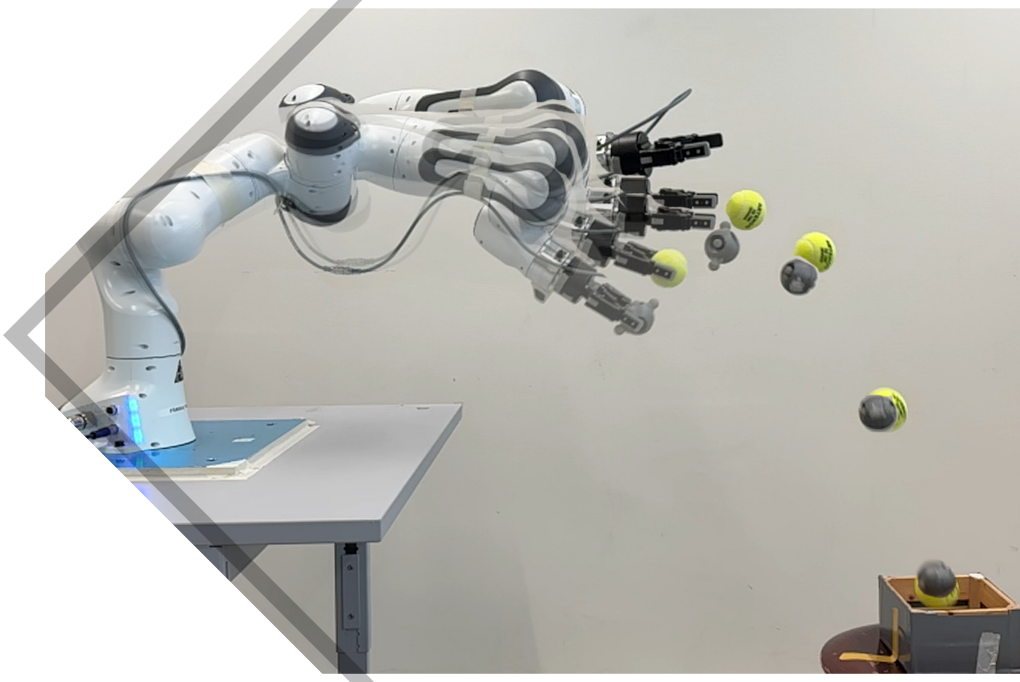
DARKO

Dynamic Agile Production Robots That Learn and Optimise Knowledge and Operations

H2020-ICT-2020-2 Grant agreement no: 101017274



DELIVERABLE 4.6

Robot control for dynamic throwing of objects

Dissemination Level: PUBLIC

Due date: month 42 (December 2024)

Deliverable type: Report and Software

Lead beneficiary: EPFL

1 Introduction

Robots equipped with throwing capabilities have remarkable potential to enhance their object transportation skills, achieving unprecedented levels of dexterity and efficiency for intralogistics. By carefully transferring momentum from the robot to the object during the throwing motion and relying on gravity for the free-flying trajectory, robots can minimize unnecessary movements of their heavy bodies. This results in a smaller motion footprint and lower energy consumption—two highly desirable characteristics for the next generation of collaborative mobile manipulators. These robots will operate outside confined industrial cages and navigate in cluttered environments, with capacity-limited batteries.

In **D4.5 - Preliminary Robot Control for Dynamic Throwing of Objects**, EPFL reported methods and algorithms for fast and adaptive (FAT) throwing planning, published in IROS 2022 [33], marking a key technical contribution to automated throwing planning. UNIPI presented methods for throwing with a pneumatic tool equipped with a bi-directional valve to jet the object toward the target using compressed air, which is a highly desirable solution for the suction cup-dominated grasps in industry.

Despite these past achievements, building a robotic throwing system with a higher Technology Readiness Level (TRL) remains a significant challenge. Such a system, designed to throw a variety of objects for agile production, requires a thorough understanding of the physics of the throwing process and the development of reliable algorithms for consistent and self-improving performance. In addition, for such robots to operate in unstructured and unpredictable dynamic environments, it is also necessary to effectively utilize elastic energy for throwing, to increase the object's reachable space, reduce energy consumption, and minimize the robot's throwing motion footprint for safe operation.

With these goals in mind, this deliverable presents our scientific progresses in designing robust control approaches for robot throwing.

1.1 Contributions

This deliverable reports on five contributions:

1. **Throwing robust to stochastic release conditions (EPFL):** We develop a method robust to the stochastic effects, due to unmodeled deformations and friction, that arise at the release time, i.e. when the object detaches from the gripper, and demonstrate that it enables to more reliably throw a variety of objects, including deformable ones.

2. **Physical Modeling of Throwing (EPFL):** We investigate in more details the physics of the release dynamics, focusing in particular on the friction effect on the velocity of the released object. We show that our modeling offers more precise estimate of the flying dynamics after release and more accurate prediction of the landing pose of the object.

3. **Learning and Adaptation from Past Throwing Experiences (EPFL):** We tackle the problem of improving accuracy of throwing by learning from past throws and active sampling to improve throw accuracy. To this end, we develop an approach that combines physics-based modeling of the object's flying dynamics with machine learning?

4. **Throwing with the Elastic Wrist (UNIPI):** How to throw using an *elastic wrist* to enhance the dynamics of robotic throwing?

5. **Throwing with the Elastic Manipulator (TUM):** How to throw using an *elastic arm* to enhance the dynamics of robotic throwing?

2 Robust Throwing, Physical Models for Throwing and Learning to Throw-Flip (EPFL)

2.1 Robust dexterous throwing against release uncertainty

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 (As shown in Fig. 1a). The uncertainty associated with releasing objects of various geometries, surface materials and deformabilities poses significant challenges. In the literature of robot throwing, the majority of previous approaches were limited to throwing one specific object type ([34]: a wooden block; [49, 38, 18, 36, 28, 40, 39, 41]: a ball; [44]: a square plastic plate; [5, 6]: heavy boxes). Although two recent works [55, 37] proposed end-to-end learning methods to throw various objects, their reliance on massive real-robot throwing trials and black-box learning models raises concerns on their scalability and wide adoption. More specifically,

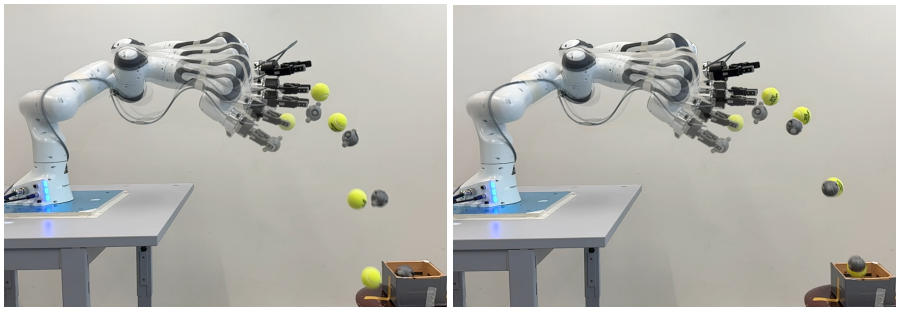
Zeng et al. [55] provides an end-to-end learning approach for throwing various objects. The throwing configuration is generated from the throwing velocity predicted by a trained model given the target box position and object image. The learned model can handle a large set of objects and multiple target 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.[37] utilize Decision Transformer[9], 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 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[1] 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 above reviewed two contributions in the following key aspects: (1) While [55, 37] 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 [37] 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.

