{bmatrix}\boldsymbol{f}_{\mathrm{GI}}\\ \boldsymbol{\tau}_{\mathrm{GI}}\end{bmatrix} = -\begin{bmatrix} m(\dot{\boldsymbol{v}}_o - \boldsymbol{R}_o\boldsymbol{g}) \\ \boldsymbol{J}\dot{\boldsymbol{\omega}}_o + \boldsymbol{\omega}_o \times \boldsymbol{J}\boldsymbol{\omega}_o\end{bmatrix},$$

where $\boldsymbol{f}_{\mathrm{GI}}$ and $\boldsymbol{\tau}_{\mathrm{GI}}$ are the gravito-inertial force and torque, m is the object's mass, $\boldsymbol{v}_o$ and $\boldsymbol{\omega}_o$ are the body-frame linear and angular velocity of the object's CoM, $\boldsymbol{g}$ is the gravitational acceleration, $\boldsymbol{J}$ is the object's inertia matrix taken about the CoM, and $\boldsymbol{R}_o$ is the rotation matrix representing the object's orientation with respect to the world. We assume that m, $\boldsymbol{c}$, and $\boldsymbol{J}$ are known.

B. Nominal Balancing Constraints

Our approach for balancing objects is to generate motion so that objects are in a *dynamic grasp* [2]—that is, they do not move with respect to the EE. Similar to [11] and [16], we do so by including all of the contact forces into the optimal control problem and constraining the solution to be consistent with the dynamic grasp. We do not use online feedback of the object state—given the initial object poses with respect to the EE, the controller generates trajectories to keep those poses constant in an open-loop manner. Assuming the object is in a dynamic grasp, we define the object's orientation as $\boldsymbol{R}_o = \boldsymbol{R}_e$, such that it is aligned with the EE's orientation $\boldsymbol{R}_e$. Furthermore, we have $\boldsymbol{v}_o = \boldsymbol{v}_e + \boldsymbol{\omega}_e \times \boldsymbol{c}$ and $\boldsymbol{\omega}_o = \boldsymbol{\omega}_e$, where $\boldsymbol{v}_e$ and $\boldsymbol{\omega}_e$ are the EE's linear and angular velocity in the body frame and $\boldsymbol{c}$ is the position of the object's CoM with respect to the EE. Thus we can write the object's gravito-inertial wrench as

$$\boldsymbol{w}_{\mathrm{GI}} = -\begin{bmatrix} m(\dot{\boldsymbol{v}}_e - \boldsymbol{R}_e\boldsymbol{g}) + m(\dot{\boldsymbol{\omega}}_e^{\times} + \boldsymbol{\omega}_e^{\times}\boldsymbol{\omega}_e^{\times})\boldsymbol{c} \\ \boldsymbol{J}\dot{\boldsymbol{\omega}}_e + \boldsymbol{\omega}_e^{\times}\boldsymbol{J}\boldsymbol{\omega}_e\end{bmatrix}, \tag{2}$$

where $(\cdot)^{\times}$ converts a vector to a skew-symmetric matrix such that $\boldsymbol{a}^{\times}\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$ for any $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$. Let us define the EE state as the tuple $\boldsymbol{e} = (\boldsymbol{R}_e, \boldsymbol{r}_e, \boldsymbol{\varpi}_e, \dot{\boldsymbol{\varpi}}_e)$, where $\boldsymbol{r}_e$ is the EE position and $\boldsymbol{\varpi}_e = [\boldsymbol{v}_e^T, \boldsymbol{\omega}_e^T]^T$ is the EE's generalized velocity. We can compute $\boldsymbol{e}$ from the robot state $\boldsymbol{x}$ via forward kinematics, in which case we may explicitly write $\boldsymbol{e}(\boldsymbol{x})$.

