Tube Acceleration: Robust Dexterous Throwing Against Release Uncertainty

Yang Liu, *Student Member, IEEE*, and Aude Billard, *Fellow, IEEE*

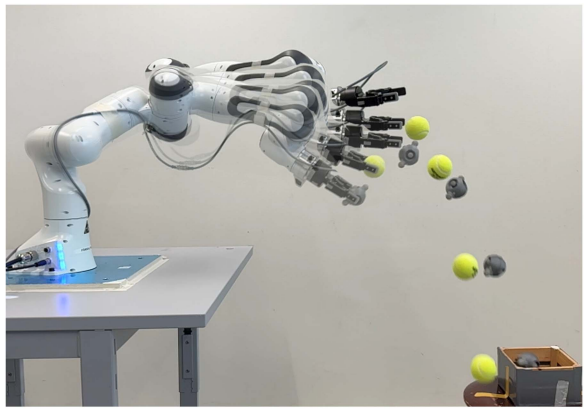
***Abstract*—In robotic throwing, the release phase involves complex dynamic interactions due to object deformation and limited gripper opening speed, often resulting in inaccurate and nonrepeatable throws. While data-driven methods can be employed to compensate for the release uncertainty, the generalizability of learned models to unseen objects is not guaranteed, and object-specific fine-tuning with new data may be required. This fine-tuning process raises concerns about the scalability of such methods for dexterous throwing, where the robot needs to execute diverse motions for throwing various objects. Instead of case-by-case fine-tuning, we aim at designing throwing motion robust against release uncertainty. We encapsulate all uncertainties resulting from complex contact dynamics in a surrogate model of their resulting effect on *gripper opening delay*. We introduce the notion of *tube acceleration* to model the class of constant-acceleration motion in joint space that guarantees a release within the set of valid throwing configurations. We propose a convex relaxation of the primal optimization problem with a tight error bound and evaluate its performance in terms of reliability and efficiency. Results show that the approach offers run-time performance to allow online computation of throws on a 7-DoF robot arm. It achieves a high accuracy and success rate (97% for planar throws) at throwing a variety of complex objects, even when using a simple ballistic model for the object's flying dynamics.**

***Index Terms*—Dexterous manipulation, dynamic manipulation, manipulation planning, robust/adaptive control of robotic systems.**

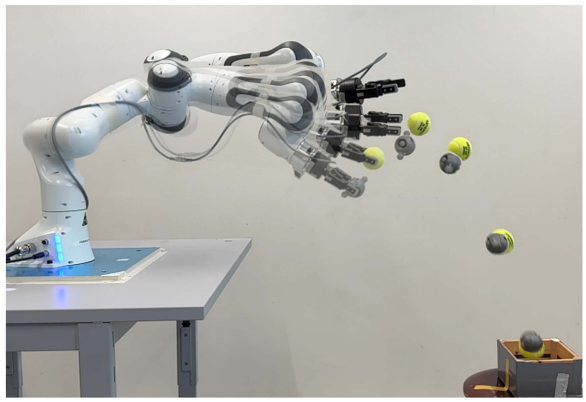
I. Introduction

ROBOTS equipped with throwing capabilities possess remarkable potential to significantly enhance their skills in object manipulation, achieving unprecedented levels of dexterity and efficiency. By carefully transferring the appropriate momentum from the robot to the object during the throwing motion and letting gravity do the rest of the work, robots save themselves from unnecessary movements of their heavy body, enabling a small body motion footprint and lower energy consumption. These two characteristics are particularly desirable for the next generation of collaborative mobile manipulators, where robots will escape from confined cages in conventional industrial setups and travel in cluttered environments, with capacity-limited batteries.

In unstructured environments, the grasp configuration, target position, and obstacles cannot be predetermined. As a result, it becomes essential for the robot to *adaptively* generate throwing motions in response to contextual information that becomes available just prior to the throw. This leads us to advocate



(a)



(b)

Fig. 1. Robot throws a hard plastic ball and a deformable tennis ball. In each figure, the motion commands and gripper opening times are identical when throwing the two objects. (a) Illustration of release uncertainty. The tennis ball escapes the gripper later than the plastic ball along the throwing trajectory due to unmodeled micromechanical deformations, resulting in a smaller horizontal velocity, hence failing to fall into the target box. (b) Thanks to the proposed tube acceleration to compensate for release time uncertainty, the robot traverses through the set of valid throwing configurations. Despite different escape times and, hence, different flying trajectories, the two objects land in identical locations.

Manuscript received 1 January 2024; accepted 27 February 2024. Date of publication 10 April 2024; date of current version 10 May 2024. This work was supported by EU project DARKO under Grant H2020 ICT-46-2020. This paper was recommended for publication by Associate Editor D. Navarro-Alarcon and Editor J. Bohg upon evaluation of the reviewers' comments. *(Corresponding author: Yang Liu.)*

The authors are with Learning Algorithms and Systems Laboratory (LASA), EPFL, 1015 Ecublens, Switzerland (e-mail: yang.liuu@epfl.ch; aude.billard@epfl.ch).

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2024.3386391, provided by the authors.

Digital Object Identifier 10.1109/TRO.2024.3386391

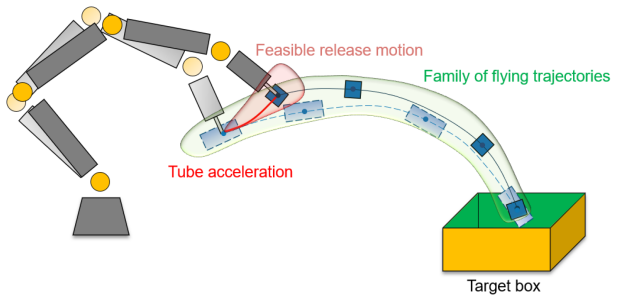


Fig. 2. Schematic for RTV. During the release phase, the robot end-effector traverses a family of valid projectile trajectories encapsulated by tube acceleration. As a result, the landing outcome is agnostic to the exact object release time and, hence, robust to release dynamics.

dexterous throwing, where the robot can apply diverse configurations for throwing. Much like a skilled basketball player who can shoot the ball in various ways and is ready to adjust to unpredictable ball-passing configurations and opponent positions, the robot should have the ability to utilize diverse configurations for throwing and maneuvering adaptively.

The goal of dexterous throwing presents a challenge to the prevailing end-to-end learning approaches for throwing generic objects, compared to the common regime of *planar throwing* in the literature [1], [2], [3], [4], [5], [6], [7], [8], [9]. Generic objects exhibit variations in deformation, restitution, and surface friction, leading to diverse finger–object interactions and resulting in different landing positions under the same robot-throwing motion command, as illustrated in Fig. 1(a). Rather than constructing a microscopic model of the release dynamics with Newton–Euler mechanics, end-to-end learning approaches [8], [9] directly regress the map from perceptual information to motion commands using real-world data, hoping the learned model can implicitly handle all the uncertainties. However, the throwing configuration dependence of release dynamics raises doubts about the scalability of these end-to-end learning approaches, considering the plethora of possible dexterous throwing configurations.

In this regard, we posit that the errors induced on the object's initial velocity at release time, due to intricate gripper-object dynamics, can be described with sufficient fidelity through a surrogate model that accounts solely for the resulting delay in throwing the object, to which we refer as *gripper opening delay*. In this model, the differentiation among various objects lies in their unknown release times, and this uncertainty can be mitigated explicitly through a robustifying procedure, called *recursive task-validity* (RTV).

As illustrated in Fig. 2, the proposed approach requires the robot's end-effector to remain within the tube of valid throwing configurations during the release phase, ensuring that the throwing outcome is independent of the precise object release time. This principle is also observed in neurophysiology: Cohen and Dagmar [10] show that humans learn to exploit the throwing-task redundancy and change their throwing motion to increase the duration of staying in the set of valid throwing configurations, in order to reduce the sensitivity to timing.

In the ideal case, RTV is achieved when the robot's end-effector follows the same path and speed as the object's flying dynamics during the gripper opening. However, due to hardware limitations, robots may not always replicate gravitational acceleration at the end-effector. Consequently, the challenge lies in ensuring RTV while operating within the constraints imposed by the robot's capabilities.

TABLE I
CLASSIFICATION OF PREVIOUS WORKS ON ROBOT THROWING

	Specific object	Generic objects
Planar throwing	[1], [2], [3], [4], [5], [6], [13]	[8], [9]
Dexterous throwing	[14], [15], [16], [17]	This work

Contributions: We formulate the problem of finding throwing motions robust to unmodeled mechanical disturbances [11], [12] at release time as an RTV optimization, meant to guarantee the generation of valid projectile trajectories under robot dynamics' feasibility constraints. The approach is based on the observation that one can travel on a manifold of feasible joint configuration and velocity leading to the same landing position through constant acceleration.

II. Related Works

Over the past two decades, numerous approaches have been offered to enable robots to throw objects [1], [2], [3], [4], [5], [6], [7], [8], [9], [13], [14], [15], [16], [17]. Besides differing in the common dichotomy of model-based methods versus model-free methods, previous approaches to robot throwing differ in their operational design domain (ODD), which we classify along two criteria: 1) the variety of robot configurations and 2) the diversity of thrown objects, see Table I. The two lenses of ODDs are crucial to bringing elegant robot throwing solutions to industrial applications. While previous works considered either throwing various complex objects with simplified planar robot configurations or throwing a single object with a variety of complex configurations, our work aims at enabling throwing of a variety of objects in a diverse set of configurations.

A. Robot Throwing

1) Planar Throwing Versus Dexterous Throwing: Most works in robot throwing typically limit the robot motion to planar throws with low degrees of freedom (≤ 3 joints). The advantage of planar throwing lies in simplifying the problem with a small number of design parameters, enabling planning with comprehensive mathematical models [1] or end-to-end learning with a single scalar output [8].[1] However, as the main attractive features of robot throwing are manipulation efficiency and flexibility, limiting robot throwing motion to be planar loses such features. In comparison, dexterous throwing, which involves more complex and varied robot movements, offers several distinct advantages:

a) Time efficiency: Dexterous throwing allows the robot to adaptively select the most time-efficient throwing configuration from multiple options, based on its current grasp configuration. This adaptability is especially important for achieving time optimality and enhancing the overall throughput of the system.

[1]In [8], the robot's joint position upon throwing is fixed, and throwing to different directions is achieved by changing the robot's first joint, which is perpendicular to the ground.

In comparison, the constraint to planar configurations could reduce these efficiencies.

b) Throwing in cluttered environments: Dexterous throwing also provides the robot with alternative strategies for operating in cluttered spaces. For example, in a scenario where an obstacle obstructs the planar throwing motion, a throw from the side can be executed without colliding with the obstacle, demonstrating greater flexibility in complex environments.

Zhang et al. [14] demonstrate arguably the first work on dexterous throwing with a 6-DoF industrial manipulator using sampling-based motion planning methods. Bombile and Billard [15], [17] offer a framework for dual-arm grabbing and tossing, enabling dynamic manipulation of heavy and bulky objects with a 14-DoF arm pair. Thanks to the fast and reactive motion generation, the system is able to toss objects to a moving target [17]. In our previous work [16], we introduce the concept of *Velocity Hedgehog*, a compact dictionary of robot throwing configurations. It can be queried *online* in milliseconds, providing thousands of qualitatively different throwing configurations for a 7-DoF manipulator. These dexterous throwing configurations allow for fast and adaptive throwing. In these previous works, the dynamic models used for robot throwing assume that the object is released instantaneously. Neglecting to model the gripper-object dynamics at release may lead to inaccurate throws, as shown in Fig. 1(a). The instantaneous object release assumption in [15], [16], and [17] is also invalid for deformable objects, so is the *zero-acceleration* strategy proposed in [14] to handle release uncertainty. In this work, we focus on addressing the gripper–object interaction as our primary challenge.

2) Throw Specific Objects Versus Throw Generic Objects: The majority of previous approaches to throwing were limited to throwing one specific object type ([1]: a wooden block; [2], [3], [4], [6], [13], [18], [19], [20]: a ball; [5]: a square plastic plate; [15], [17]: heavy boxes). In contrast, two recent approaches targeted throwing of a variety of generic objects [8], [9], we review these next.

Zeng et al. [8] provide an end-to-end learning approach for throwing various objects. The throwing configuration is generated from the throwing velocity predicted by a trained model given the target box position and object image. The learned model can handle a large set of objects and multiple target positions in the training set. However, its performance degrades for unseen objects. While one could retrain the model with new data, it is not clear how quickly the robot can learn to throw new objects.

Monastirsky et al. [9] utilize decision transformer [21], a framework that abstracts reinforcement learning (RL) as conditional sequence modeling problems, to generate robot throwing motion by conditioning the learned autoregressive model on the desired landing position, past states (joint position history), and past actions (joint velocity history). Although the direct deployment of the model trained purely in simulation results in very inaccurate throws, the model can throw accurately after fine-tuning with a handful of real throwing experiments. By applying domain randomization[2] on robot control error and gripper opening delay during training, the final policy generalizes well to unseen and even deformable objects.

Both approaches take a learning approach to modeling the complexity of the throwing behavior. Our approach differs from the earlier reviewed two contributions in the following key aspects.