Facing the release uncertainty, EPFL proposed a novel method to synthesize robust throwing motion during the release phase, such that the same manipulator motion is valid to throw different objects (shown in Fig. 1b). The method encapsulates all uncertainties resulting from complex contact dynamics in a surrogate kinematic model of their resulting effect on "gripper opening delay". Then EPFL introduces 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, as illustrated in Fig. 2. Through insightful observations, the primal robust throwing problem is relaxed to a convex one with a tight

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



(a) (b)

Figure 1: The robot throws a hard plastic ball and a deformable tennis ball. The motion commands and gripper opening times are identical in each subfigure. (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 the ball fails to fall into the target box. (b) The proposed tube acceleration compensates for release time uncertainty. Thus, despite different escape times and hence different flying trajectories, the two objects land in identical locations. This is possible as the robot traverses within the set of valid throwing configurations – the tube acceleration.

error bound and leads to online computation of robust throws on a 7-DoF robot arm (<50ms).

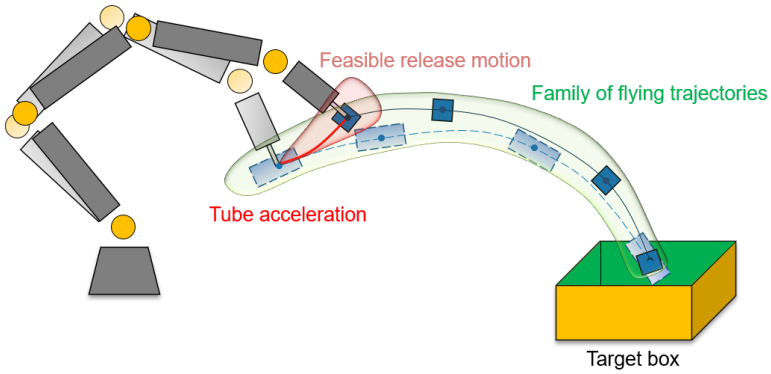


Figure 2: Schematic for robust throwing. During the release phase, the robot end-effector traverses a family of valid projectile trajectories encapsulated by the tube acceleration. As a result, the landing outcome is agnostic to the exact object release time and hence robust to release dynamics.

The tube acceleration method achieves high accuracy and success rate at throwing a variety of complex objects with diverse throwing configurations, without training on experimental data. For planar throwing which is common in the literature, robust throwing with tube acceleration achieved 97.3% accuracy, which is higher than the best-reported accuracy (85%) with end-to-end learning methods [55]. This work on robust dexterous throwing is the first in the literature that can throw various objects with dexterous postures. It is published as "*Tube Acceleration: Robust Dexterous Throwing Against Release Uncertainty,*" *Liu and Billard, IEEE Transactions on Robotics, 2024* [30].

2.1.1 Preliminaries

1) Geometric Modeling of Throwing

The geometric modeling is illustrated in Fig. 3, with notations introduced in Table 1. 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 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.

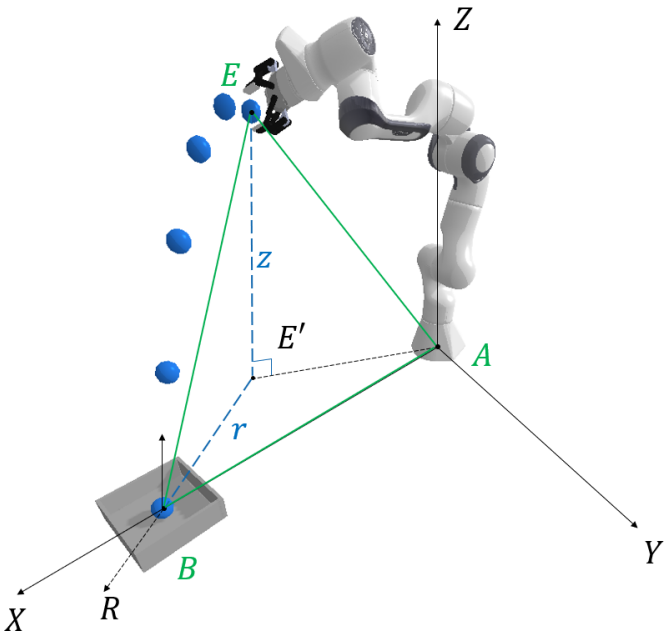


Figure 3: Geometry of 3D throwing

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

Table 1: Notations for geometric modeling in Fig. 3.

2) 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 $\dot{\xi} = f_{fly}(\xi)$. The flying trajectory of f_{fly} starting from state ξ^0 are denoted as $\zeta_{f_{fly},\xi^0}(t) : [0, +\infty] \to \mathbb{R}^4$. 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:

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

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') (Th. 3.5 [27]). 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 $f_{BRT}(\xi^0) : \mathbb{R}^4 \to \mathbb{R}$, with the following interpretations:

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

3) Object Flying Flowmap

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 state at a given time or the state upon a certain *event* happening.

Using the *adjoint sensitivity method* (Pontryagin et al., 1962 [45]), Neural ODE [12] 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 error.

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 [11] 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:

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

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.

2.1.2 Robust Throwing Formulation

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

- End-effector's state remains inside the BRT for a time window $[0, T]$,
- 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.

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 Recursive Task-Validity (RTV) Problem can be formulated as follows:

Problem *RTV*

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

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

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

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

$$[v_x, v_y]^\top [-{}^E p^B_y(q), {}^E p^B_x(q)] = 0, \tag{1e}$$

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

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

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

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

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

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

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

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

Problem *RTV* is difficult to solve due to the functional decision variables and the non-convex constraints. In order to let the robust throwing motion generator be ready to handle large amounts of throwing configurations for dexterous throwing, we choose to convexify the primal problem and obtain the following formulation of **Problem *Tube-CVX***. Blue variables are induced by constant tube acceleration $\ddot{q}_{tube}$, while black variables can be viewed as *parameters* of the program and hence are treated as fixed in the solver.

(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

Table 2: Explanation of constraints in **Problem *RTV***.