Now consider an arrangement of objects with N total contact points $\{C_i\}_{i\in\mathcal{I}}$ and corresponding contact forces $\{\boldsymbol{f}_i\}_{i\in\mathcal{I}}$, where $\mathcal{I} = \{1, \ldots, N\}$ (see Fig. 2). By Coulomb's law, each $\boldsymbol{f}_i$ must be inside its friction cone. We use an inner pyramidal approximation of the friction cone

$$\|\boldsymbol{f}_i^t\|_1 \le \mu_i f_i^n, \tag{3}$$

where $f_i^n \triangleq \hat{\boldsymbol{n}}_i^T\boldsymbol{f}_i$ is the force along the contact normal $\hat{\boldsymbol{n}}_i$, $\boldsymbol{f}_i^t$ is the force tangent to $\hat{\boldsymbol{n}}_i$, and μ_i is the friction coefficient. The total contact wrench acting on an individual object is

$$\boldsymbol{w}_{\mathrm{C}} \triangleq \begin{bmatrix}\boldsymbol{f}_{\mathrm{C}}\\ \boldsymbol{\tau}_{\mathrm{C}}\end{bmatrix} = \sum_{j\in\mathcal{J}}\begin{bmatrix}\boldsymbol{f}_j\\ \boldsymbol{r}_j \times \boldsymbol{f}_j\end{bmatrix}, \tag{4}$$

where $\boldsymbol{f}_{\mathrm{C}}$ and $\boldsymbol{\tau}_{\mathrm{C}}$ are the total contact force and torque, $\mathcal{J} \subseteq \mathcal{I}$ is the subset of contact indices for this particular object, and $\boldsymbol{r}_j$ is the location of C_j with respect to the object's CoM. The object is successfully balanced for a given $\boldsymbol{e}$ if a set of contact forces can be found each satisfying (3) and consistent with (1), (2), and (4). Like [11] we assume each contact patch can be represented as a quadrilateral with contact points at the vertices and uniform μ. We also need an extra constraint for each contact point shared between two objects: let $\mathcal{O}^a$ and $\mathcal{O}^b$ be two objects in contact at some point C_i, and denote $\boldsymbol{f}_i^a$ and $\boldsymbol{f}_i^b$ the corresponding contact forces acting on $\mathcal{O}^a$ and $\mathcal{O}^b$, respectively. Then we have the constraint

$$\boldsymbol{f}_i^a = -\boldsymbol{f}_i^b. \tag{5}$$

To lighten the notation going forward, we gather all contact forces into the vector $\boldsymbol{\xi} = [\boldsymbol{f}_1^T, \ldots, \boldsymbol{f}_N^T]^T$, and write $(\boldsymbol{e}, \boldsymbol{\xi}) \in \mathcal{B}$ to indicate that the balancing constraints (1)–(5) are satisfied for all objects.

C. Robust Contact Force Constraints

The constraint (3) ensures all contact forces are inside their respective friction cones, but the friction coefficients may be uncertain or the constraint may be violated by disturbances like vibrations and air resistance. To improve the controller's robustness, it is thus desirable for the tangential contact forces to be small, keeping the forces away from the friction cone boundaries [11]. We propose to plan trajectories using the minimum statically-feasible values of the friction coefficients; that is, the smallest coefficients for which there exists an EE orientation $\boldsymbol{R}_e$ and contact forces $\boldsymbol{\xi}$ satisfying the balancing constraints with zero EE velocity and acceleration, which ensures that the controller can always converge to a stationary position. Here we will focus on the common case when the

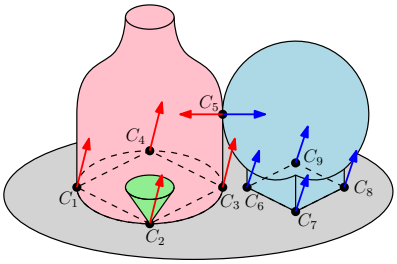

Fig. 2: A bottle (red) and globe (blue) balanced on a tray. This arrangement has a total of $N = 9$ contact points (black dots), with each object having $n = 5$ (C_5 is shared). Contact forces (arrows) at each contact point must belong to their friction cones (one shown in green). The circular contact patch of the bottle is approximated by a quadrilateral. The contact force acting on each object at the shared contact point C_5 must be equal and opposite. If $\mu_i = 0$, the friction cone at C_i collapses to the line along the normal $\hat{\boldsymbol{n}}_i$.