1) While the works in [8] and [9] are designed for planar throwing, our framework can handle full throwing configurations as input, hence enlarging the range and complexity of throwing types.
2) While the author in[9] treats errors due to gripper-object dynamics as a domain randomization issue, we explicitly control for these uncertainties through the tube acceleration and offer theoretical justification for the approach.
3) We explicitly ensure that the generated throws are dynamically feasible.

We show that we achieve comparable throwing accuracies to those documented in [8] and [9]. Our approach relies on convex optimization and, hence, offers a computationally cheaper alternative to learning methods that require a large number of training examples.

B. Robust Throwing Against Uncertainties

Pekarovskiy and Buss [5] conduct a comprehensive analysis of the set of valid throwing configurations (goal manifold), for planar nonprehensile throwing. They formulate an optimal control problem with the goal manifold as boundary conditions and antislipping constraints to ensure dynamic grasping during the throwing motion. In the presence of uncertainty due to contact friction, the ratio between contact normal force and tangential force is maximized to maintain antislipping constraints as effectively as possible.

Following a similar planar approach to throwing as in [5], the authors in [18], [19], and [20] perform sensitivity analysis of the landing position w.r.t. the robot's throwing configuration, initial configuration, and joint friction. The sensitivity is represented as the gradient of landing position w.r.t. parameters of interest. To design a robust throwing configuration, they enumerate discretized valid configurations and select the one with the minimum norm (or weighted norm, considering error covariance [20]) of the sensitivity gradient.

The analysis of the goal manifold in [5] suggests the existence of a "Nullspace" of throwing configurations that result in the same object landing position. However, this property has not yet been leveraged for robust motion design. In this work, we exploit the slackness of the goal manifold as a buffer for uncertainty, accounting for our lack of knowledge about release dynamics.

C. Strategies to Handle Release Uncertainty

Handling release uncertainty has been a challenging aspect of robot throwing, and different strategies have been explored in the literature. Some works ignore this uncertainty while others attempt to minimize hand–object interaction. Senoo et al. [2] develop a specialized high-speed gripper that opens instantly, leading to quick and accurate object release during throwing. In the context of nonprehensile throwing, Lynch and Mason [1] design the arm motion to be maximally decelerated at the

[2]Adding noise during training in simulation, which encourages the policy to be robust.

release instant, resulting in nearly instantaneous object release. However, as noted in [22], if the object is firmly grasped before the release phase to avoid unexpected slips, the gripper will gradually loosen its grip around the object during the release phase, causing it to enter free-flight after a certain amount of time. While fixed release timing is adequate for accurate throws of rigid objects, as demonstrated in TidyBot [23], the throwing of generic deformable objects necessitates a more careful design of the robot's release motion. This is particularly important given the diverse characteristics of deformable objects, such as variations in contact geometry, surface material, and degree of deformability. Zhang et al. [14] recognize the potential issue of release uncertainty during the gripper opening delay and compensate for this delay by designing the robot motion with constant velocity (i.e., *zero acceleration*) during the gripper opening. However, as we will demonstrate in our experiments, this strategy is insufficient, as the landing positions of the initial flying states on a zero-acceleration trajectory are not invariant to the release time.

Compared to previous heuristic approaches for handling release uncertainty, our work addresses this problem systematically. Our framework is similar in spirit to the concept of soft-catching proposed by Salehian et al. [24], where a dynamical system (DS)-based motion generator is used to *follow* the flying object and reduce collisions between the object and the finger. This soft-catching strategy significantly increases the catching success rate compared to hard-catching without object following. In this work, we use convex programming to generate dynamically feasible *tube acceleration* to drive the robot end-effector traversing the beam of virtual flying objects.

III. Preliminaries

A. Release Dynamics in Robot Throwing

A simplified description of release dynamics, without considering hand-object collision, involves the following process (see Fig. 3).

1) The gripper firmly grasps the object $\mathcal{O}$ to prevent unexpected slips before the release, causing object deformation due to the applied force.
2) The robot executes the throwing motion $q(\cdot)$ under the motor torque command sequence $\tau(\cdot)$, which spans the entire throwing process, including the after-release phase.
3) At a specific instant during the throwing motion, the gripper receives the opening command $u(\cdot)$ and gradually begins to open.
4) In reaction, the previously squeezed object gradually expands back to its original shape, leading to *changing* normal force F_n, *changing* tangential friction limit $F_f^{\max}$, and *changing* torsional friction limit $\tau_f^{\max}$.
5) As the gripper continues to open, it reaches a point where it can no longer maintain static equilibrium between the object and the gripper. At this stage, the object starts to slide within the gripper, experiencing the effects of normal force, friction force, and gravity.

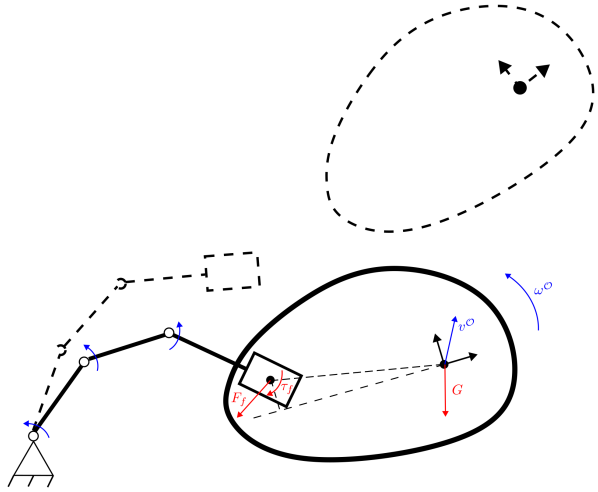


Fig. 3. Snapshot of the release instant with the robot on the left and the thrown object on the right. Black arrows consist of the reference frame of the object $\mathcal{O}$, located at its CoM and along its principle axis. Blue arrows denote motion vectors and red arrows denote forces applied on $\mathcal{O}$. Note that the directions of friction forces F_f, τ_f indicate the relative motion of the gripper and the object. The object is exaggerated to visualize the complex interaction during object release.

TABLE II
Notations for Geometric Modeling in Fig. 4

A	Robot base
B	Target box
E	Robot end-effector
E'	Projection of E on $X-Y$ plane
EBE'	Throwing plane, with origin at B
r	Object's horizontal coordinate in the throwing plane
z	Object's vertical coordinate in the throwing plane
$\dot{r}$	Object's horizontal velocity
$\dot{z}$	Object's vertical velocity

6) Finally, the object fully detaches from the gripper and enters a free-flying state, moving toward the desired target position.

This process is notoriously difficult to model, and its estimation procedure would require complex and expensive hardware setups, including tactile sensors and high-precision motion capture systems. In industrial setups, the diversity of object classes and the variability of object instances within the same class also hinder the availability of a detailed microscopic description of release dynamics for each throwing task. In this regard, we hypothesize that a kinematic model of *gripper opening delay* could be a simple surrogate model to capture the macroscopic effect on the momentum exchange between the robot and the object during the release, which can be utilized to design motion for robust throwing.

B. Geometric Modeling of Robot Throwing

The geometric modeling of the robot throwing problem is adopted from our previous work in [16] and is illustrated in Fig. 4, with notations introduced in Table II. We define the object's horizontal velocity direction as the positive horizontal direction in the throwing plane.

In this model, we assume that the object is grasped at the center of mass (CoM); hence, if the object is perfectly released

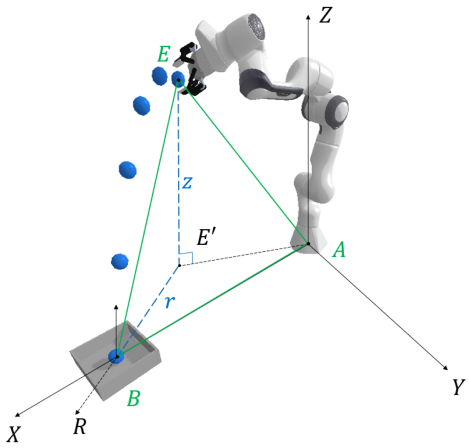


Fig. 4. Geometry of 3-D throwing.

and commences free-flying at a given valid nominal throwing state, it will land at point B. However, if the release dynamics delays the object entering free-flying, the landing point might not overlap with the target B.

C. Backward Reachable Tube

In the throwing plane, the object flying state is denoted as $\xi = [r, z, \dot{r}, \dot{z}]^\top \in \mathbb{R}^4$. The flying dynamics is described by a first-order differential equation $\dot{\xi} = f_{\text{fly}}(\xi)$. The flying trajectory of f_{fly} starting from state ξ^0 is denoted as $\zeta_{f_{\text{fly}},\xi^0}(t) : [0, +\infty] \to \mathbb{R}^4$. We assume that a user has provided the robot with a landing target set $\mathcal{X} \subset \mathbb{R}^4$, which describes the allowed landing position slack and the range of allowed landing velocities.

For a flying trajectory that enters the landing target set, any state on this trajectory segment is a valid throwing configuration. Therefore, by aggregating all the trajectories that eventually enter the landing target set, we obtain the set of valid throwing configurations, which we call the backward reachable tube (BRT). Mathematically, the BRT is defined as

$$\mathcal{G}\left(f_{\text{fly}}, \mathcal{X}\right) = \left\{\xi^0 \,|\, \exists\, t \geq 0,\, \zeta_{f_{\text{fly}},\xi^0}(t) \in \mathcal{X}\right\}.$$

Given a connected target set $\mathcal{X} \subset \mathbb{R}^4$, the BRT $\mathcal{G}$ associated with a smooth continuous flying dynamics f_{fly} is also a connected set in $\mathbb{R}^4$ without any isolated regions (or 'holes') (see [25], Th. 3.5). As a result, BRT $\mathcal{G}$ is defined in a topological space, with well-defined topological concepts such as boundaries and interiors. Hence, the BRT can be represented as a level-set function $f_{\text{BRT}}(\xi^0) : \mathbb{R}^4 \to \mathbb{R}$, with the following interpretations.

1) $f_{\text{BRT}}(\xi^0) > 0 \Leftrightarrow \xi^0 \notin \mathcal{G}$, indicating that the initial flying state ξ^0 is not a valid throwing configuration.
2) $f_{\text{BRT}}(\xi^0) < 0 \Leftrightarrow \xi^0 \in \text{Int}\mathcal{G}$, implying that the initial flying state ξ^0 is a valid throwing configuration.
3) $f_{\text{BRT}}(\xi^0) = 0 \Leftrightarrow \xi^0 \in \partial\mathcal{G}$, indicating that the initial flying state ξ^0 lies on the boundary of the BRT.

However, obtaining digital representations of BRT is generally computationally expensive [26], and the choice of representation depends on the specific downstream tasks. In our previous work on nominal throwing configuration planning [16], BRT is represented as a 4-D point cloud of valid throwing configurations. In the next section, we will introduce a neural implicit representation of BRT, which is particularly suitable for the robust throwing planning problem.

D. Flowmap and Neural ODE

We define flowmap Φ as the mapping from the *initial condition* ξ^0 to a *scalar outcome* driven by the flying dynamics. The *outcome of interest* could be the function of the state at a given time or the state upon a certain *event* happening. Using the *adjoint sensitivity method* (see [27]), neural ODE [28] is able to efficiently compute $\nabla_{\xi^0}\Phi \in \mathbb{R}^d$, which is the gradient of the scalar function Φ w.r.t. initial condition ξ^0. The method scales linearly with problem size, has low memory cost, and explicitly controls numerical errors.

In the context of robotic throwing, we are interested in the object's flying flowmap that maps from the release state $\xi^0 = (r^0, z^0, \dot{r}^0, \dot{z}^0)$ to the horizontal landing position in the throwing plane EBE', denoted as r_{land}. However, in this scenario, the landing time is implicitly defined by the release state, flying dynamics, and landing height, making it difficult to determine the termination criterion explicitly. To resolve this difficulty, neural event ODE [29] models the event as a scalar function $h(\xi)$ of the state ξ, which is equal to zero if and only if the event happens. The event function is integrated together with neural ODE and differentiated through. In our setting, the landing event function can be defined as

$$h(r, z, \dot{r}, \dot{z}) = z + \max(\dot{z}, 0).$$

The condition on $\dot{z}$ ensures that the vertical velocity is negative upon landing. Therefore, if the object's initial position is lower than the landing height and the initial vertical velocity is positive, the solver will continue integration when passing the landing height during the upward flight and will terminate integration only when flying downward.

As a result, we obtain the following *flying flowmap* of the object's flying dynamics:

$$r_{\text{land}} = \Phi_{\text{fly}}\left(r^0, z^0, \dot{r}^0, \dot{z}^0\right). \qquad (\textit{flying flowmap})$$

Since we define the origin of the throwing plane EB at target B, the *outcome of interest* r_{land} should be as close to zero as possible.

It is worth noting that although the analytical expression of the flying flowmap is available for projectile motion—where the object is solely influenced by gravitational force, such analytical solutions are generally not available for nonlinear flying dynamics. This holds true even for object flying dynamics with a simple quadratic air drag model, as discussed in the review by Lubarda and Lubarda [30]. On the other hand, the neural event ODE is an efficient numerical method capable of handling arbitrary flying dynamics models.