Problem *Tube-CVX*

$$\text{Find: } \{\ddot{q}_{tube}\} \tag{2a}$$
$$\text{subject to: } q_T = q_0 + T\dot{q}_0, \tag{2b}$$
$$\dot{q}_T = \dot{q}_0 + T\ddot{q}_{tube}, \tag{2c}$$
$$r_T = -\left\| {}^{E}p^{B}_{xy}(q_T) \right\|_2, \tag{2d}$$
$$z_T = -{}^{E}p^{B}_{z}(q_T), \tag{2e}$$
$$v_T(q_T, \dot{q}_T) = J(q_T)\dot{q}_T, \tag{2f}$$
$$[v_{T,x}, v_{T,y}]^\top [-{}^{E}p^{B}_{y}(q_T), {}^{E}p^{B}_{x}(q_T)] = 0, \tag{2g}$$
$$\dot{r}_T = \|v_{T,xy}\|_2, \tag{2h}$$
$$\dot{z}_T = v_{T,z}, \tag{2i}$$
$$-\Phi_{fly}(0) = \left[\frac{\partial \Phi_{fly}}{\partial \dot{r}_T(\ddot{q}_{zero})}, \frac{\partial \Phi_{fly}}{\partial \dot{z}_T(\ddot{q}_{zero})} \right]^\top \begin{bmatrix} \dot{r}_T(\ddot{q}_{tube}) - \dot{r}_T(\ddot{q}_{zero}) \\ \dot{z}_T(\ddot{q}_{tube}) - \dot{z}_T(\ddot{q}_{zero}) \end{bmatrix} \tag{2j}$$
$$\dot{q}_{\min} \le \dot{q}_T \le \dot{q}_{\max}, \tag{2k}$$
$$\ddot{q}_{\min} \le \ddot{q}_{tube} \le \ddot{q}_{\max}. \tag{2l}$$

2.1.3 Experiments

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 during the release phase: zero acceleration and tube acceleration.

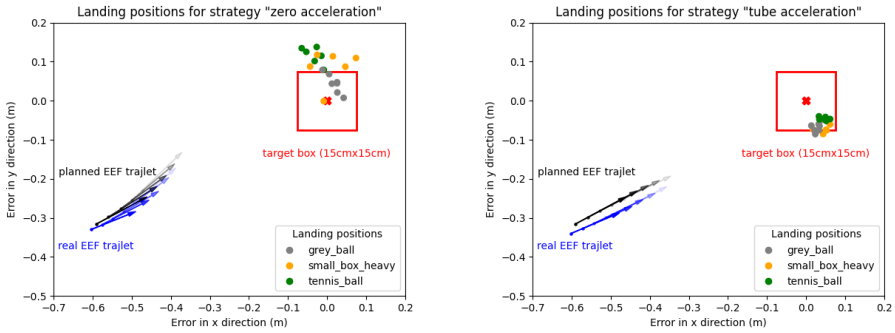
1) Quantitative experiment on robust throwing

In this quantitative experiment, 3 objects are selected to be thrown: a 3D-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 3. 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 240Hz with a spatial accuracy of 0.2mm. For each strategy-object combination, we perform 5-6 throws. The results of the experiment are summarized in Fig.4 and Table4.

Discussion on Fig. 4a: 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

			
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 3: Thrown objects in the quantitative experiment.



(a) throwing with zero acceleration **(b)** throwing with tube acceleration

Figure 4: The landing positions of the throws with the 3 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 15cm×15cm.

between the gripper fingers and the box is less predictable.

Discussion on Fig. 4b: A remarkable fact of the tube acceleration is that the landing positions among the 3 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.4b. A practical remedy to mitigate this offset is to employ robot dynamics learning techniques, as demonstrated in[24, 26], to achieve more accurate motion command tracking. In the case of mobile manipulator throwing [33], the offset can be easily eliminated by moving the base aside.

Discussion on Table 4:

- **grey_ball**: Tube acceleration results in an enlarged mean landing error due to a large trajectory tracking error.
- **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.
- **tennis_ball**: Tube acceleration reduces 50% of the mean landing error, while zero

	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

Table 4: Landing position error statistics of the two robot motion strategies (Tube and Zero) after the nominal release state. Each strategy-object pair is repeated 5-6 times.

.

acceleration experiences a large landing error due to a significant "gripper opening delay".

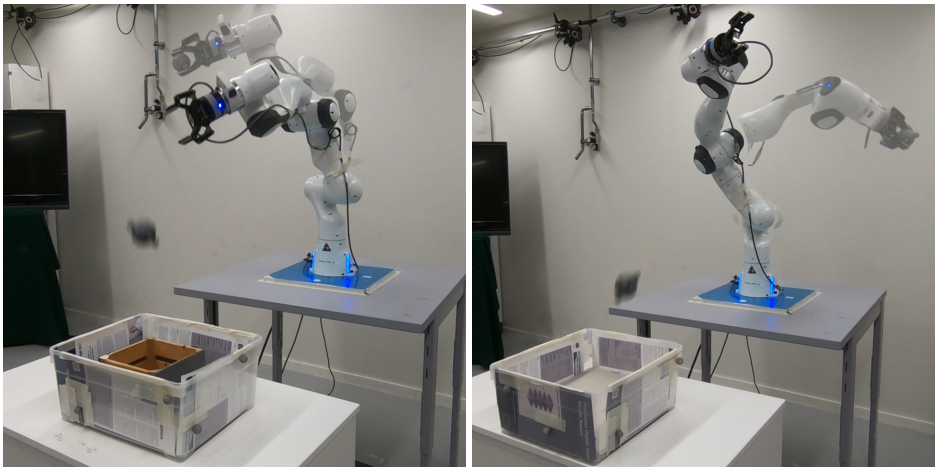
- **Overall**: Tube acceleration reduces the mean landing error by 14% and the standard deviation of the landing error by 70% across all the throws.

2) Qualitative experiment on robust throwing

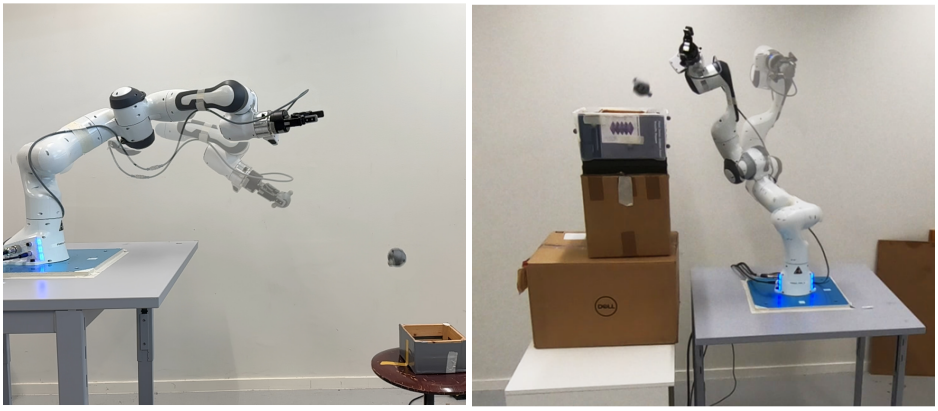
In the qualitative experiment, we demonstrate the robustifying capability of tube acceleration for dexterous throwing configurations. We conduct throwing experiments for 1 planar throwing configuration and 4 distinct non-planar throwing configurations, as shown in Fig. 5. The set of 18 thrown objects used in the experiment is shown in Fig. 6a. It is worth noting that the set of objects is arguably the most diverse in the literature to date: TossingBot [55] has a collection of 80+ different objects, including toy blocks, fake fruit, decorative items, and office objects, but it lacks deformable objects; the work by Monastirsky et al. [37] has 7 objects, including 2 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 15cm×15cm and the outer large box with dimensions 37cm×33cm. This design emulates the concept of Top-1 classification accuracy and Top-5 classification accuracy used in ImageNet [15]. 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 experiment results are presented in Table 6, demonstrating the significant improvement in throwing accuracy achieved through tube acceleration for all 5 throwing configurations.