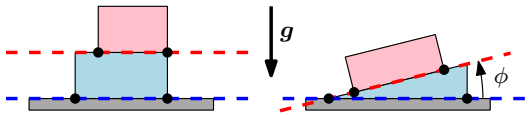

Fig. 3: Planar view of two arrangements of objects, each with two objects balanced on a tray and a total of four contact points (black dots). *Left:* the support planes (dashed lines) of each object are parallel, so the orientation shown is feasible in the presence of gravity with no friction forces (i.e., we can take $\mu_i = 0$ for all $i \in \mathcal{I}$). *Right:* the support planes are *not* parallel, so some friction is *always* required to balance this arrangement.

support planes of each object are parallel to each other (see Fig. 3), so the minimum statically-feasible friction coefficients are simply $\mu_i = 0$ for all $i \in \mathcal{I}$, but in general we can solve an optimization problem to find the minimum values (see [5]).

While choosing the minimum friction coefficients may appear overly conservative, this approach has a number of benefits. First, it removes the need for accurate friction coefficient estimates, which requires time-consuming physical manipulation of the objects to estimate. Second, *mobile* manipulation can produce significant EE vibration, requiring robust motions to ensure objects are balanced. Third, in the common case when $\mu_i = 0$ for all $i \in \mathcal{I}$, the optimal control problem can be simplified as follows. In general we require one contact force variable $\boldsymbol{f}_i \in \mathbb{R}^3$ per contact point, each constrained to satisfy (3). However, when $\mu_i = 0$, we can parameterize the force with a single scalar $f_i \geq 0$ such that $\boldsymbol{f}_i = f_i \hat{\boldsymbol{n}}_i$. This reduces the number of force decision variables by two thirds and replaces (3) with a simple bound, making the optimization problem faster to solve.

D. Constrained Model Predictive Controller

We now formulate a model predictive controller to solve the waiter's problem. The controller optimizes trajectories $\boldsymbol{x}(\tau)$, $\boldsymbol{u}(\tau)$, and $\boldsymbol{\xi}(\tau)$ over a time horizon $\tau \in [t, t+T]$ by solving a nonlinear optimization problem at each control timestep t. Suppressing the time dependencies, the problem is

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\xi}}{\text{argmin}} \quad & \frac{1}{2}\int_{\tau=t}^{t+T} L(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\xi})\, d\tau \\ \text{subject to} \quad & \dot{\boldsymbol{x}} = \boldsymbol{a}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} && \text{(system model)} \\ & (\boldsymbol{e}(\boldsymbol{x}), \boldsymbol{\xi}) \in \mathcal{B} && \text{(balancing)} \\ & \boldsymbol{0} \leq \boldsymbol{d}(\boldsymbol{x}) && \text{(collision)} \\ & \underline{\boldsymbol{x}} \leq \boldsymbol{x} \leq \bar{\boldsymbol{x}} && \text{(state limits)} \\ & \underline{\boldsymbol{u}} \leq \boldsymbol{u} \leq \bar{\boldsymbol{u}} && \text{(input limits)} \end{aligned} \tag{6}$$

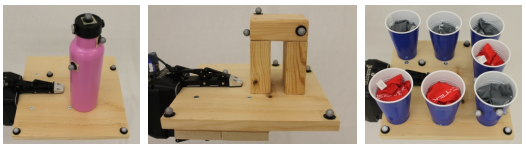

Fig. 4: Bottle, Arch, and Cups object arrangements used for experiments. The arch is an example of non-coplanar contact (the three blocks composing the arch are not attached together). The bottle is filled with sugar and the cups each contain bean bags instead of liquid to avoid spills in the lab.