IV. Problem Description

As shown in Fig 2, for a n-DoF manipulator given a nominal throwing configuration $(q_0, \dot{q}_0) \in \mathbb{R}^{2n}$, where the corresponding

TABLE III
EXPLANATION OF CONSTRAINTS IN PROBLEM RTV

(1b)–(1c)	double integrator constraints
(1d)	differential forward kinematics with Jacobian function J
(1e)	throwing velocity direction aligned with ${}^{E}p^{B}_{xy}$
(1f)–(1i)	relate Cartesian variables with throwing plane variables
(1j)	correct landing position in the throwing plane EB
(1k) -(1m)	robot hardware limits

end-effector's state is inside the BRT of the target box position $p \in \mathbb{R}^3$, and known object flying dynamics $f_{\text{fly}} : \mathbb{R}^4 \to \mathbb{R}^4$, the goal is to find a motion sequence $q(\cdot) : [0, T] \to \mathbb{R}^n$ such that the following holds.

1) End-effector's state remains inside the BRT for a time window $[0, T]$.
2) Motion sequence $q(\cdot)$ is dynamically feasible.

Then, in the "gripper opening delay" model, the robot can perform a valid throw regardless of the exact release time within the release phase.

1) Spatial Algebra Notations: All spatial vectors are expressed in the robot base frame. ${}^{A}p^{B} \in \mathbb{R}^3$ denotes the vector from point A to point B. $v \in \mathbb{R}^3$ denotes the Cartesian velocity of the robot end-effector. Subscripts of spatial vectors represent their elements or collections of elements, e.g., ${}^{A}p^{B}_{z}$ denotes the vertical component of ${}^{A}p^{B}$, ${}^{A}p^{B}_{xy} = [{}^{A}p^{B}_{x}, {}^{A}p^{B}_{y}]$ denotes the collection of the horizontal components of ${}^{A}p^{B}$.

The RTV problem can be formulated as follows:

Problem RTV

$$\text{Find: } \{q(\cdot), \dot{q}(\cdot), \ddot{q}(\cdot)\} \tag{1a}$$

$$\text{subject to: } q(t) = \int_0^t \dot{q}(\tau)d\tau + q_0 \quad \forall t \in [0, T] \tag{1b}$$

$$\dot{q}(t) = \int_0^t \ddot{q}(\tau)d\tau + \dot{q}_0 \quad \forall t \in [0, T] \tag{1c}$$

$$v(q, \dot{q}) = J(q)\dot{q} \tag{1d}$$

$$[v_x, v_y]^{\top}[-{}^{E}p^{B}_{y}(q),\ {}^{E}p^{B}_{x}(q)] = 0 \tag{1e}$$

$$r(q) = -\left\|{}^{E}p^{B}_{xy}(q)\right\|_2 \tag{1f}$$

$$z(q) = -{}^{E}p^{B}_{z}(q) \tag{1g}$$

$$\dot{r}(q, \dot{q}) = \|v_{xy}\|_2 \tag{1h}$$

$$\dot{z}(q, \dot{q}) = v_z \tag{1i}$$

$$\Phi_{\text{fly}}(r(q), z(q), \dot{r}(q, \dot{q}), \dot{z}(q, \dot{q})) = 0 \tag{1j}$$

$$q_{\min} \le q(t) \le q_{\max} \quad \forall t \in [0, T] \tag{1k}$$

$$\dot{q}_{\min} \le \dot{q}(t) \le \dot{q}_{\max} \quad \forall t \in [0, T] \tag{1l}$$

$$\ddot{q}_{\min} \le \ddot{q}(t) \le \ddot{q}_{\max} \quad \forall t \in [0, T]. \tag{1m}$$

Note that constraints (1d)–(1j) also have to be satisfied throughout the release phase. Additionally, the problem formulation could optimize certain metrics, e.g., maximize window duration, maximize margin to the constraint boundary [31], etc.

V. APPROACH

Problem RTV is difficult to solve due to the functional decision variables and the nonconvex constraints. In order to let the robust throwing motion generator be ready to handle large amounts of throwing configurations in dexterous throwing, we choose to convexify Problem RTV and derive the error bound introduced by the convex approximation. This approach allows us to efficiently and reliably find a motion sequence that satisfies the robustness constraints and provides a lower bound on the suboptimality of the solution compared to the true nonconvex problem.

A. Quasi-Static Approximation With Tube Acceleration

Our approach to convexifying the problem is as follows.

1) *Constant tube acceleration:* We restrict the decision space of the release motion to constant acceleration. This choice is driven by two design considerations.
a) Due to higher-order effects on robot dynamics, constant acceleration is easier to track compared to changing accelerations.
b) The integrator constraint (1c) is linearized under constant acceleration.
2) *Quasi-static approximation:* We discretize the double integrator dynamics (1b)–(1c) with one time step of size T, the duration of the tube acceleration. Additionally, we ignore the effect of tube acceleration on joint positions. As a result, the following statements hold.
a) Positional states, including q, r, z, become *parameters* of the problem rather than decision variables.
b) The throwing direction constraint (1e) gets linearized.
c) Robot acceleration limits (1m) can be represented as convex polytopes.

As a result, we obtain the following formulation Problem Tube-CVX for the recursive validity problem, blue variables are induced by constant tube acceleration $\ddot{q}_{\text{tube}}$ while black variables can be viewed as *parameters* of the program and, hence, are treated as fixed in the solver:

Problem Tube-CVX

$$\text{Find: } \{\ddot{q}_{\text{tube}}\} \tag{2a}$$

$$\text{subject to: } q_T = q_0 + T\dot{q}_0 \tag{2b}$$

$$\dot{q}_T = \dot{q}_0 + T\ddot{q}_{\text{tube}} \tag{2c}$$

$$r_T = -\left\|{}^{E}p^{B}_{xy}(q_T)\right\|_2 \tag{2d}$$

$$z_T = -{}^{E}p^{B}_{z}(q_T) \tag{2e}$$

$$v_T(q_T, \dot{q}_T) = J(q_T)\dot{q}_T \tag{2f}$$

$$[v_{T,x}, v_{T,y}]^{\top}\left[-{}^{E}p^{B}_{y}(q_T),\ {}^{E}p^{B}_{x}(q_T)\right] = 0 \tag{2g}$$

$$\dot{r}_T = \|v_{T,xy}\|_2 \tag{2h}$$

$$\dot{z}_T = v_{T,z} \tag{2i}$$

$$\Phi_{\text{fly}}\left(r_T, z_T, \dot{r}_T, \dot{z}_T\right) = 0 \tag{2j}$$

$$\dot{q}_{\min} \le \dot{q}_T \le \dot{q}_{\max} \tag{2k}$$

$$\ddot{q}_{\min} \le \ddot{q}_{\text{tube}} \le \ddot{q}_{\max}. \tag{2l}$$

In the previous formulation, all the equality constraints except for the BRT constraint (2j) are linear, and all the inequality constraints are polytopic, resulting in a convex problem [32].

Linearizing the BRT constraint (2j) is a key ingredient of our method and will be explained in detail in Section V-C. With this approximation, the torque limits on the robot can be formulated as the following polytopic constraint:

$$\tau_{\min} \leq \tau = M(q_0)\ddot{q}_{\text{tube}} + C\left(q_0, \dot{q}_0\right)\dot{q}_0 + G\left(q_0\right) \leq \tau_{\max} \tag{2m}$$

where $\tau \in \mathbb{R}^n$ denotes the motor torque, $M \in \mathbb{R}^{n\times n}$ denotes the mass matrix, $C(q,\dot{q})\dot{q} \in \mathbb{R}^n$ denotes the Coriolis force, and $G(q) \in \mathbb{R}^n$ denotes the gravity force.

Discussion on the previous approximation scheme is as follows.

1) *Potential error from ignored $\ddot{q}_{\text{tube}}$ in the integrator constraint (2c):* With $T = 100$ ms and typical $\ddot{q}_{\max} = 10$ rad/s^2, the corresponding maximum joint position error will be around $\frac{1}{2}T^2\ddot{q}_{\max} = 0.05$ rad $= 2.86°$. For a robot with a 1-m reach, this corresponds to a 5-cm error at the end-effector.
2) *Treatment of constraints (1d)–(1m):* Only the constraints at the end of the release motion (at T) are imposed while, in principle, they have to be satisfied all the time during the gripper opening window. However, the following statements hold.

a) For the throwing plane constraint (1e), we show in Section V-B that the error in end-effector velocity direction is of second-order in T, which is assumed to be small.
b) For the *flying flowmap* constraint (1j), we show in Section V-C that due to the flatness of the object *flying flowmap*, the constraint violation of (1j) is small.
c) For robot limit constraints (1k)–(1m), they can be handled by the safety margin of real hardware limits.

B. On Terminal Time-Only Throwing Plane Constraint

The imposed throwing direction constraint (2g) at time T corrects the throwing direction error induced by the second-order curvature of the robot forward kinematics at nominal throwing configuration, and hence, the ignored throwing direction constraint for $t \in (0, T)$ is of second-order of T, which is assumed to be small. To show this, we write the end-effector velocity driven by the tube acceleration during the release window as follows:

$$v\left(q_t, \dot{q}_t\right) = J(q_t)\dot{q}_t \tag{3}$$

$$= \left(J(q_0) + (q_t - q_0)^\top H(q_0) + O\left(\|q_t - q_0\|^2\right)\right)\dot{q}_t \tag{4}$$

$$= \left(J(q_0) + \left(q_0 + t\dot{q}_0 + \frac{1}{2}t^2\ddot{q}_{\text{tube}} - q_0\right)^\top H(q_0)\right)$$
$$\left(\dot{q}_0 + t\ddot{q}_{\text{tube}}\right) + O(t^2) \tag{5}$$

$$= \left(J(q_0) + t\dot{q}_0^\top H(q_0)\right)\left(\dot{q}_0 + t\ddot{q}_{\text{tube}}\right) + O(t^2) \tag{6}$$

$$\approx J(q_0)\dot{q}_0 + t\left(J(q_0)\ddot{q}_{\text{tube}} + \dot{q}_0^\top H(q_0)\dot{q}_0\right) \tag{7}$$

where $H \in \mathbb{R}^{3\times n\times n}$ is the Hessian of the robot forward kinematics. Now we see that the end-effector velocity is approximately affine in time t. Therefore, by only imposing the direction of the velocity at the start $t = 0$ (valid nominal throwing configuration) and at the end $t = T$ (constraint (2g)), the throwing direction is approximately constant throughout the release motion with $T \ll 1$. This property is also shown in Fig. 5.

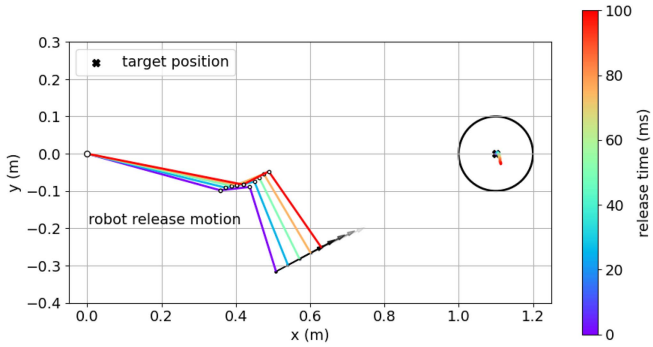


Fig. 5. Robot motion with throwing plane constraint imposed only at terminal time T approximately keeps the correct velocity direction throughout the release window.

C. Flowmap Constraint via Neural Implicit Reachability

The flowmap constraint (1j) ensures that in the throwing plane, the object can land at the target position. The given nominal throwing configuration $(q_0, \dot{q}_0)$ is assumed to satisfy this constraint, and hence, we have

$$\Phi_{\text{fly}}(r(q_0), z(q_0), \dot{r}(q_0, \dot{q}_0), \dot{z}(q_0, \dot{q}_0)) = 0. \tag{8}$$

Similarly, we can write the BRT constraint for the throwing state at the end of the gripper opening window induced by tube acceleration $\ddot{q}_{\text{tube}}$ as the following flowmap constraint:

$$\Phi_{\text{fly}}(r_T, z_T, \dot{r}(\ddot{q}_{\text{tube}}), \dot{z}(\ddot{q}_{\text{tube}})) = 0 \tag{9}$$

note that with quasi-approximation, r_T and z_T are fixed parameters in Problem Tube-CVX, and hence, here we drop the *flying flowmap*'s dependency on q. In the "gripper opening delay" model, the initial free-flying state of the object is assumed to be identical to the state of the robot end-effector at an unknown time during the release phase. In general, with zero tube acceleration $\ddot{q}_{\text{zero}}$ (constant joint velocity) during the release phase after bypassing the nominal throwing configuration, the end-effector state will not lead the object to land in the desired box position, i.e.,

$$\Phi_{\text{fly}}(r_t, z_t, \dot{r}_t(\ddot{q}_{\text{zero}}), \dot{z}_t(\ddot{q}_{\text{zero}})) \neq 0 \quad \forall t \in (0, T]. \tag{10}$$

However, if one follows the local gradient around the isoline of the *flying flowmap* at the *end* of the release window T with *nominal throwing velocity* as shown in Fig. 6, one can find a tube acceleration that corrects the release velocity at T to ensure that the object lands on the target *if the "gripper opening delay" equals exactly* T. Then, during the release phase driven by the tube acceleration, at each infinitesimal timestep, the change of *landing position* due to the change *end-effector position* is compensated correctly by the change of *end-effector velocity*, thanks to the flatness of the *flying flowmap*. In this way, RTV is achieved.