Compared to the two previous works on planar throwing of different objects [55, 37], our small box measures 15cm×15cm, aligning with the target dimensions in Monastirsky et al. [37] and more compact than the 15cm×25cm box in TossingBot [55]. 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 [55]. While Monastirsky et al. [37] 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.



(a) (b)



(c) (d)

Figure 5: 4 dexterous throwing configurations in the qualitative experiment.

Moreover, certain throwing configurations, such as the one depicted in Fig. 5b, may exhibit more variability in release uncertainty, resulting in the lowest throwing accuracy among the 5 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.

2.1.4 Limitation of Tube Acceleration for Off-CoM Grasps

In the robust throwing formulation, the recursive task-validity problem is based on the assumption that the intricate release dynamics can be effectively approximated 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 the previous subsection 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

(a) 18 thrown objects in the qualitative experiment.

(b) Target box with two levels of error tolerance. Inner small box: 15cm × 15cm. Outer large box: 37cm × 33cm.

Figure 6: (a) Thrown objects and (b) target box used in the qualitative experiment.

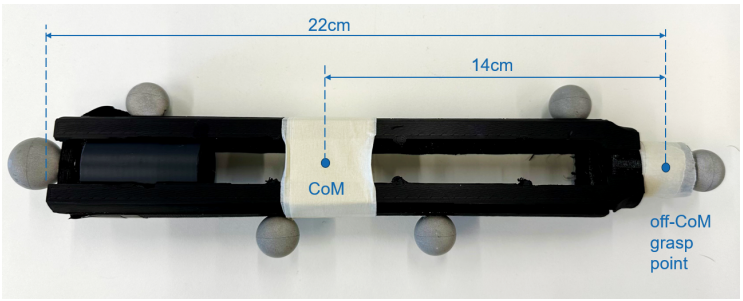


Figure 7: 3D-printed bar to study the effect of grasp CoM offset on throwing. The two designed grasp points, the CoM and one end, are covered with paper tape to maintain consistent contact properties.

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 3D-printed an object with known mass distribution, as shown in Fig. 7, 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. 8.

As shown in Fig. 8a, when grasped at CoM, the nominal landing position is 1.21m, and the landing position of the object CoM is almost identical to the nominal landing position. However, when grasped with a 0.14m CoM offset, the object lands 0.3m further away from the model prediction (1.67m vs. 1.37m). 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. 9). This finding confirms that our kinematic release model is not just quantitatively, but also qualitatively, inaccurate for off-CoM grasps.

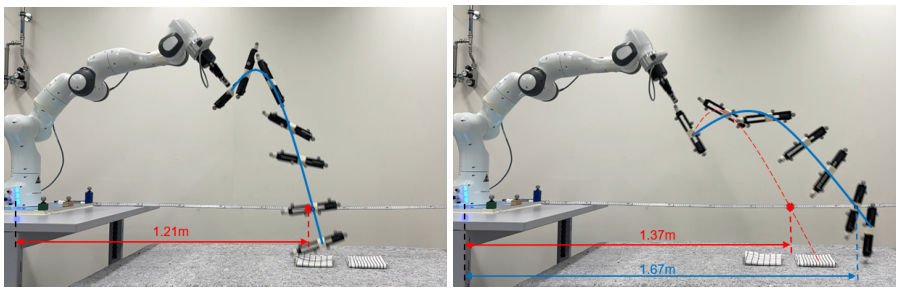
Object ID	Object name	Mass (g)	Dimension (mm)	Surface Material
1	plastic_ball	92	radius 30	plastic
2	fake_peach	34	radius 30	plastic
3	tennis_ball_hard	58	radius 35	nylon
4	tennis_ball_soft	46	radius 40	nylon
5	squeeze_ball_blue	18	radius 35	foam
6	squash_ball_red	27	radius 30	rubber
7	cardboard_box	135	80×60×60	cardboard
8	carton_box	99	150×65×28	carton
9	cleaning_sponge	8	95×65×40	foam
10	small_towel	42	110×75×32	textile
11	medium_towel	90	140×100×50	textile
12	empty_jar	15	80×50×50	plastic
13	whiteboard_pen	18	138×20×20	plastic
14	folded_tshirt	130	210×90×55	textile
15	fake_banana	68	190×38×32	plastic
16	plush_mole	51	160×70×55	textile
17	wrapped_rubber_pump	163	220×200×80	thick plastic bag
18	wrapped_foam_tape	32	160×140×60	thin plastic bag

Table 5: Summuary of object properties in the quantitative experiment.

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

Table 6: Summary of the quantitative throwing experiments. The box positions are expressed in the base frame of the robot.

.



(a) Accurate throw with CoM grasp **(b)** Large throwing error with CoM offset

Figure 8: Comparison of throws with different grasps.

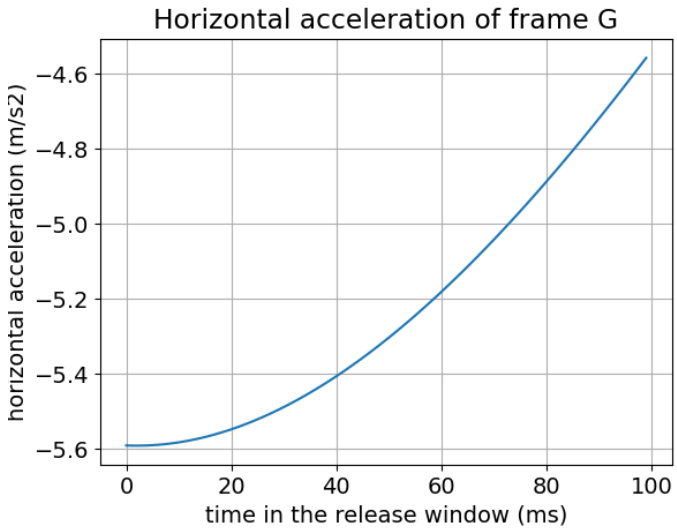


Figure 9: The horizontal acceleration of the virtual frame G is negative throughout the 100ms release window.

2.2 Physical modeling of transient release dynamics

In the previous subsection, we demonstrate a highly performant method for robust dexterous throwing of various objects for CoM grasps. However, as object-in-hand pose upon grasping can be arbitrary on the object body, either due to perception and control uncertainties during grasping action, or planned deliberately to ease the grasping from a cluttered source box, throwing with off-CoM grasps (eccentric throwing for brevity) have to be considered systematically for an automated throwing system.

In the robot throwing literature, only data-intensive end-to-end learning has been employed to handle eccentric throwing: TossingBot [55] learns an end-to-end mapping from grasping to throwing, leveraging visual observations (RGB-D images) to implicitly capture the relationship between grasping offsets from the CoM and throwing outcomes; TossNet [10] learns an autoregressive model to predict object landing poses using joint motion and wrist force/torque sensor measurements before the release, where the friction interaction is also learned implicitly. Despite their pioneering advancements, both TossingBot and TossNet suffer from inherent limitations of data-driven end-to-end approaches:

- **Lack of Physical Insights:** These models provide limited human understanding of the underlying physics of throwing, focusing instead on predictive accuracy through data.
- **Data-Intensive Training:** The reliance on thousands of samples for effective model training contrasts with the “zero-shot” nature of physical modeling.
- **Limited Transferability:** End-to-end methods face challenges in adapting to different robot embodiments or operational conditions. On the other hand, physical models, adhering to SI units, offer excellent transferability.