where the stage cost is

$$L(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\xi}) = \|\Delta\boldsymbol{r}(\boldsymbol{x})\|^2_{\boldsymbol{W}_r} + \|\boldsymbol{x}\|^2_{\boldsymbol{W}_x} + \|\boldsymbol{u}\|^2_{\boldsymbol{W}_u} + \|\boldsymbol{\xi}\|^2_{\boldsymbol{W}_f},$$

with $\|\cdot\|^2_{\boldsymbol{W}} = (\cdot)^T\boldsymbol{W}(\cdot)$ for weight matrix $\boldsymbol{W}$. The EE position error is $\Delta\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{r}_d - \boldsymbol{r}_e(\boldsymbol{x})$. We focus on the case where the desired position $\boldsymbol{r}_d$ is constant, to assess the ability of our controller to rapidly move to a new position without a priori trajectory information. The matrices $\boldsymbol{W}_r$ and $\boldsymbol{W}_x$ are positive semidefinite; $\boldsymbol{W}_u$ and $\boldsymbol{W}_f$ are positive definite. Notice that we do not include a desired orientation: we allow the balancing constraints to handle orientation as needed. If $\mu_i = 0$, then only a scalar f_i is included as a decision variable for each contact force (contained in $\boldsymbol{\xi}$) and (3) is replaced by the constraint $f_i \geq 0$. The vector $\boldsymbol{d}(\boldsymbol{x})$ contains the distances between all pairs of collision spheres representing obstacles and the robot body, which must be non-negative to avoid collisions. When dynamic obstacles are used, then we also augment the state $\boldsymbol{x}$ to predict their motion (see Sec. III-B). We discretize the prediction horizon of (6) with a fixed timestep Δt and solve it online using sequential quadratic programming (SQP) via the open-source framework OCS2 [17].

III. Experiments

We perform experiments on a real 9-DOF velocity-controller mobile manipulator consisting of a Ridgeback mobile base and UR10 arm, depicted in Fig. 1. In all experiments we use $\Delta t = 0.1\,\text{s}$, $T = 2\,\text{s}$, and weights

$$\begin{aligned} \boldsymbol{W}_r &= \boldsymbol{I}_3, & \boldsymbol{W}_x &= \text{diag}(0\boldsymbol{I}_9, 0.1\boldsymbol{I}_9, 0.01\boldsymbol{I}_9), \\ \boldsymbol{W}_u &= 0.001\boldsymbol{I}_9, & \boldsymbol{W}_f &= 0.001\boldsymbol{I}_{\dim(\boldsymbol{\xi})}, \end{aligned}$$

where $\boldsymbol{I}_n$ is the $n \times n$ identity matrix. We use a single SQP iteration per control policy update. Position feedback is provided for the arm by joint encoders and for the base by a Vicon motion capture system, which is used in a Kalman filter to estimate the full robot state. We also use motion capture to track the position of the balanced objects, which is only used for error reporting. The controller is run on a standard laptop with eight Intel Xeon CPUs at 3 GHz and 16 GB of RAM. The balanced objects are shown in Fig. 4.

A. Balancing Constraint Comparison

First, we perform experiments with different combinations of objects and desired EE positions, and compare the trajectories that result from imposing four different sets of balancing constraints:

- **None:** No constraints.
- **Upward:** A constraint to keep the tray oriented upward.

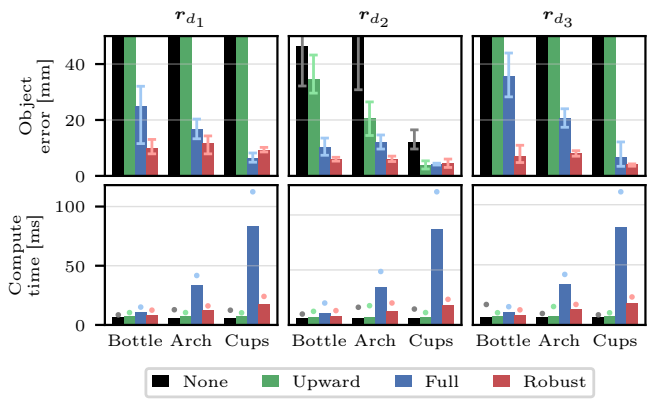

Fig. 5: Object error (top row) and policy compute time (bottom row) for different combinations of objects, goal positions, and constraints in free space. The object error is the maximum distance the object moves from its initial position relative to the tray. In arrangements with multiple objects, only a single one is tracked. The bar shows the average of three runs; the error bars show the minimum and maximum values. The object was completely dropped in all cases where the error extends beyond the axis limits. The compute time is the average time required to compute an updated MPC policy (i.e., one iteration of (6)). The bar shows the average across the three runs (up to the first 6 s of the trajectory); the dot shows the average maximum value across the three runs.