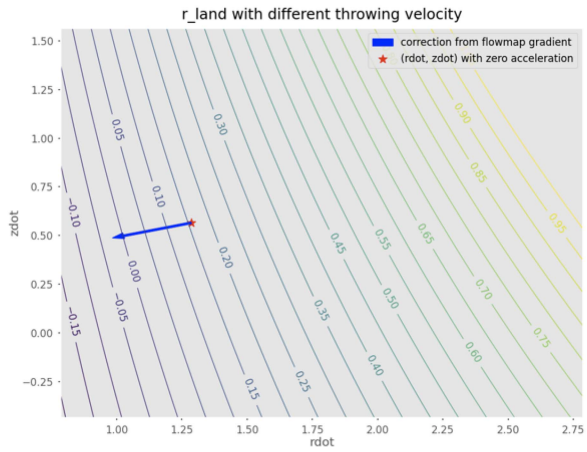


Fig. 6. Illustration of object *flying flowmap* of projectile dynamics and retraction constraint.

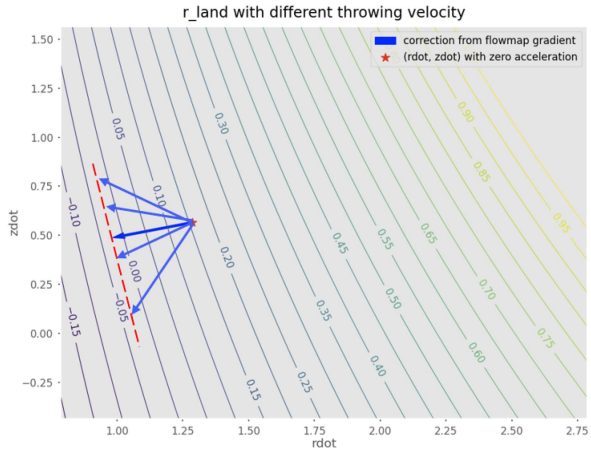


Fig. 7. Retraction constraint is a 1-D subspace constraint, any solution on the red dashed line satisfies the retraction constraint.

Mathematically, let

$$\Phi_{\text{fly}}(0) := \Phi_{\text{fly}}\left(r_T, z_T, \dot{r}_T(\ddot{q}_{\text{zero}}), \dot{z}_T(\ddot{q}_{\text{zero}})\right) \tag{11}$$

which is the landing position if the object is released at time T with robot motion driven by zero acceleration. We define a *retraction constraint* as

$$-\Phi_{\text{fly}}(0) = \left[\frac{\partial \Phi_{\text{fly}}}{\partial \dot{r}_T(\ddot{q}_{\text{zero}})}, \frac{\partial \Phi_{\text{fly}}}{\partial \dot{z}_T(\ddot{q}_{\text{zero}})}\right]^\top \begin{bmatrix} \dot{r}_T(\ddot{q}_{\text{tube}}) - \dot{r}_T(\ddot{q}_{\text{zero}}) \\ \dot{z}_T(\ddot{q}_{\text{tube}}) - \dot{z}_T(\ddot{q}_{\text{zero}}) \end{bmatrix} \quad (\textit{retraction constraint})$$

this replaces the nonlinear inequality constraint (2k) with a linear equality constraint in Problem CVX. Next, we study the error induced on the true solution of the optimization through the linearization of this constraint.

VI. Error Analysis of Retraction Constraint

A. Intuition: One-Step Newton's Method for Root-Finding

The idea behind the retraction constraint and its error analysis resembles applying Newton's method for one-step for root finding. For a generic 1-D nonlinear equation $y = f(x)$ with root x^*, we can find an approximate solution $\hat{x}$ from an initial guess x_0 using the following formula:

$$\hat{x} = x_0 - \nabla f^{-1}(x_0)\, f(x_0). \tag{12}$$

In particular, the following statements hold.

1) If $f(x)$ is linear $\Rightarrow$ one-step Newton is exact.
2) If $f(x)$ has small curvature and $|\hat{x} - x_0|$ is small $\Rightarrow$ $|f(\hat{x})|$ is small.

The technical difficulty in analyzing the retraction constraint lies in the following two aspects.

1) The object *flying flowmap* is a multivariate function, which makes the analysis more complex compared to the above 1-D analysis.
2) The values of $(\dot{r}_T, \dot{z}_T)$ obtained from the convex program do not admit a closed-form solution. As shown in Fig. 7, the exact solution on the *retraction constraint* subspace selected by the convex program is unknown *a priori*.

B. Flatness of Object Flying Flowmap

From Fig. 7, we observe that the object flying flowmap of projectile dynamics is "quasi-flat." To capture this flatness and relate it to the approximation error of the *retraction constraint*, we compute the spectral radius of the Hessian of the object flying flowmap using finite difference.

For a multivariate function $f(x) : \mathbb{R}^n \to \mathbb{R}$, the spectral radius of its Hessian matrix is the maximum absolute value of the eigenvalues of the Hessian $H(x) \in \mathbb{R}^{n\times n}$, where $H_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$. In our case, we consider the error function induced by the linearization of the BRT constraint

$$e\left(\dot{r}_T, \dot{z}_T\right) = \Phi_{\text{fly}}\left(r_T, z_T, \dot{r}_T, \dot{z}_T\right). \tag{13}$$

We analyze the spectral decomposition of the Hessian for two planar flying dynamics: 1) gravity-only flying dynamics and 2) flying dynamics with quadratic Newton air drag. The gravity-only flying dynamics are described by

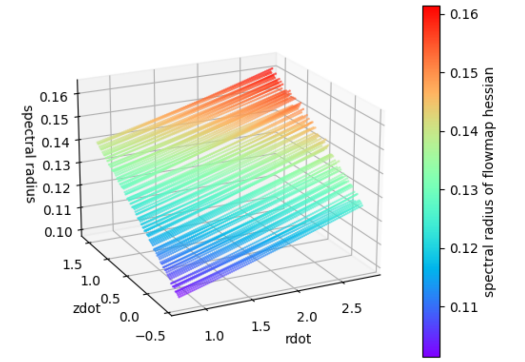
$$\dot{\xi} = \frac{d}{dt}\begin{bmatrix} r \\ z \\ \dot{r} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \dot{r} \\ \dot{z} \\ 0 \\ -g \end{bmatrix} \tag{14}$$

where g is the gravitational constant. The flying dynamics with quadratic Newton air drag are described by

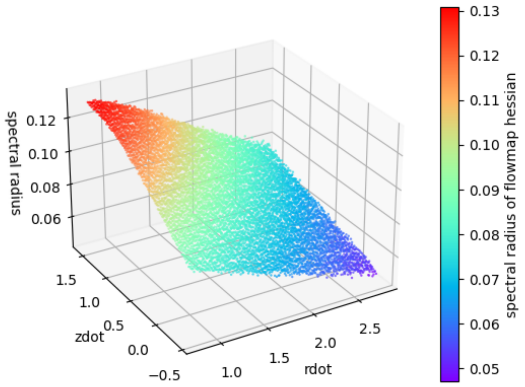
$$\dot{\xi} = \frac{d}{dt}\begin{bmatrix} r \\ z \\ \dot{r} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \dot{r} \\ \dot{z} \\ -\mu \dot{r}\sqrt{\dot{r}^2 + \dot{z}^2} \\ -\mu \dot{z}\sqrt{\dot{r}^2 + \dot{z}^2} - g \end{bmatrix} \tag{15}$$

where $\mu = Am\rho$, with A being the cross-sectional area in the motion direction, m being the object weight, and ρ being the air density. In this analysis, we set $A = 0.2\pi, m = 0.5$, and $\rho = 1.29$.

Fig. 8 shows the spectral radius of the flowmap Hessian at different throwing velocities for the two flying dynamics. Over a



(a)



(b)

Fig. 8. Spectral radius of the object flying flowmap with different flying dynamics. (a) Gravity-only flying dynamics. (b) Flying dynamics with Newton airdrag.

large region of throwing velocities, the maximum spectral radius of the gravity-only dynamics is approximately 0.16, and for the flying dynamics with Newton air drag, it is approximately 0.13. We will use these values to compute the parametric error bound of the *retraction constraint*.

C. Error Bound

Lemma 1: For a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$, if the spectral radius of the Hessian is bounded by L, then the gradient of f is L-Lipschitz continuous.

Proof: Let $x, y \in \mathbb{R}^n$ be any two points. Using the mean value theorem for vector-valued functions, we have

$$|\nabla f(x) - \nabla f(y)| = \left| \int_0^1 \nabla^2 f(y + t(x-y))(x-y)dt \right| \tag{16}$$

$$\leq \int_0^1 |\nabla^2 f(y + t(x-y))| \cdot |x-y| dt. \tag{17}$$

Since the spectral radius of the Hessian is bounded by L, we have $|\nabla^2 f(y + t(x-y))| \leq L$ for all $t \in [0,1]$. Therefore

$$|\nabla f(x) - \nabla f(y)| \leq L \int_0^1 |x-y| dt = L|x-y|. \tag{18}$$

Thus, we have shown that $|\nabla f(x) - \nabla f(y)| \leq L|x-y|$, which means the gradient of f is L-Lipschitz continuous. □

Lemma 2: For a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ with L-Lipschitz continuous gradient, the following inequality holds for all $x, y \in \mathbb{R}^n$:

$$\left| f(y) - f(x) - \nabla f(x)^{\mathrm{T}}(y-x) \right| \leq \frac{L}{2}\|y-x\|^2 \tag{19}$$

where L is the Lipschitz constant of the gradient.

In general, the norms on $\mathbb{R}^n$ in Lemma 2 can be arbitrary, and the Lipschitz constant L will depend on the choice of the norm. This is a classical result in convex analysis, and its proof can be found in [33], Th. 2.1.5.

Now, we present our main result.

Theorem 1: For a robot with maximum Cartesian acceleration $\ddot{p}_{\max} \in \mathbb{R}^3$ throwing an object whose *flying flowmap* Φ_{fly} is with L-bounded spectral radius of the Hessian, the landing error in the throwing plane induced by $\ddot{q}_{\text{cvx}}$, the tube acceleration obtained from Problem Tube-CVX with release window T, denoted as $|\Phi_{\text{fly}}(\ddot{q}_{\text{cvx}})|$, is upper bounded

$$|\Phi_{\text{fly}}(\ddot{q}_{\text{cvx}})| \leq \frac{L}{2} \left\| T\ddot{p}_{\max} \right\|^2. \tag{20}$$

Proof: We rewrite *retraction constraint* in the following form:

$$\begin{aligned} &0 - \Phi_{\text{fly}}(0) \\ &= \nabla\Phi_{\text{fly}}(\dot{r}_T(\ddot{q}_{\text{zero}}), \dot{z}_T(\ddot{q}_{\text{zero}}))^\top \begin{bmatrix} \dot{r}_T(\ddot{q}_{\text{cvx}}) - \dot{r}_T(\ddot{q}_{\text{zero}}) \\ \dot{z}_T(\ddot{q}_{\text{cvx}}) - \dot{z}_T(\ddot{q}_{\text{zero}}) \end{bmatrix} \end{aligned} \tag{21}$$

Then, we apply Lemma 2 with $x = [\dot{r}_T(\ddot{q}_{\text{zero}}), \dot{z}_T(\ddot{q}_{\text{zero}})]^\top$, $y = [\dot{r}_T(\ddot{q}_{\text{cvx}}), \dot{z}_T(\ddot{q}_{\text{cvx}})]^\top$, $f = \Phi_{\text{fly}}$, we have

$$\begin{aligned} &\Big| \Phi_{\text{fly}}(\ddot{q}_{\text{cvx}}) - \Phi_{\text{fly}}(0) \\ &- \nabla\Phi_{\text{fly}}(\dot{r}_T(\ddot{q}_{\text{zero}}), \dot{z}_T(\ddot{q}_{\text{zero}}))^\top \begin{bmatrix} \dot{r}_T(\ddot{q}_{\text{cvx}}) - \dot{r}_T(\ddot{q}_{\text{zero}}) \\ \dot{z}_T(\ddot{q}_{\text{cvx}}) - \dot{z}_T(\ddot{q}_{\text{zero}}) \end{bmatrix} \Big| \end{aligned} \tag{22}$$

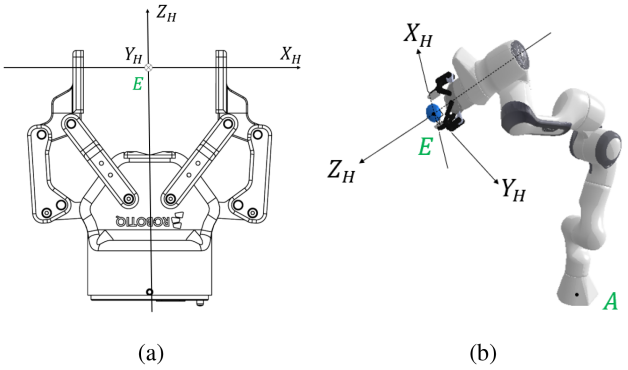
$$= |\Phi_{\text{fly}}(\ddot{q}_{\text{cvx}})| \leq \frac{L}{2} \left\| \begin{bmatrix} \dot{r}_T(\ddot{q}_{\text{cvx}}) - \dot{r}_T(\ddot{q}_{\text{zero}}) \\ \dot{z}_T(\ddot{q}_{\text{cvx}}) - \dot{z}_T(\ddot{q}_{\text{zero}}) \end{bmatrix} \right\|^2 \tag{23}$$

$$\leq \frac{L}{2} \left\| T\ddot{p}_{\max} \right\|^2. \tag{24}$$

□

Discussion on the earlier result is as follows.