The challenge of eccentric throwing motivates our study on the physical modeling of the transient releasing dynamics that describes the momentum transfer from the robot to the object via the frictional patch.

We start with a conventional modeling strategy: combining rigid body dynamics with the common patch friction model *Limit Surface (LS)* [19, 23, 54], and identify that the velocity-magnitude-independent frictional wrench, described by the LS models, causes the robot-object dynamical system to be discontinuous, leading to pathological behavior (Zeno's phenomenon) [8] and limited model accuracy due to its sensitivity to uncertainty. In response, we propose *Sliding Pivot*, a physical surrogate model that offers smoother dynamics. Extensive robot throwing experiments are conducted to validate and assess the models' accuracies. Our model reduces horizontal velocity prediction error by 76% and angular velocity prediction error by 80%, achieving a mean absolute error (MAE) of 2.6 cm for landing position and 15 degrees for landing orientation, with significantly lower variability and systematic bias across a wide range of experimental conditions. To our knowledge, this work is the first study on the physical modeling of transient release dynamics in robot throwing and is currently under review as "*On Transient Release Dynamics in Robot Throwing: A Sliding Pivot Model," Liu and Billard, submitted to IEEE Transactions on Robotics, 2025* [31].

2.2.1 Parameters and variables in dynamic modeling of throwing

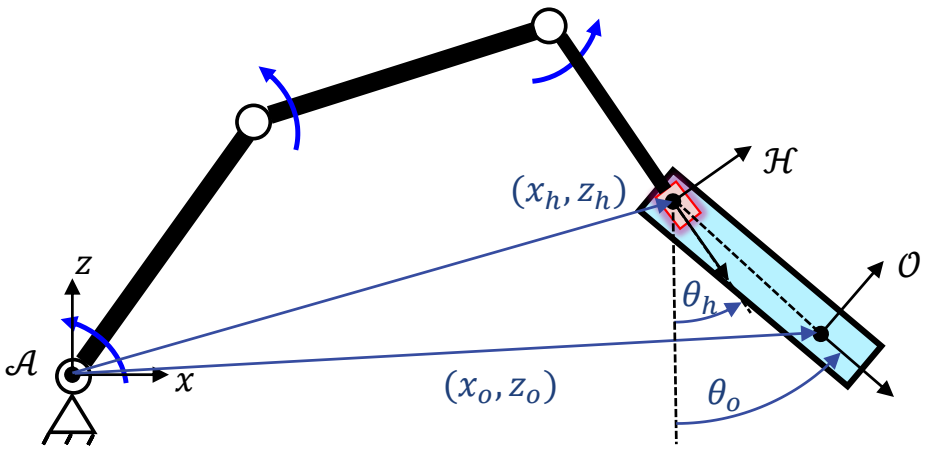


Figure 10: Major notations for release dynamics modeling.

As shown in Fig. 15, we use robot base frame $\mathscr{A}$ as the world frame. Hand frame $\mathscr{H}$ is located at the center of the two fingers. Object frame $\mathscr{O}$ is located at its center of mass (CoM). The generalized coordinates of the hand are denoted by,

$$\mathbf{q}^h = [x^h, z^h, \theta^h]^\top \in \mathbb{R}^3 \tag{1}$$

where x^h is the horizontal position of the hand frame $\mathscr{H}$ relative to the robot base frame $\mathscr{A}$, z^h is the vertical position of the hand frame $\mathscr{H}$ relative to the robot base frame $\mathscr{A}$, θ^h is the hand's orientation relative to the vertical orientation. Twist, acceleration, and applied wrench at the center of the two fingers are expressed in the world frame $\mathscr{A}$ and

are denoted by,

$$\mathbf{v}^h = [v_x^h, v_z^h, \omega^h]^\top \in \mathbb{R}^3 \tag{2}$$
$$\mathbf{a}^o = [a_x^h, a_z^h, \alpha^h]^\top \in \mathbb{R}^3 \tag{3}$$
$$\mathbf{f}^o = [f_x^h, f_z^h, \tau^h]^\top \in \mathbb{R}^3 \tag{4}$$

The generalized coordinates of the object are denoted by,

$$\mathbf{q}^o = [x^o, z^o, \theta^o]^\top \in \mathbb{R}^3 \tag{5}$$

where x^o is the horizontal position of its CoM relative to the robot base frame $\mathscr{A}$, z^o is the vertical position of its CoM relative to the robot base frame $\mathscr{A}$, θ^o is the object's orientation relative to the vertical orientation. Note that θ^o is unwrapped from $[0, 2\pi]$ in order to differentiate different number of flips during its free flying. Likewise, twist, acceleration, and applied wrench at CoM are expressed in the world frame $\mathscr{A}$ and are denoted by,

$$\mathbf{v}^o = [v_x^o, v_z^o, \omega^o]^\top \in \mathbb{R}^3 \tag{6}$$
$$\mathbf{a}^o = [a_x^o, a_z^o, \alpha^o]^\top \in \mathbb{R}^3 \tag{7}$$
$$\mathbf{f}^o = [f_x^o, f_z^o, \tau^o]^\top \in \mathbb{R}^3 \tag{8}$$

Define the relative coordinates of object frame $\mathscr{O}$ w.r.t. hand frame $\mathscr{H}$,

$$\mathbf{q}^r = [x^r, z^r, \theta^r]^\top \tag{9}$$
$$:= \mathbf{q}^o - \mathbf{q}^h \in \mathbb{R}^3 \tag{10}$$

For the contact point $\mathscr{C}$ on the object specified by a relative vector $\mathbf{q}^r$, the linear velocity of $\mathscr{C}$ expressed in robot base frame $\mathscr{A}$, denoted as $\mathbf{v}_{xz}^c$, is

$$\mathbf{v}_{xz}^c = \mathbf{v}_{xz}^o + w_o \times -\mathbf{q}_{xz}^r = \begin{bmatrix} v_x^o + w^o z^r \\ v_z^o - w^o x^r \end{bmatrix}, \tag{11}$$

where $\mathbf{v}_{xz}^o$ is the linear velocity $\mathscr{C}$ expressed in robot base frame $\mathscr{A}$. Hence, we can define the transformation matrix $\mathbf{G}(\mathbf{q}^r)$ that relates the contact point twist $\mathbf{v}^c$ and object twist $\mathbf{v}^o$:

$$\mathbf{v}^c = \begin{bmatrix} v_x^c \\ v_z^c \\ w^c \end{bmatrix} = \begin{bmatrix} v_x^o + w^o z^r \\ v_z^o - w^o x^r \\ w^o \end{bmatrix} = \mathbf{G}(\mathbf{q}_r)\mathbf{v}^o \tag{12}$$

with

$$\mathbf{G}(\mathbf{q}_r) = \begin{bmatrix} 1 & 0 & z^r \\ 0 & 1 & -x^r \\ 0 & 0 & 1 \end{bmatrix}, \tag{13}$$