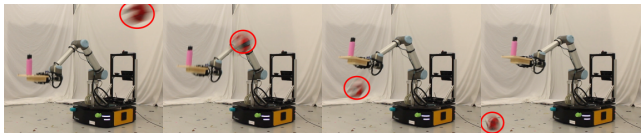

Fig. 6: Example of the robot dodging the volleyball (circled red) while balancing the bottle, with frames spaced by $\Delta = 0.15$ s. Once the ball has passed, the EE moves back to the initial position.

- **Full:** The full set of balancing constraints $(\boldsymbol{e}, \boldsymbol{\xi}) \in \mathcal{B}$ with each μ_i set to 90% of the measured value.[1]
- **Robust:** The full set of constraints $(\boldsymbol{e}, \boldsymbol{\xi}) \in \mathcal{B}$ with $\mu_i = 0$ for all $i \in \mathcal{I}$.

The desired positions (in meters) are $\boldsymbol{r}_{d_1} = [-2, 1, 0]^T$, $\boldsymbol{r}_{d_2} = [2, 0, -0.25]^T$, and $\boldsymbol{r}_{d_3} = [0, 2, 0.25]^T$. The object error and controller compute time in an obstacle-free environment are shown in Fig. 5. As expected, the None and Upward approaches almost always fail—the notable exception is for goal $\boldsymbol{r}_{d_2}$, which requires more base motion and is thus slower than the other trajectories. The Robust constraints typically produce the lowest object error or are close to it. In addition, the Robust constraints scale much better computationally with the number of contacts than the Full constraints, since they require less decision variables and use simpler bounds. The Full constraints also require reasonably accurate friction coefficient estimates; the effectiveness of the Robust constraints show that we need not fear frictional uncertainty and (when statically feasible) can set $\mu_i = 0$ for all $i \in \mathcal{I}$ to reduce compute time.

B. Projectile Avoidance

We now consider an example of an environment that changes over time due to a dynamic obstacle. We use a ball with position $\boldsymbol{r}_b$ and state $\boldsymbol{b} = [\boldsymbol{r}_b^T, \dot{\boldsymbol{r}}_b^T]^T$ modelled as a simple projectile with $\ddot{\boldsymbol{r}}_b = \boldsymbol{g}$. The ball has a known radius but the controller does not know its trajectory a priori. The ball is thrown toward the EE, and the robot must move to avoid the objects being hit while also keeping them balanced. We use the Bottle arrangement and the Robust constraint method. The state $\boldsymbol{b}$ is estimated using the motion capture system and provided to the controller once the ball exceeds the height of 1 m. The state $\boldsymbol{b}$ and the projectile dynamics are added to (6) to predict the ball's motion.

[1]We only use 90% of the measured value to provide some robustness to noise and other disturbances. We subtract a small margin from the support area for the same reason.

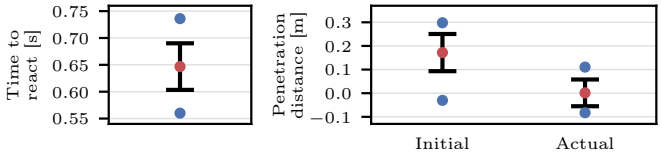

Fig. 7: The projectile avoidance results over 20 trials. In each plot the red dot is the mean, the error bars represent the standard deviation, and the blue dots are the minimum and maximum values. *Left:* The time at which collision would first occur if the robot did not move. In all cases the controller has less than 0.75 s to react. *Right:* Maximum penetration distance between the (virtual) collision spheres around the ball and EE. The "Initial" values represent the maximum penetration distances that would have occurred if the robot had not moved. The "Actual" values are what really happened given that the robot did move.