1) If we set the *contraction constraint violation* as the minimization objective in Problem Tube-CVX, the error bound effectively serves as a suboptimality certificate. In other words, if the minimal feasible contraction constraint violation is large, there is a high chance that the original



(a) (b)

Fig. 9. Typical installation: Gripper Z-axis is aligned with the last (wrist) joint of the manipulator.

Problem RTV is also not feasible, implying a nonrobustifiable nominal throwing configuration.

2) The bound is still conservative, as one can minimize the norm of tube acceleration as an objective to shrink $\ddot{p}_{\max}$.
3) The use of Hessian is due to lower computational complexity.

a) In Fig. 8, each Hessian map is computed from $m = 100 \times 100$ different throwing velocities in 1 h on a Laptop with an 8-core 1.8 GHz CPU (Intel i7-10510 U).
b) Direct estimation of the Lipschitz constant is expensive with $O(m^2)$ complexity.

4) Lipschitz constant estimation of black-box functions/neural networks is an active field of research [34].

D. Practical Robust Throwing Motion Generation

In this section, we address several practical issues when using tube acceleration to robustify throwing.

1) Infeasible Problem Tube-CVX: Due to the constraints stemming from a particular robot's kinematics and motors (bounds on achievable velocity/acceleration/torques), a throwing configuration may not admit a feasible solution for Problem Tube-CVX. One possibility is to relax the constraints by assuming a tolerance with radius ϵ on the target box. The following relaxed linear inequality constraints can then replace the retraction constraint:

$$\Big|\Phi_{\text{fly}}(0) + \begin{bmatrix} \dfrac{\partial \Phi_{\text{fly}}}{\partial \dot{r}_T(\ddot{q}_{\text{zero}})}, \dfrac{\partial \Phi_{\text{fly}}}{\partial \dot{z}_T(\ddot{q}_{\text{zero}})} \end{bmatrix}^\top \begin{bmatrix} \dot{r}_T(\ddot{q}_{\text{tube}}) - \dot{r}_T(\ddot{q}_{\text{zero}}) \\ \dot{z}_T(\ddot{q}_{\text{tube}}) - \dot{z}_T(\ddot{q}_{\text{zero}}) \end{bmatrix}\Big| \leq \epsilon.$$

2) Finger-Object Collision Avoidance: Once the object has detached, the robot motion still needs to be carefully designed such that the object does not collide with the gripper or robot arm as it falls. While this could be resolved by designing a separate finger-object collision avoidance planning problem, a simple heuristic, which we describe next, can work well in practice.

On typical robot manipulators, the Z-axis of the gripper is aligned with the rotational axis of the last joint, as illustrated in Fig. 9. In this case, the state of the last joint does not affect the translational state of the end-effector, but only the rotational state. This separation allows us to utilize this extra DoF of the last joint for finger-object collision avoidance. The method is straightforward: Move the last joint such that the Cartesian velocity of the end-effector frame E always lies in the $Y - Z$ plane of the gripper.

VII. Experiment

To validate the effectiveness of tube acceleration, we conduct four sets of experiments:

1) batch experiments in simulation;
2) case study in simulation;
3) quantitative experiment on the real robot;
4) qualitative experiment on the real robot.

A. Suboptimality and Computational Efficiency of Convexified Problem

The main objective of this experiment is to focus solely on the *recursive validity* of tube acceleration obtained from Problem Tube-CVX by comparing it to the solution of a generic nonconvex problem. To achieve this, we isolate other uncertainties and set up the simulation with the following conditions.

1) *Projectile Flying-Dynamics:* Only gravity force is applied to the object during free-flying.
2) *No Finger–Object Interaction:* The object's initial flying state (position and velocity) is set to match the end-effector state along the trajectory during the release time window.

Let C be the landing point of the object at the target height, which may not overlap with target box B. We formulate the nonconvex problem as follows, where the maximum landing error ${}^B p^C$ along the release motion is minimized:

Problem Tube-NCP

$$\min_{\ddot{q}_{\text{tube}}} \max_k \left\| {}^A p_k^C - {}^A p^B \right\|_2 \tag{25a}$$

$$\text{subject to: } t_k = \frac{k}{K} T \quad \forall k \in [1, 2, \ldots, K] \tag{25b}$$

$$q_k = q_d + \dot{q}_k t_k + \ddot{q}_{\text{tube}} t_k^2 \tag{25c}$$

$$\dot{q}_k = \dot{q}_d + \ddot{q}_{\text{tube}} t_k \tag{25d}$$

$$z_k = -{}^E p_z^B(q_k) \tag{25e}$$

$$v_k = J(q_k)\dot{q}_k \tag{25f}$$

$$\dot{r}_k = \|v_{k,xy}\| \tag{25g}$$

$$\dot{z}_k = v_{k,z}, \tag{25h}$$

$$r_{\text{land},k} = \Phi_{\text{fly}}(0, z_k, \dot{r}_k, \dot{z}_k) \tag{25i}$$

$${}^A p_k^C = {}^A p_k^E + r_{\text{land},k} \frac{v_{k,xy}}{\|v_{k,xy}\|_2} \tag{25j}$$

$$\dot{q}_{\min} \leq \dot{q}_k \leq \dot{q}_{\max} \tag{25k}$$

$$\ddot{q}_{\min} \leq \ddot{q}_{\text{tube}} \leq \ddot{q}_{\max}. \tag{25l}$$

Table III summarizes the interpretation of the constraints in Problem RTV. Note that constraints (25c)–(25l) have to be satisfied for all K discretization steps. Constraint (25i) actually computes the flying distance before landing and the Cartesian

TABLE IV
PERFORMANCE STATISTICS AMONG 16 THROWING CONFIGURATIONS

Release motion	Max. landing error (cm)	Solving time
zero	17.51 ± 7.29	—
tube-cvx	6.22 ± 3.98	30.20 ± 3.84 ms
tube-ncp-1	4.20 ± 3.42	1.56 ± 0.64 s
tube-ncp-10	3.97 ± 3.58	25.46 ± 15.70 s
tube-ncp-100	3.92 ± 3.62	294.62 ± 345.77 s

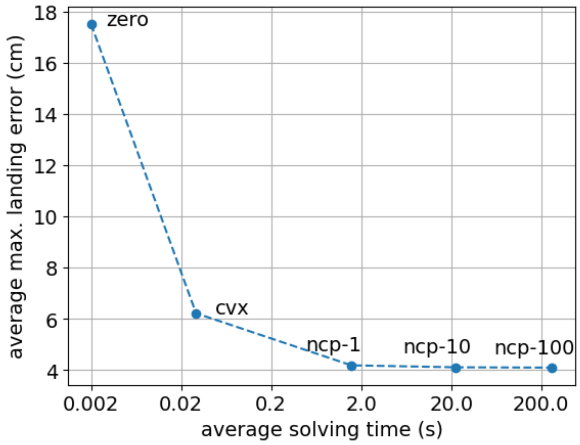
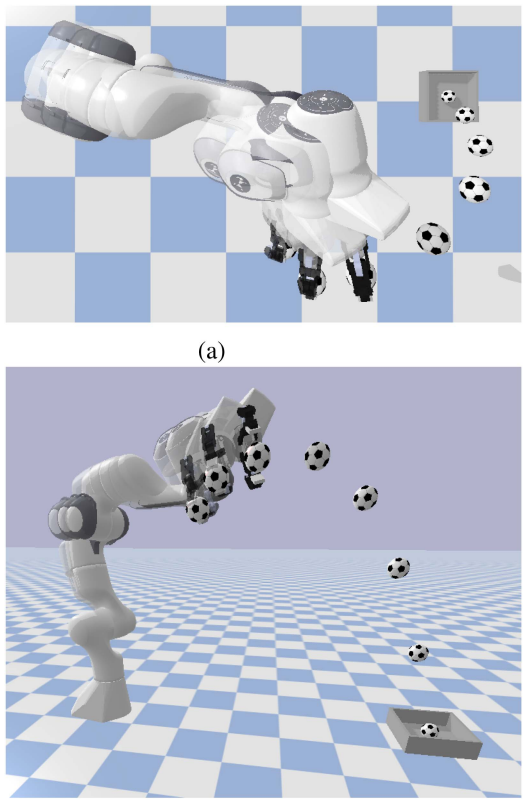


Fig. 10. Performance efficiency tradeoff of different problem formulations.



(a)

(b)

Fig. 11. Visualization of the examined throwing configuration. (a) Top view. (b) Side view.

coordinate of landing point C is obtained by constraint (25j). We set tube acceleration as the decision variable so that the nonconvex problem has the same decision space as Problem Tube-CVX for a fair comparison. It is worth mentioning that the formulation detail of Problem Tube-NCP is optimized through trial-and-error to ensure good solving performance.

The convex problem in Problem Tube-CVX is implemented using CVXPY [35] while the nonconvex problem in Problem Tube-NCP is implemented using SciPy [36] and is solved by the sequential least squares programming solver. The computational performance is evaluated on a Laptop with an 8-core 1.8-GHz CPU (Intel i7-10510U) for timing performance. For both problems, the release phase duration T is set to 100 ms based on empirical observations of hard-coded hand-tuned gripper opening delays[3] for various objects, which typically range from 40 to 140 ms. In practical scenarios, we can consistently implement a minimal gripper opening delay of 40 ms and apply tube acceleration for 100 ms, ensuring the system's robustness against uncertainties in releasing various objects.

To understand the tradeoff between solution quality and solver efficiency for dexterous throwing, we compare the two problem formulations for 16 nominal throwing configurations. The nominal throwing configurations are obtained through velocity hedgehog matching proposed in our previous work [16]. The results are summarized in Table IV and depicted in Fig. 10, where experiments of Problem Tube-NCP with K discretization steps are labeled as "tube-ncp-K."

Discussion:

1) By varying the discretization steps K, we speculate that "tube-ncp-100" with the smallest average worst-case error obtains global-optimal solutions.
2) "tube-ncp-1" exhibits negligible difference compared to "tube-ncp-100."
3) Compared to "tube-ncp-1," "tube-cvx" trades 2-cm average worst-case error for 50× speedup, entering millisecond regime.

The fast and stable solving speed has even greater implications for today's data-driven methods in robotics. Machine learning methods acknowledge the inadequacy of physical models and aim to *improve* task performance from real-world data. Since the typical duration of a throwing trajectory is between 1–1.5 s, when new information from proprietary modules arrives, e.g., updating the object flying dynamics during throwing, a robot with nonconvex formulation would have to abort the executing throwing motion, replan, and try again. In this scenario, the 30-ms computation time of "tube-cvx" opens up the potential for *on-trajectory adjustment* of the robustifying release motion. This capability could greatly enhance the overall agility and adaptability of the throwing robot.

Finally, the tube acceleration obtained from Problem Tube-CVX essentially lies in the intersection between a 5-D subspace and a 7-D polytope, which is a nontrivial solution. This could serve as a strong initial guess for the nonconvex problem.

[3]The gripper opening command is given in advance along the trajectory toward the nominal throwing configuration.

TABLE V
SUMMARY OF THE EXAMINED THROWING CONFIGURATION IN SIMULATION

Property	Notation	Unit	Value
Nominal joint position	q_0	rad	$[-0.44, -0.22, 0.14, -1.57, -0.98, 2.02, 0.]$
Nominal joint velocity	$\dot{q}_0$	rad/s	$[0.64, 1.56, 0.69, 1.84, -0.62, 2.23, -2.4]$
Nominal end-effector position	${}^{A}p_0^E$	m	$[0.51, -0.32, 0.76]$
Nominal end-effector velocity	v_0	m/s	[1.31, 0.70, 0.54]
Nominal end-effector speed	s_0	m/s	1.58
Nominal landing position	${}^{A}p_0^B$	m	$[1.1, 0, 0]$
Robot joint acc. limit	$\ddot{q}_{\max}$	rad/s^2	[9.00, 4.50, 6.00, 7.50, 9.00, 12.00, 12.00]
Resolved acc. at nominal config.	$\ddot{q}_{\text{resolved}}(q, \dot{q})$	rad/s^2	[−1.47, 7.69, −1.85, −6.75, 1.60, −1.85, 0.]
Tube acc. from Convex Program	$\ddot{q}_{\text{cvx}}$	rad/s^2	[−3.46, 4.41, −3.03, −1.03, −0.84, 3.66, 0.]
Tube acc. from Nonconvex Program	$\ddot{q}_{\text{ncp}}$	rad/s^2	[−2.40, 3.80, −2.42, −0.98, −0.15, −0.20, 0.]