Similarly, the wrenches are related via,

$$\mathbf{f}^o = \mathbf{G}^\top(\mathbf{q}^r)\mathbf{f}^c \tag{14}$$

The relative velocity $\mathbf{v}^r$ is defined as the difference between the velocity of the *contact point* $\mathbf{v}^c$ and the velocity of the hand $\mathbf{v}^h$,

$$\mathbf{v}^r := \mathbf{v}^c - \mathbf{v}^h \tag{15}$$

The Cartesian acceleration of the contact patch on the object $\mathbf{a}^c$ can be written as,

$$\mathbf{a}^c = \begin{bmatrix} a_x^c \\ a_z^c \\ \alpha^c \end{bmatrix} = \mathbf{a}^o + \begin{bmatrix} -\alpha^o z^r - \omega^{o2} x^r \\ \alpha^o x^r - \omega^{o2} z^r \\ 0 \end{bmatrix} \tag{16}$$

Then define the relative acceleration $\mathbf{a}_r$ as the *slip acceleration*,

$$\mathbf{a}_r := \mathbf{a}_c - \mathbf{a}_h \tag{17}$$

The rigid body dynamics expressed in the object frame is,

$$\mathbf{M}\mathbf{a}^o = \mathbf{G}^\top(\mathbf{q}^r)\mathbf{f}^c + \mathbf{g} \tag{18}$$

where $\mathbf{M} = \text{diag}(m, m, I)$ denotes the mass matrix of the object, $\mathbf{g} = [0, -g, 0]^T$ represents the gravitational wrench acting on the object in the vertical plane, and $\mathbf{f}^c$ is the frictional wrench applied on the object.

The finger normal force is defined as $f_N \in \mathbb{R}^+$. The tangential friction force limit is $f_{\max} = \mu f_N$ and the torsional friction torque limit is $\tau_{\max} = ca\mu f_N$, where μ is the contact patch's friction coefficient, a is the radius of the contact patch, and the constant $c \in [0, 1]$ represents the factor accounting for the contact geometry under uniform pressure distribution.

2.2.2 Sliding Pivot model

1) Sticking dynamics
During sticking, finger acceleration $\mathbf{a}^h$ equals the acceleration of the contact patch on the object $\mathbf{a}^c$, i.e. relative acceleration $\mathbf{a}^r$ is zero. Thus, we obtain the object acceleration $\mathbf{a}^o$ by setting $\mathbf{a}^c = \mathbf{a}^h$ in Equation 16:

$$\mathbf{a}^o = \begin{bmatrix} a_x^o \\ a_z^o \\ \alpha^o \end{bmatrix} = \mathbf{a}^h - \begin{bmatrix} -\alpha^o z^r - \omega^{o2} x^r \\ \alpha^o x^r - \omega^{o2} z^r \\ 0 \end{bmatrix} \tag{19}$$

and the *required* friction wrench $\mathbf{f}^p$ to achieve this acceleration to remain sticking:

$$\mathbf{f}^p = \mathbf{G}^\top(-\mathbf{q}^r)(\mathbf{M}\mathbf{a}^o - \mathbf{g}) \tag{20}$$

If $|\tau^p| \leq \tau_{\max}$ and $\|[f_x^p, f_z^p]\| \leq f_{\max}$, the object is sticking to the hand and its motion can be determined by Equation 19, i.e. propagating hand motion with the object be the extended body.
2) Pivoting dynamics
If $|\tau^p| > \tau_{\max}$ and $\|[f_x^p, f_z^p]\| \leq f_{\max}$, then the friction patch cannot generate enough torsional friction to resist spinning but can generate enough linear friction to prevent the contact patch from sliding. In this case, the system becomes a 1-DoF pivot around the finger pad. This pivoting can be conveniently described by the state of relative rotation $(\theta^r, \omega^r) = (\theta^o - \theta^h, \omega^o - \omega^h)$, with pivot acceleration α^r be,

$$\alpha^r := \dot{\omega}^r = \frac{\text{sgn}(|\tau^p|)(|\tau^p| - \tau_{\max}(f_N))}{ml^2 + I} \tag{21}$$

where $l = \|[x^r, z^r]\| = \left\|[x^o - x^h, z^o - z^h]\right\|$ be the CoM offset, $\tau_{\max}(f_N)$ be the time-varying torsional friction limit. In other words, the Karnopp routine is applied during

pivoting. Note that sticking dynamics and pivoting dynamics can be written in the following compact form:

$$\alpha^r = \dot{\omega}^r = \frac{\text{sgn}(|\tau^p|)(\max\{|\tau^p|, \tau_{\max}\} - \tau_{\max}(f_N))}{ml^2 + I} \tag{22}$$

3) Sliding dynamics
If $\|[f_x^p, f_z^p]\| > f_{\max}$, the pivot starts sliding on the contact surface, The sliding dynamics is given by,

$$\begin{bmatrix} a_x^r \\ a_z^r \end{bmatrix} = \frac{1}{m}\left(\begin{bmatrix} f_x^p \frac{f_{\max}}{\|[f_x^p, f_z^p]\|} \\ f_z^p \frac{f_{\max}}{\|[f_x^p, f_z^p]\|} \end{bmatrix} + \mathbf{g}_{xz}\right) \tag{23}$$

where $[a_x^r, a_z^r]^\top$ is the sliding acceleration, $\mathbf{g}_{xz} = [0, -g]^\top$. Pivot acceleration is also influenced by the decreased lever torque due to the insufficient friction force to maintain pivoting. Mathematically, define α^s be the amount of decreased pivot acceleration, given by,

$$\alpha_s = \frac{\begin{bmatrix} -x^r \\ -z^r \end{bmatrix} \times \begin{bmatrix} f_x^p(1 - \frac{f_{\max}}{\|[f_x^p, f_z^p]\|}) \\ f_z^p(1 - \frac{f_{\max}}{\|[f_x^p, f_z^p]\|}) \end{bmatrix}}{ml^2 + I} \tag{24}$$

As a result, the pivot acceleration during sliding is:

$$\alpha^r = \dot{\omega}^r = \frac{\text{sgn}(|\tau^p|)(\max\{|\tau^p|, \tau_{\max}\} - \tau_{\max}(f_N))}{ml^2 + I} - \alpha^s \tag{25}$$

2.2.3 Release dynamics model validation

Hardware setup
The throwing experiments are conducted by a 7 degrees of freedom fully actuated manipulator (Franka Emika Panda) mounted with Robotiq 2F-85 parallel gripper. To measure the vanishing normal force during gripper opening, an ATI Nano 17 F/T sensor is attached behind one of the gripper's finger pads. The experiment hardware setup is illustrated in Fig. 11. The thrown object is the 3D-printed bar with a known and configurable mass distribution attached with markers, shown in Fig. 7.
Batch throwing experiments
We conduct batch throwing experiments that cover a large space of landing poses and summarize the predictions of the conventional model LS and the proposed model SP. The throwing conditions are configured as follows:

- **2 CoM offsets**: a metal cylinder payload (mass: 154 gram) is configured at **2** different slots in the 3D-printed bar (mass: 92 gram), yielding two distinct CoM offsets from the tip ('CoM-1' - 10 cm, 'CoM-2' - 16 cm).
- **3 throwing pitch angles**: 3 different nominal throwing postures with distinct pitch angles (θ^h upon release).
- **6 release acceleration**: 6 different release motions ranging from acceleration to deceleration.