The results for 20 throws are shown in Fig. 7 and images from one throw are shown in Fig. 6. Throws are split evenly between two directions: toward the front of the EE and toward its side. In all cases, the controller has less than 0.75 s to react and avoid the ball. Out of the 20 trials, there is one in which the ball would not have penetrated the collision sphere even if the robot did not move; and another where the bottle was actually dropped. This failure was not due to a collision, but because the bottle tipped over due to the aggressive motion used to avoid the ball. Also notice that the controller does not always completely pull the robot out of collision: there is a trade-off between balancing the object and avoiding collision. However, since the collision spheres are conservatively large, we did not experience any failures due to collisions. The maximum object error and policy compute time were 32 mm (ignoring the single failure) and 20 ms, respectively, across the 20 trials.

IV. Conclusion

We presented an MPC-based approach for balancing objects with a velocity-controlled mobile manipulator and demonstrated its performance in a variety of real-world experiments. In particular, our method is able to react quickly to moving obstacles. We also proposed using minimal values of μ to add robustness to frictional uncertainty and other force disturbances, and demonstrated that this approach is effective and computationally efficient. Future work will explore the effect of uncertainty in the objects' inertial parameters and the use of object state feedback in the controller.

References

[1] F. G. Flores and A. Kecskeméthy, "Time-optimal path planning for the general waiter motion problem," in *Advances in Mechanisms, Robotics and Design Education and Research*, 2013, pp. 189–203.

[2] K. M. Lynch, "Nonprehensile robotic manipulation: Controllability and planning," Ph.D., Carnegie Mellon University, 1996.

[3] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.

[4] Q.-C. Pham, S. Caron, P. Lertkultanon, and Y. Nakamura, "Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots," *Int. J. of Robotics Research*, vol. 36, no. 1, pp. 44–67, 2017.

[5] A. Heins and A. P. Schoellig, "Keep it upright: Model predictive control for nonprehensile object transportation with obstacle avoidance on a mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 7986–7993, 2023.

[6] B. A. Maxwell *et al.*, "Alfred: The robot waiter who remembers you," in *Proc. AAAI Workshop on Robotics*, 1999, pp. 1–12.

[7] A. Cheong, M. Lau, E. Foo, J. Hedley, and J. W. Bo, "Development of a robotic waiter system," *IFAC-PapersOnLine*, vol. 49, no. 21, pp. 681–686, 2016.

[8] A. Y. S. Wan, Y. D. Soong, E. Foo, W. L. E. Wong, and W. S. M. Lau, "Waiter robots conveying drinks," *Technologies*, vol. 8, no. 3, p. 44, 2020.

[9] 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, 2017.

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

[11] M. Selvaggio, J. Cacace, C. Pacchierotti, F. Ruggiero, and P. R. Giordano, "A shared-control teleoperation architecture for nonprehensile object transportation," *IEEE Trans. on Robotics*, vol. 38, no. 1, pp. 569–583, 2022.

[12] R. Subburaman, M. Selvaggio, and F. Ruggiero, "A non-prehensile object transportation framework with adaptive tilting based on quadratic programming," *IEEE Robotics and Automation Letters*, pp. 1–8, 2023.

[13] V. Morlando, M. Selvaggio, and F. Ruggiero, "Nonprehensile object transportation with a legged manipulator," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2022, pp. 6628–6634.

[14] J. M. Garcia-Haro, "Object oriented control system in humanoid robots for transport tasks," Ph.D., Universidad Carlos III de Madrid, 2019.

[15] C. Zhou, M. Lei, L. Zhao, Z. Wang, and Y. Zheng, "TOPP-MPC-based dual-arm dynamic collaborative manipulation for multi-object nonprehensile transportation," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2022, pp. 999–1005.

[16] M. Selvaggio, A. Garg, F. Ruggiero, G. Oriolo, and B. Siciliano, "Non-prehensile object transportation via model predictive non-sliding manipulation control," *IEEE Trans. Control Systems Technology*, vol. 31, no. 5, pp. 2231–2244, 2023.

[17] "OCS2: An open source library for optimal control of switched systems." [Online]. Available: https://github.com/leggedrobotics/ocs2