Note that the resolved acceleration at the second joint violates the joint acceleration limit of the robot (7.69 versus 4.50).

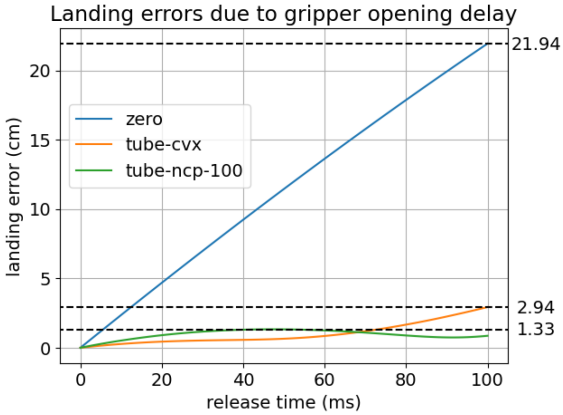


Fig. 12. Comparison of landing errors of different release motions.

B. Case Study in Simulation

We proceed with the examination of a specific throwing configuration to demonstrate the consequences of gripper opening delay and the effectiveness of tube acceleration in mitigating its effects. Fig. 11 visualizes the nominal throwing configuration, and its characteristics are summarized in Table V. The impact of gripper opening delay on different release motions and their resulting landing errors are depicted in Figs. 12 and 13, respectively.

Discussion

1) Zero acceleration during the release window can lead to a significant landing error (>20 cm).
2) Resolved acceleration of gravity at the nominal robot throwing state violates the joint acceleration limit of Franka Emika Panda, rendering it dynamically infeasible for this specific configuration. However, the tube accelerations remain within the allowable limits, showcasing its advantage of safety and feasibility.
3) Tube acceleration from the convex formulation is suboptimal and with a larger worst-case error compared to the solution from the nonconvex formulation (2.94 cm versus 1.33 cm).

It is worth noting that the computational efficiency of the two problem formulations differs significantly. The convex problem is 8900× faster to solve than the nonconvex problem (41 ms versus 365 s) for this specific throwing configuration.

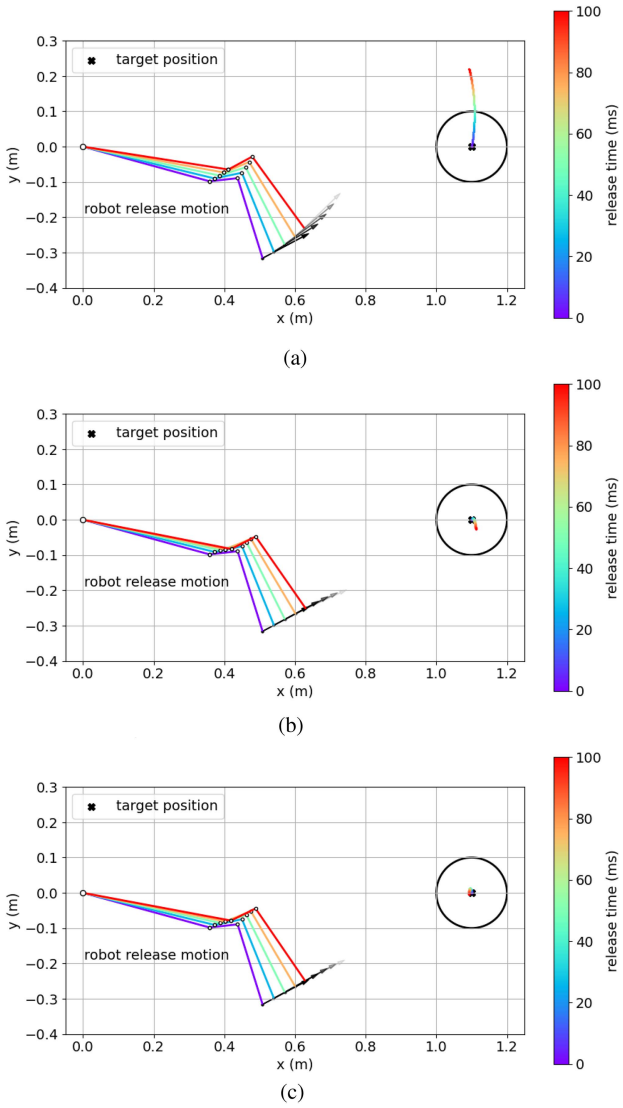


Fig. 13. Release motions driven by different accelerations and their resulting landing positions. (a) Zero acceleration. (b) Tube acceleration from convexified problem. (c) Tube acceleration from the nonconvex problem.

C. Quantitative Experiment on Real Robot Throwing

For the real-robot throwing experiments, we use 7-DoF Franka Emika Panda manipulator mounted with Robotiq 2f-85 parallel gripper. We compare two strategies of motion design

TABLE VI
THROWN OBJECTS IN THE QUANTITATIVE EXPERIMENT

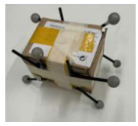
			
Object	grey_ball	small_box_heavy	tennis_ball
Weight (g)	120	100	70
Size (mm)	80 × 60 × 60	80 × 65 × 60	75 × 75 × 75

TABLE VII
LANDING POSITION ERROR STATISTICS OF THE TWO ROBOT MOTION STRATEGIES (TUBE AND ZERO) AFTER THE NOMINAL RELEASE STATE

	Mean (mm)		Std. (mm)	
Object	Tube	Zero	Tube	Zero
grey_ball	74.52	**53.70**	**9.51**	15.60
small_box_heavy	**87.12**	95.30	**7.47**	44.57
tennis_ball	**66.23**	121.07	**9.55**	25.97
overall	**75.88**	88.11	**12.04**	40.74

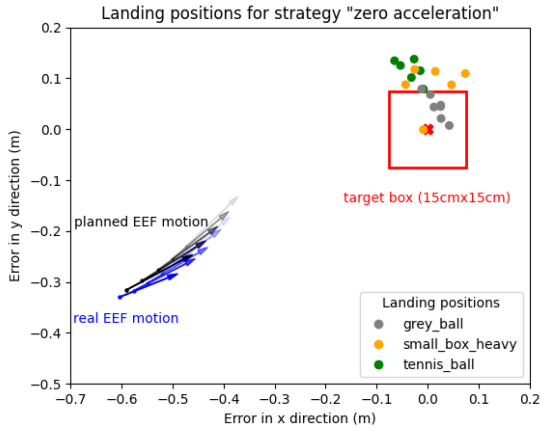
Each strategy-object pair is repeated 5 to 6 times.
The bold means smaller landing position error Mean/Std., indicating better performance.

during the release phase: 1) zero acceleration and 2) tube acceleration. To showcase its potential for real-time adaptation, the tube acceleration is computed from Problem Tube-CVX.

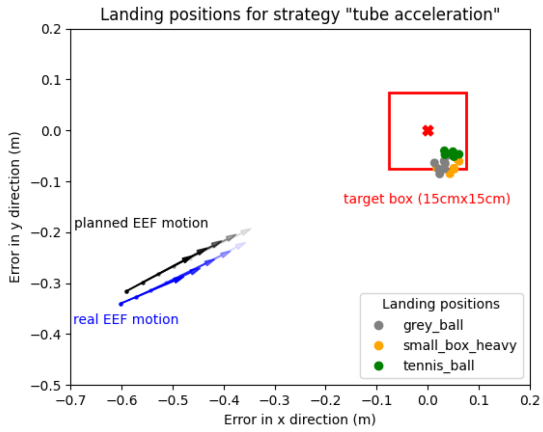
In this quantitative experiment, three objects are selected to be thrown: a 3-D-printed plastic ball ("grey _ ball"), a cardboard box ("small_box_heavy"), and a tennis ball ("tennis_ball"). The photos and properties of each object are listed in Table VI. To accurately track the landing positions, each object is equipped with markers, which are monitored using an OptiTrack motion capture system. The markers' positions are recorded at 240 Hz with a spatial accuracy of 0.2 mm. The nominal throwing configuration is the same one examined in the case study of Section VII-A. For each strategy-object combination, we perform 5–6 throws. The results of the experiment are summarized in Fig. 14 and Table VII.

Discussion on Fig. 14(a): Based on the end-effector motion during the release phase (100 ms after the nominal throwing configuration), we observe that, on average, the objects' release delay follows the order: "tennis _ ball" > "small _ box_heavy" > "grey_ball", which corresponds to the order of their deformability. This suggests that objects with higher deformability tend to experience a longer delay before being released from the gripper. Additionally, the landing positions of the "small _ box_heavy" object are more spread out compared to the other two objects. This spread indicates that the dynamic interaction between the gripper fingers and the box is less predictable.

Discussion on Fig. 14(b): A remarkable fact of the tube acceleration is that the landing positions among the three objects are much more condensed, compared to the zero-acceleration strategy. However, the landing positions in the tube acceleration strategy exhibit a constant offset. This offset primarily stems from the larger trajectory tracking error, as depicted in Fig. 14(b).



(a)



(b)

Fig. 14. Landing positions of the throws with the three objects. The black arrows show the end-effector motion driven by the planned release motion in joint space while the blue arrows show the real end-effector's release motion in one throwing experiment. The red box resembles a virtual target box with a size of 15 cm × 15 cm. (a) Throwing with zero acceleration. (b) Throwing with tube acceleration.

A practical remedy to mitigate this offset is to employ robot dynamics learning techniques, as demonstrated in [37] and [38], to achieve more accurate motion command tracking. In the case of mobile manipulator throwing [16], the offset can be easily eliminated by moving the base aside.

Discussion on Table VII:

1) *grey_ball:* Tube acceleration results in an enlarged mean landing error due to a large trajectory tracking error.
2) *small_box_heavy:* Tube acceleration reduces 83% of the standard deviation of the landing error, whereas zero acceleration suffers from a large variance due to unpredictable gripper– object interaction.
3) *tennis_ball:* Tube acceleration reduces 50% of the mean landing error while zero acceleration experiences a large landing error due to a significant "gripper opening delay."
4) *Overall:* Tube acceleration reduces the mean landing error by 14% and the standard deviation of the landing error by 70% across all the throws.

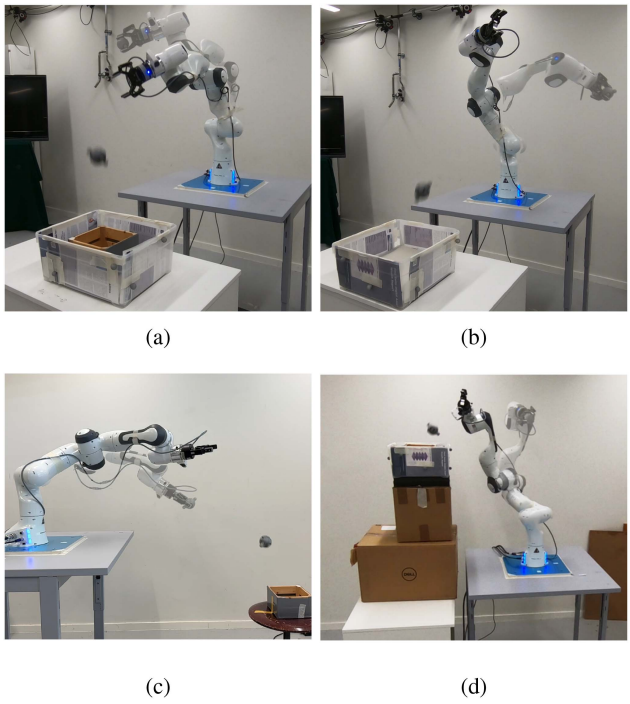


Fig. 15. Four dexterous throwing configurations in the qualitative experiment.

Fig. 16. 18 thrown objects in the qualitative experiment.

D. Qualitative Experiment on Real Robot Throwing

In the qualitative experiment, we demonstrate the robustifying capability of tube acceleration for dexterous throwing configurations. We conduct throwing experiments for one planar throwing configuration and four distinct nonplanar throwing configurations, as shown in Fig. 15. The set of 18 thrown objects used in the experiment is shown in Fig. 16. The properties of the 18 objects are summarized in Table VIII. It is worth noting that the set of objects is arguably the most diverse in the literature to date: TossingBot [8] has a collection of more than 80 different objects, including toy blocks, fake fruits, decorative items, and office objects, but it lacks deformable objects; the work by Monastirsky et al. [9] has seven objects, including two deformable (a sand ball and a squeeze ball), but this selection does not encompass deformable objects with varying contact geometries and material. In contrast, our object set includes not only items from these previous studies but also a folded T-shirt, towels, a plush toy, and various rubber and foam objects, enhancing the diversity of deformability, contact surface, and geometry. To assess the throwing accuracy w.r.t. different target sizes, we design the target box with two levels of error tolerance, represented by the inner small box with dimensions 15 × 15 cm and the outer large box with dimensions 37 × 33 cm, as shown in Fig. 17. This design emulates the concept of Top-1 classification accuracy and Top-5 classification accuracy used in ImageNet [39]. Considering the stochasticity among different throws of the same object, we throw each object 5–8 times. In total, we conduct 1114 real throwing experiments. The summary of the qualitative throwing experiments is available in this Google Drive folder: https://rb.gy/30rord.