Each **(CoM, pitch angle, release acceleration)** tuple is thrown **5** times, resulting in **180** throws. The collection of landing poses is shown in Fig. 12, demonstrating that the experiment design covers a wide range of landing poses.
Result and discussion
The landing pose prediction errors, defined as the difference between the predicted and

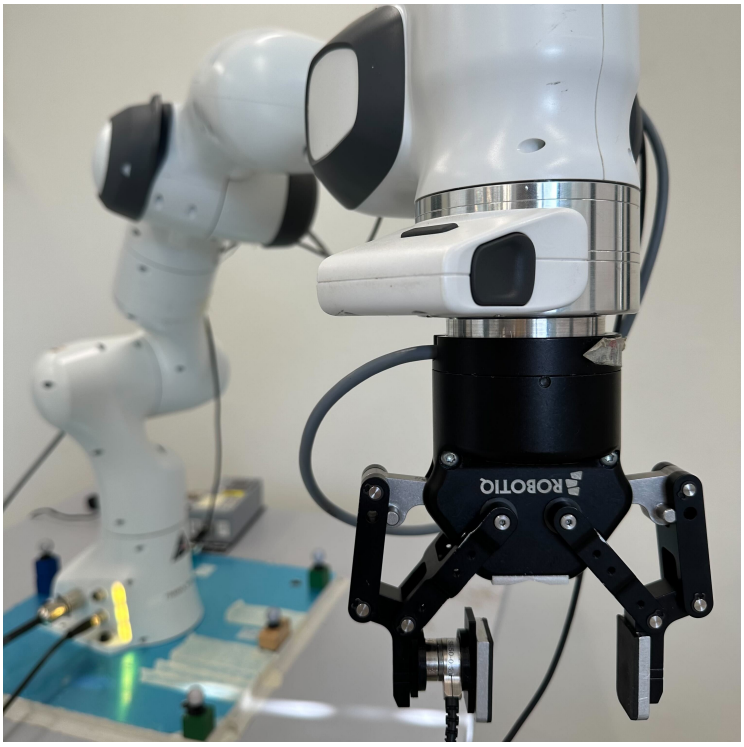


Figure 11: Hardware setup: Franka Emika Panda manipulator with Robotiq 2F-85 gripper. An ATI Nano 17 F/T sensor is mounted behind one finger pad.

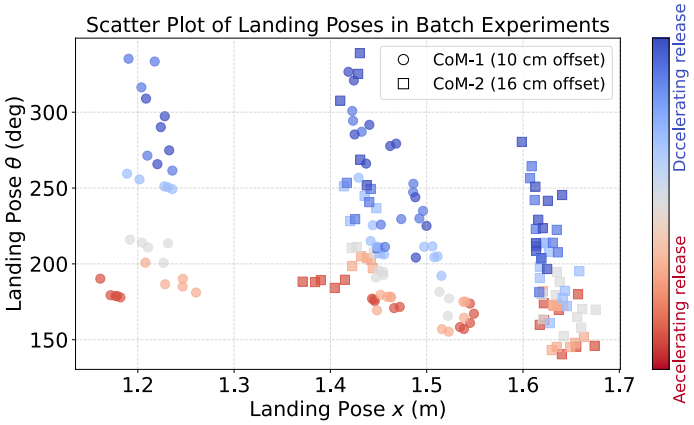


Figure 12: Scatter plot of landing poses in batch experiments. The marker shape indicates the two bar CoM configurations tested. The color bar represents the family of release motions, ranging from acceleration to deceleration. To avoid clutter, the three pitch angles are not explicitly labeled in the scatter plot but can be inferred from the three vertical clusters of landing poses.

actual landing poses, are shown in Fig. 13. The LS predictions exhibit a systematic bias, tending toward smaller horizontal displacements (negative error in landing x) and smaller rotations (negative error in landing θ). We hypothesize that this bias arises from frequent oscillations dissipating the energy transferred to the object. In contrast, the SP predictions

show no discernible trend or bias, indicating the absence of systematic error. Moreover, SP prediction errors are significantly more concentrated around zero compared to LS, highlighting the superior accuracy and consistency of SP in predicting landing poses.

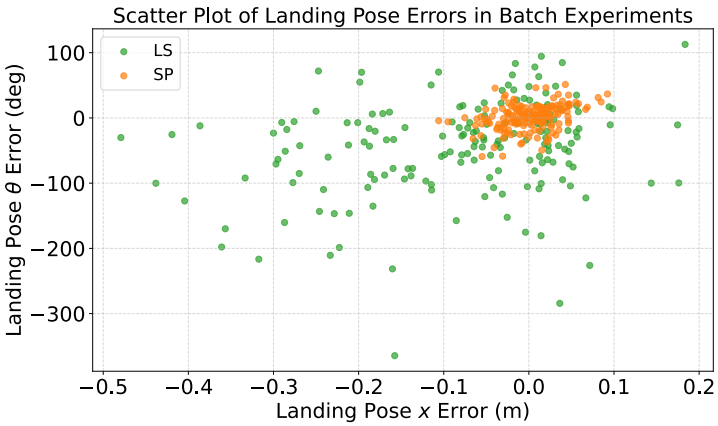


Figure 13: Scatter plot comparing the landing pose errors of LS and SP predictions on the 180 throws. The SP error population is much more condensed around zero compared to LS.

Table 7: MAE and STD of LS and SP predictions for free flying and landing metrics. Errors are MAE(±STD).

Model	Free Flying Twist		Landing Pose	
	$\dot{x}$ (m/s)	ω (deg/s)	x (m)	θ (deg)
LS	0.192(±0.159)	112.30(±101.72)	0.110(±0.105)	62.47(±57.74)
SP	**0.045(±0.030)**	**22.13(±20.27)**	**0.026(±0.020)**	**14.77(±12.34)**

The quantitative results of the landing pose prediction errors are summarized in Table 7, which compares the Mean Absolute Error (MAE) and Standard Deviation (STD) of the absolute error of the predictions from the conventional LS model and the proposed SP model across multiple metrics. The metrics include free-flying twist variables: horizontal velocity ($\dot{x}$) and angular velocity (ω), and landing pose variables: horizontal landing position (x) and landing orientation (θ).

The table highlights a significant improvement in prediction accuracy and consistency achieved by the SP model. Specifically, for free-flying twist predictions, SP demonstrates over a fourfold reduction in MAE for horizontal velocity ($\dot{x}$) and angular velocity (ω), with MAEs of 0.045 m/s and 22.13 deg/s, respectively, compared to 0.192 m/s and 112.30 deg/s for LS. Similarly, SP outperforms LS in landing pose prediction, achieving MAEs of 0.026 m and 14.77 deg for horizontal displacement (x) and orientation (θ), respectively, compared to LS's larger errors of 0.110 m and 62.47 deg.