Fig. 17. Target box with two levels of error tolerance. Inner small box: 15 × 15 cm. Outer large box: 37 × 33 cm.

The experiment results are presented in Table IX, demonstrating the significant improvement in throwing accuracy achieved through tube acceleration for all five throwing configurations.

Compared to the two previous works on planar throwing of different objects [8], [9], our small box measures 15 × 15 cm, aligning with the target dimensions in Monastirsky et al. [9] and more compact than the 15 × 25 cm box in TossingBot [8]. With a fair setup in target size, our planar throwing driven by tube acceleration achieved a throwing accuracy of 97.3%, surpassing the best-reported accuracy of 85% in TossingBot [8]. While Monastirsky et al. [9] report a 100% accuracy rate, it is worth noting that the objects that failed in our experiments are the squash ball and the wrapped foam tape, both of which are significantly softer than those tested in their study. This difference in object properties could account for the variance in performance.

TABLE VIII
SUMMARY OF OBJECT PROPERTIES IN THE QUALITATIVE EXPERIMENT

Object ID	Object name	Mass (g)	Dimension (mm)	Surface Material	Geometry
1	plastic_ball	92	Radius 30	Plastic	Sphere
2	fake_peach	34	Radius 30	Plastic	Sphere
3	tennis_ball_hard	58	Radius 35	Nylon	Sphere
4	tennis_ball_soft	46	Radius 40	Nylon	Sphere
5	squeeze_ball_blue	18	Radius 35	Foam	Sphere
6	squash_ball_red	27	Radius 30	Rubber	Sphere
7	cardboard_box	135	$80 \times 60 \times 60$	Cardboard	Box
8	carton_box	99	$150 \times 65 \times 28$	Carton	Box
9	cleaning_sponge	8	$95 \times 65 \times 40$	Foam	Box
10	small_towel	42	$110 \times 75 \times 32$	Textile	$\sim$Box
11	medium_towel	90	$140 \times 100 \times 50$	Textile	$\sim$Box
12	empty_jar	15	$80 \times 50 \times 50$	Plastic	Cylinder
13	whiteboard_pen	18	$138 \times 20 \times 20$	Plastic	Cylinder
14	folded_tshirt	130	$210 \times 90 \times 55$	Textile	$\sim$Cylinder
15	fake_banana	68	$190 \times 38 \times 32$	Plastic	Irregular
16	plush_mole	51	$160 \times 70 \times 55$	Textile	Irregular
17	wrapped_rubber_pump	163	$220 \times 200 \times 80$	Thick plastic bag	Irregular
18	wrapped_foam_tape	32	$160 \times 140 \times 60$	Thin plastic bag	Irregular

TABLE IX
SUMMARY OF THROWING SUCCESS RATES IN THE QUALITATIVE EXPERIMENT

			in small box		in large box	
Configuration	Box position (m)	Nominal EEF speed (m/s)	Tube	Zero	Tube	Zero
planar	[1.3, 0, −0.2]	1.77	**97.3%**	57.9%	**100.0%**	91.3%
(a)	[1.1,0,0]	1.58	**91.2%**	9.0%	**97.4%**	79.3%
(b)	[1.1,0,0]	1.55	**40.7%**	1%	**88.5%**	4.1%
(c)	[1.3,0,−0.3]	1.64	**81.4%**	25.5%	**95.6%**	90.4%
(d)	[0, −0.7, 0.7]	1.64	**63.7%**	27.6%	96.0%	**96.5%**

The box positions are expressed in the base frame of the robot.
The bold means higher throwing success rate, indicating better performance.

Moreover, certain throwing configurations, such as the one depicted in Fig. 15(b), may exhibit more variability in release uncertainty, resulting in the lowest throwing accuracy among the five throwing configurations. A comprehensive study of configuration-dependent release uncertainties would be valuable future work to gain deeper insights into the throwing system's fundamental limits.

VIII. UNCERTAINTIES IN ROBOT THROWING AND ROBUSTNESS OF TUBE ACCELERATION

With a limited number of throwing experiments, it is challenging to fully assess the throwing accuracy under various uncertainty sources beyond release dynamics. In this section, we present a framework to model and simulate uncertainties beyond release uncertainty, aiding in the study and synthesis of robust throwing motions. Through this carefully designed stochastic throwing model, we demonstrate that tube acceleration has a great robustifying potential in the presence of other uncertainty sources.

A. Uncertainty Sources in Robot Throwing

Based on our robot-throwing experiments, we have observed that several uncertainty sources have a detrimental effect on the final throwing outcome when a grasped object is thrown. These sources of uncertainty are as follows.

1) *Robot dynamics:* The unavailability of an accurate rigid-body dynamics model for the robot manipulator, coupled with the unmodeled effects of robot dynamics (such as elastic actuators and friction), leads to deviations between the intended motion command and the actual robot motion.
2) *Release dynamics:* During the release phase, the object's dynamics transition from forced motion to free-flying. Consequently, the object's release state may not align perfectly with the end-effector state.
3) *Object flying dynamics:* Modeling air drag with generic object shapes can be challenging, introducing uncertainties in the object's flight trajectory.

Addressing these sources of uncertainty is crucial for improving throwing accuracy and ensuring the robustness of the throwing process.

B. Mathematical Modeling of Throwing Uncertainty

Consider a robot motion segment $(q(\cdot), \dot{q}(\cdot))$ during the release phase, we can model robot dynamics uncertainty and release dynamics uncertainty by defining tubes around the joint motion sequence and the end-effector motion sequence, respectively. Within these tubes, any state can represent a possible object release state ξ^0, which is the instant when the object starts its free-flying trajectory.

Mathematically, for a given robot motion segment $(q(\cdot), \dot{q}(\cdot))$ and uncertainty set Σ, there exists a set (tube) of possible object

release states $\mathcal{M}$ with a probability measure P induced by Σ. Due to the influence of gravity, the set of plausible release states $\mathcal{M}$ is then "push forwarded" onto the landing plane, resulting in a probability measure of landing positions $P_{\mathcal{L}}$ in $\mathbb{R}^2$. Ideally, the landing positions' probability distribution $P_{\mathcal{L}}$ induced by tube acceleration should concentrate around the nominal landing position p^*. In the perfect case when $P_{\mathcal{L}}$ is a Dirac measure δ_{p^*} at the nominal landing position p^*, the objects will consistently land at the same spot after repetitive throws, resembling a pile or mountainlike formation.

C. Simulating Throwing Uncertainty

To visualize the effect of throwing uncertainty, we employ Monte Carlo simulation to sample random throwing configurations and collect their corresponding landing positions. For the motion segment $(q(\cdot), \dot{q}(\cdot))$ with a duration of 100 ms, we discretize the segment into a sequence of 11 joint states with a 10-ms interval, denoted as $\Xi = \{(q^k, \dot{q}^k)\}_{k=0}^{10}$. Since Panda has seven joints, the space of uncertainty set Σ can be quite large. To manage this, we make the following design choices for the uncertainty set around each throwing configuration $(q^k, \dot{q}^k)$.

1) The axis of Panda's 1st joint is perpendicular to the ground and typically experiences low payload. In practice, we observe high command tracking accuracy on the 1st joint; therefore, we set no uncertainty on the 1st joint.
2) The state of the last (7th) joint of Panda does not affect the translational state of the end-effector, but only the end-effector's orientational state. Since we assume point-mass objects in this work, we set no uncertainty on the 7th joint.
3) For the five joints in the middle of the kinematic chain (2nd–6th joints), we sample joint state disturbances from the following sets: $\delta q_i \in \{0.0, 0.02, 0.05\}$, $\delta \dot{q}_i \in \{0.0, 0.03, 0.08\}$, for all $i \in \{2, 3, 4, 5, 6\}$. This results in a total of $(3 \times 3)^5 = 59\,049$ possible joint state disturbances. The choice of three different error levels for joint position and joint velocity is based on our observation of typical error values on the real robot.

1) Discussion on Release Dynamics Uncertainty: On one hand, the uncertainty in robot dynamics designed earlier might already incorporate the release dynamics uncertainty. On the other hand, one can consider adding directional-dependent uncertainty in the object release state based on the gripper orientation.

2) Discussion on Object Flying Dynamics Uncertainty: In [40], accurate prediction of object landing position is achieved by explicitly modeling air-drag and the Magnus effect with calibrated model parameters. Their model can also be used to study object flying dynamics uncertainty by adjusting model parameters.

We acknowledge these aspects as potential future work for further exploration.

D. Robustness of Tube Acceleration

Interestingly, tube acceleration also demonstrates an improvement in landing accuracy in the stochastic setting. In the Monte Carlo simulation shown in Fig. 18, each throwing configuration $(q^k, \dot{q}^k)$ on the discretized throwing motion segment $\Xi = (q^k, \dot{q}^k)_{k=0}^{10}$ is repeated 100 times and disturbed by 100 sampled joint state disturbances out of the 59 049 possible disturbances, as designed in Section VIII-C. Therefore, each plot in Fig. 18 comprises a total of $11 \times 100 = 1100$ random throwing simulations.

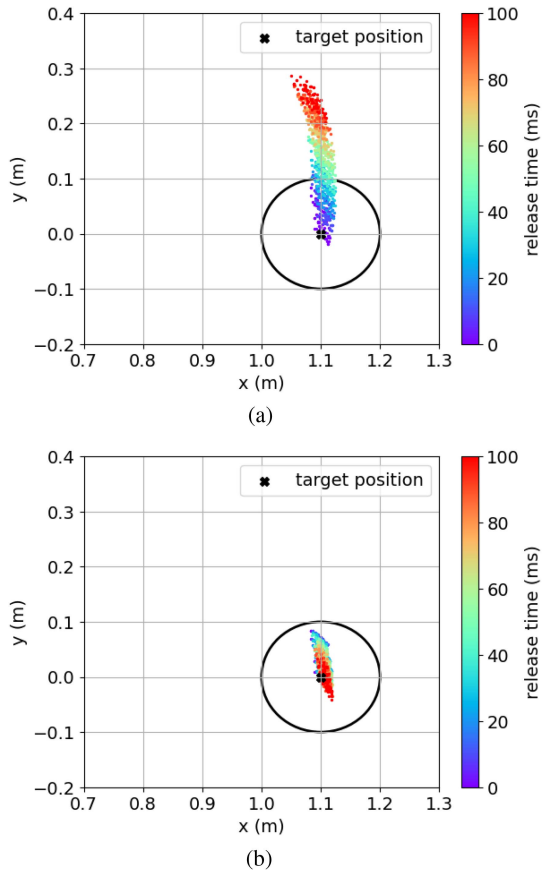


Fig. 18. Monte Carlo simulation of landing positions with two different robot motions. The nominal throwing configuration is the one examined in Sections VII-B and VII-C. The random seeds are identical in the two plots. The robot base is at the origin, and the target is at (1.1, 0.0, 0.0) in the robot frame (the same height as the table that mounts the robot). The black circle resembles a target bin with a radius of 10 cm. (a) Throwing without tube acceleration. (b) Throwing with tube acceleration.

Fig. 18(a) illustrates the population of landing positions under the robot motion *without tube acceleration* (constant joint velocity). As the release time progresses, the landing positions gradually deviate farther away from the target position. In comparison, the population of landing positions under the robot motion *with tube acceleration* is much more concentrated around the target position, as shown in Fig. 18(b). This indicates the ability of tube acceleration to reduce the variabilities and uncertainties from various sources, resulting in improved accuracy.

IX. Limitations of Kinematic Release Model for Off-CoM Grasps

In this work, the RTV problem is based on the assumption that the intricate release dynamics can be effectively approximated

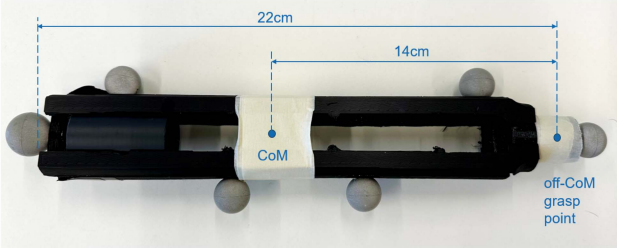
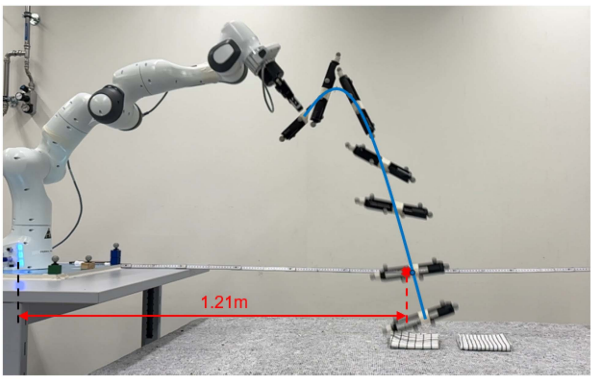
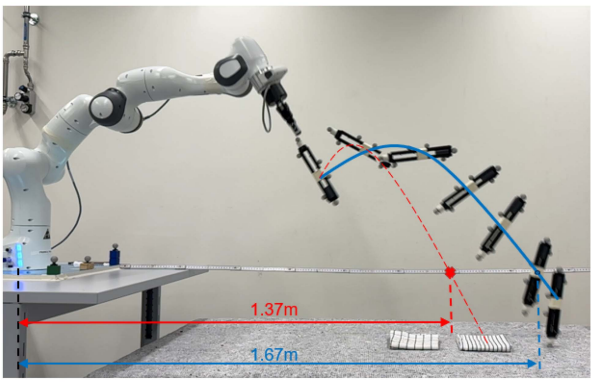


Fig. 19. 3-D-printed bar to study the effect of grasp CoM offset on throwing. The two designed grasp points, the CoM and one end, are covered with paper tape to maintain consistent contact properties.



(a)



(b)

Fig. 20. Comparison of throws with different grasps. (a) Accurate throw with CoM grasp. (b) Large throwing error with CoM offset.

by a kinematic "gripper opening delay" model, so that the release motion, designed to be robust against the *unknown time delay* in the kinematic model, can robustify the *true uncertainties* in the release dynamics. The experimental results in Sections VII-C and VII-D validate the fidelity of the kinematic release model when objects are grasped at their CoM. However, this model fails to capture the release dynamics when objects are grasped with CoM offset.

To account for grasp offset without modifying our algorithm, we translate the gripper frame from the original finger frame E to a virtual frame G, attached to the gripper and overlapping with the object's CoM upon grasping. This is a straightforward adjustment based on the kinematic release model.

To evaluate how the location of the grasp influences the accuracy of the throws, we 3-D-printed an object with known

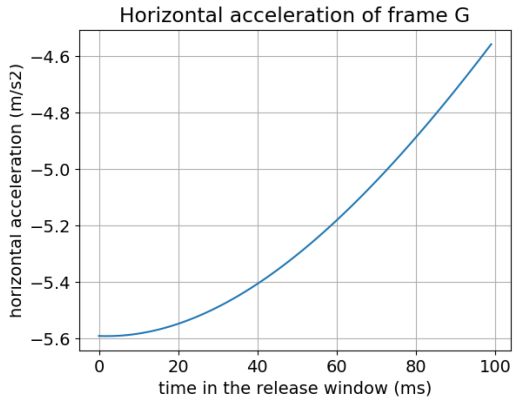


Fig. 21. Horizontal acceleration of the virtual frame G is negative throughout the 100-ms release window.

mass distribution, as shown in Fig. 19, which we throw using two different grasps in the same planar throwing configuration. The corresponding release motions are generated by Problem Tube-CVX, assuming projectile flying dynamics at the object's CoM. Snapshots of the throws are shown in Fig. 20.

As shown in Fig. 20(a), when grasped at CoM, the nominal landing position is 1.21 m, and the landing position of the object CoM is almost identical to the nominal landing position. However, when grasped with a 0.14-m CoM offset, the object lands 0.3 m further away from the model prediction (1.67 m versus 1.37 m, shown in Fig. 20(b)). The larger flying distance indicates that the CoM is accelerated drastically during the release while the computed tube acceleration generates a negative horizontal acceleration at the virtual frame G (shown in Fig. 21). This finding confirms that our kinematic release model is not just quantitatively, but also qualitatively, inaccurate for off-CoM grasps.

X. Conclusion

In the realm of robot dexterous throwing, a stringent challenge is to suppress the release uncertainty to throw various objects with various robot-throwing motions. Conventional end-to-end learning methods for robot throwing would require training data for each nominal throwing motion and, hence, exhibit limited scalability. This work proposes tube acceleration, a novel solution concept to robustify against release uncertainty, given robot geometry, actuator limits, and nominal throwing configuration. Although based on a simple kinematic model of gripper opening delay, strong empirical performance demonstrates the descriptive power of the macroscopic model in capturing microscopic release dynamics. To fully utilize the simplicity of the surrogate model, we design a convexification scheme to obtain tube acceleration under 50 ms with bounded approximation errors. As a result, the online-generated robustifying motion drives the robot to throw diverse objects with a similar accuracy compared to end-to-end learning approaches, without relying on any experimental data. We hope this plug-and-play robust throwing algorithm could pave the way toward ubiquitous throwing robots.

This framework has a notable limitation: It assumes that the object is grasped at its CoM, making the kinematic gripper opening delay an effective surrogate model. However, empirical observations have revealed that when the object is grasped with a significant CoM offset, a simple kinematic model fails to capture the actual release dynamics accurately. Consequently, a crucial aspect of our future work involves synthesizing robust motion for objects grasped with a CoM offset while maintaining a similar level of solving efficiency and reliability.

ACKNOWLEDGMENT

The authors would like to thank Aradhana Nayak and Lukas Huber for technical discussions, James Hermus and Kunpeng Yao for their feedback on the manuscript, Albéric de Lajarte and Elise Jeandupeux for system integration, and Tristan Bonato for 3D prototyping. The authors are also grateful for the invaluable feedback and constructive comments by the anonymous reviewers.

REFERENCES

[1] K. M. Lynch and M. T. Mason, "Dynamic nonprehensile manipulation: Controllability, planning, and experiments," *Int. J. Robot. Res.*, vol. 18, no. 1, pp. 64–92, 1999.

[2] T. Senoo, A. Namiki, and M. Ishikawa, "High-speed throwing motion based on kinetic chain approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 3206–3211.

[3] W. Mori, J. Ueda, and T. Ogasawara, "1-DoF dynamic pitching robot that independently controls velocity, angular velocity, and direction of a ball: Contact models and motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 1655–1661.

[4] Y. Gai, Y. Kobayashi, Y. Hoshino, and T. Emaru, "Motion control of a ball throwing robot with a flexible robotic arm," *Int. J. Comput. Inf. Eng.*, vol. 7, no. 7, pp. 937–945, 2013.

[5] A. Pekarovskiy and M. Buss, "Optimal control goal manifolds for planar nonprehensile throwing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 4518–4524.

[6] H. Miyashita, T. Yamawaki, and M. Yashima, "Control for throwing manipulation by one joint robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 1273–1278.

[7] O. Taylor and A. Rodriguez, "Optimal shape and motion planning for dynamic planar manipulation," *Auton. Robots*, vol. 43, no. 2, pp. 327–344, 2019.

[8] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to throw arbitrary objects with residual physics," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1307–1319, Aug. 2020.

[9] M. Monastirsky, O. Azulay, and A. Sintov, "Learning to throw with a handful of samples using decision transformers," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 576–583, Feb. 2023.

[10] R. G. Cohen and D. Sternad, "State space analysis of timing: Exploiting task redundancy to reduce sensitivity to timing," *J. Neuriophysiol.*, vol. 107, no. 2, pp. 618–627, 2012.

[11] R. D. Howe and M. R. Cutkosky, "Practical force-motion models for sliding manipulation," *Int. J. Robot. Res.*, vol. 15, no. 6, pp. 557–572, 1996.

[12] I. Kao, K. Lynch, and J. W. Burdick, *Contact Modeling and Manipulation*. Berlin, Germany: Springer, 2008, pp. 647–669, doi: 10.1007/978-3-540-30301-5_28.

[13] J. Kober, K. Muelling, and J. Peters, "Learning throwing and catching skills," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5167–5168.

[14] Y. Zhang, J. Luo, and K. Hauser, "Sampling-based motion planning with dynamic intermediate state objectives: Application to throwing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 2551–2556.

[15] M. Bombile and A. Billard, "Dual-arm control for coordinated fast grabbing and tossing of an object: Proposing a new approach," *IEEE Robot. Autom. Mag.*, vol. 29, no. 3, pp. 127–138, Sep. 2022.

[16] Y. Liu, A. Nayak, and A. Billard, "A solution to adaptive mobile manipulator throwing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 1625–1632.

[17] M. Bombile and A. Billard, "Bimanual dynamic grabbing and tossing of objects onto a moving target," *Robot. Auton. Syst.*, vol. 167, 2023, Art. no. 104481.

[18] M. Okada, A. Pekarovskiy, and M. Buss, "Robust trajectory design for object throwing based on sensitivity for model uncertainties," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 3089–3094.

[19] M. Okada, S. Oniwa, and W. Hijikata, "Robust throwing design based on dynamic sensitivity analysis," *Mech. Eng. J.*, vol. 5, no. 1, pp. 17–00442, 2018. [Online]. Available: https://www.jstage.jst.go.jp/article/mej/5/1/5_17-00442/_article/-char/en

[20] M. Okada and T. Sekiguchi, "Throwing motion design based on minimum sensitivity with respect to error covariance of robot dynamic parameters," *Mech. Eng. J.*, vol. 8, no. 1, pp. 20–00299, 2021. [Online]. Available: https://www.jstage.jst.go.jp/article/mej/8/1/8_20-00299/_article/-char/en

[21] L. Chen et al., "Decision transformer: Reinforcement learning via sequence modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 15084–15097.

[22] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1711–1718, Jul. 2018.

[23] J. Wu et al., "TidyBot: Personalized robot assistance with large language models," *Auton. Robots*, vol. 47, pp. 1087–1102, 2023.

[24] S. S. M. Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for softly catching a flying object: Theory and experiment," *IEEE Trans. Robot.*, vol. 32, no. 2, pp. 462–471, Apr. 2016.

[25] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.

[26] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 2242–2253.

[27] L. S. Pontryagin, E. Mishchenko, V. Boltyanskii, and R. Gamkrelidze, *The Mathematical Theory of Optimal Processes*. New York, NY, USA: Taylor & Francis, 1962.

[28] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018. [Online]. Available: https://papers.nips.cc/paper_files/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html

[29] R. T. Q. Chen, B. Amos, and M. Nickel, "Learning neural event functions for ordinary differential equations," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/95e1533eb1b20a97777749fb94fdb944-Abstract.html

[30] M. V. Lubarda and V. A. Lubarda, "A review of the analysis of wind-influenced projectile motion in the presence of linear and nonlinear drag force," *Arch. Appl. Mechanics*, vol. 92, no. 7, pp. 1997–2017, 2022.

[31] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 729–736.

[32] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[33] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87, Berlin, Germany: Springer, 2003.

[34] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of Lipschitz constants for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[35] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.

[36] P. Virtanen et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[37] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau5872.

[38] F. Khadivar, S. Gupta, W. Amanhoud, and A. Billard, "Efficient configuration exploration in inverse dynamics acquisition of robotic manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 1934–1941.

[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[40] Y.-B. Jia, M. Gardner, and X. Mu, "Batting an in-flight object to the target," *Int. J. Robot. Res.*, vol. 38, no. 4, pp. 451–485, 2019.

Yang Liu (Student Member, IEEE) received the B.Sc. degree in mechanical engineering from Jilin University, Changchun, China, in 2017, and the M.Sc. degree in robotics, systems, and control from ETH Zurich, Zurich, Switzerland, in 2020. He is currently working toward the Ph.D. degree in robotics with Learning Algorithms and Systems Laboratory (LASA), Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland.

His research interests include dexterous dynamic manipulation and adaptive motion planning for robots.

Aude Billard (Fellow, IEEE) received the B.Sc. and M.Sc. degrees in physics from the Swiss Institute of Technology Lausanne (EPFL), Lausanne, Switzerland, and the Ph.D. degree in artificial intelligence from the University of Edinburgh, Edinburgh, U.K., in 1994, 1995, and 1998, respectively.

She is currently a Full Professor, the Head of the LASA laboratory, and the Associate Dean for Education with the School of Engineering, EPFL. Her research interests include the fields of machine learning and robotics with a particular emphasis on fast and reactive control and on safe human–robot interaction.

Prof. Billard is currently the President of the IEEE Robotics and Automation Society, the Director of the ELLIS Robot Learning Program, and the Co-Director of the Robot Learning Foundation, a nonprofit corporation that serves as the governing body behind the Conference on Robot Learning (CoRL), and leads the Innovation Booster Robotics, a program funding technology transfer in robotics and powered by the Swiss Innovation Agency, Innosuisse. She is the recipient of numerous recognitions, among which the Intel Corporation Teaching award, the Swiss National Science Foundation career award, the Outstanding Young Person in Science and Innovation from the Swiss Chamber of Commerce, the IEEE RAS Distinguished Award, and the IEEE-RAS Best Reviewer Award. She was a Plenary Speaker at major robotics, AI, and Control conferences (ICRA, AAAI, CoRL, HRI, CASE, ICDL, ECML, L4DC, IFAC Symposium, ROMAN, Humanoids, and many others) and acted in various positions on the organization committee of numerous International Conferences in Robotics. Her research received numerous best conference paper awards, as well as the prestigious King-Sun Fu Memorial Award for the best IEEE TRANSACTION IN ROBOTICS paper, and is regularly featured in premier venues (*BBC*, IEEE SPECTRUM, and *WIRED*).