In addition to achieving lower MAE, SP predictions also exhibit significantly smaller STD across all metrics, indicating a higher level of consistency in its predictions. This is particularly evident in the angular velocity (ω) and orientation (θ), where LS predictions show high variability (STD of 101.72 deg/s and 57.74 deg, respectively), whereas SP predictions are much more concentrated (STD of 20.27 deg/s and 12.34 deg). These results align with the visual analysis in Fig. 13, further demonstrating that the SP model not only eliminates the systematic biases observed in LS but also delivers robust and

reliable predictions across diverse experimental conditions.

2.2.4 Discussion on Impact on Robot Throwing

The proposed physical model on transient release dynamics opens up new possibilities for scalable and industrial-grade robot throwing.

Fast and accurate sorting. As shown in Fig.13, the throwing displacement error is bounded within ±11 cm across a wide range of throwing configurations and landing poses. This provides a direct comparison with TossingBot[55], an end-to-end learning model that achieves 85% success rate in throwing various objects with a 25 cm tolerance, but at the cost of tens of thousands of real throwing trials. In contrast, our physical model has the potential to enable precise throwing planning with over 99% success rate, given stock-keeping unit (SKU) information and accurate robot motion. This represents a significant leap in the technology readiness level (TRL) of robot throwing.

Throwing with desired landing pose. Additionally, Fig. 13 demonstrates that the landing orientation error is bounded within ±50 degrees, indicating that it is possible to plan throwing motions to achieve desired landing poses (position and orientation) with high precision. This capability can facilitate bar code scanning, optimize space utilization, and improve ergonomic access for human operators or other robots in industrial automation.

Knowledge transfer among fleets of throwing robots. Compared to end-to-end learning models, a key advantage of physical models is their excellent transferability. Object SKU information, sensor measurements, and robot motion are all described in the common language of SI units, allowing knowledge to be easily transferred across different embodiments and operating conditions. On the other hand, it is unclear how information can be transferred between end-to-end learning models trained for individual robots under varying conditions.

Physics-guided active learning. Finally, recent studies in operations research [46, 53] have shown that algorithms leveraging structural information achieve better exploration-exploitation trade-offs than classical structure-free methods in online decision-making. Therefore, for robot throwing in unstructured environments where object information is unavailable, the algebraic structure encoded in the physical model has the strong potential to guide strategic active learning and reduce sample complexity.

2.3 Learning to throw-flip with desired landing pose

Recent advances in robot throwing from EPFL [33, 30] as well as from the literature [55, 6, 25, 56] have shown that robots can throw a variety of objects accurately. However, attention has so far been given solely to having the object land precisely in a given location. Precise control over the final orientation of objects when thrown, however, is crucial—for instance, to enable accurate barcode scanning.

In this third contribution from EPFL, we explore controlled landing poses in the challenging task of throw-flipping a bar into a narrow box with a desired orientation (referred to as "throw-flip" hereafter for brevity). Throw-flipping is arguably one of the most complex dynamic manipulation tasks due to two key challenges:

1) Coupled displacement and rotation: For manipulators with revolute joints, increasing the linear velocity of the end-effector (EEF) to reach farther targets inherently increases its angular velocity, introducing a parasitic effect of exaggerated rotation. Thus, it is essential to design robot throwing actions that can independently steer both landing position and orientation.

2) Intricate release dynamics: The in-hand sliding and spinning that occur during the transient phase (approximately 50 ms) of vanishing gripper normal force are difficult to

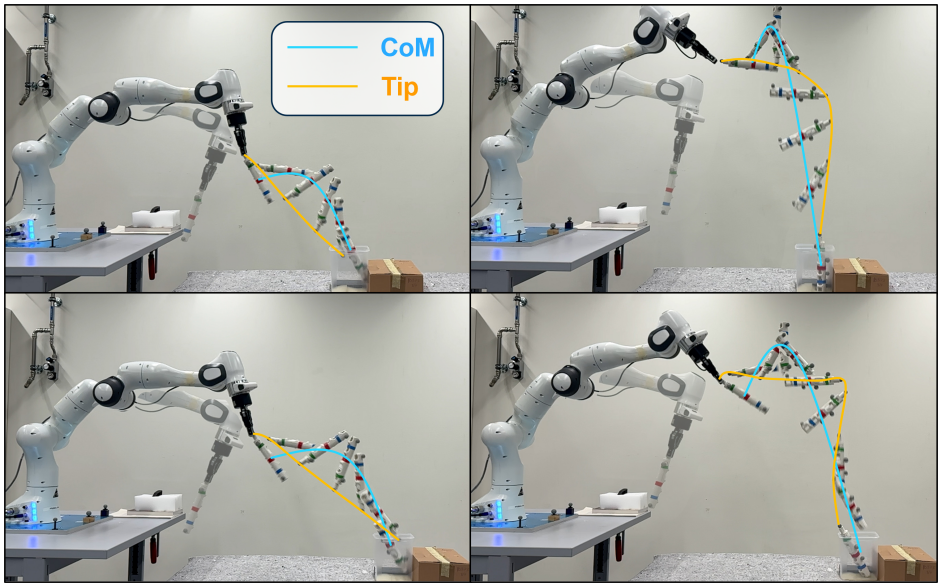


Figure 14: Robot throw-flips the bar with 4 different landing poses.

model. Additionally, some physical parameters, such as friction and deformations induced by a tight hold, are difficult to measure accurately.

In the literature, robot throwing with desired *landing poses (position and orientation)* has received much less attention, with only a few notable exceptions:

1) Non-prehensile throwing with desired pose: In non-prehensile throwing, objects are held immobile via inertial forces during acceleration and are released instantaneously by maximizing deceleration of the hand to minimize the effects of friction during release, resulting in accurate throws. Most early works on robot throwing considered non-prehensile throwing [2, 48, 1, 35, 34]. [34] applied sequential quadratic programming (SQP) for motion generation to throw a block with a flat pad to achieve a desired landing pose. More recently, [44] adapted a similar setup in [34] to achieve highly accurate throwing at high speed by utilizing the set of valid release states that end up with successful throws. However, in these setups, the instantaneous release from the "palm" inherently binds the object's feasible free-flying motion to the motion capabilities of the end-effector. Consequently, for typical robot manipulators with revolute joints, the tight coupling between linear and angular motion at the end-effector severely restricts the range of achievable landing poses for the thrown object.

2) Prehensile throwing with desired pose: Prehensile throwing, where the object is firmly grasped to prevent slippage during acceleration, poses additional challenges for achieving accurate throws due to release uncertainties arising from dynamic friction and deformations. By leveraging a large amount of real throwing data (on the scale of thousands) to implicitly encapsulate all sources of uncertainty, including transient friction, TossNet [10] learns an autoregressive model to predict object landing poses from joint motion and wrist force/torque sensor measurements prior to the release. The trained model is then used to determine robot motions for accurate throws through a bisection method, including "pitching" bottles to desired poses. However, as demonstrated in the work, the obtained "pitching" motion only involves linear displacement without rotation (either for the end-effector or the flying object), implying that the landing orientation can only align with the end-effector's orientation at the moment of detach. Notably, TossNet also evaluated the accuracy of a benchmark physics model (rigid-body and projectile dynamics,

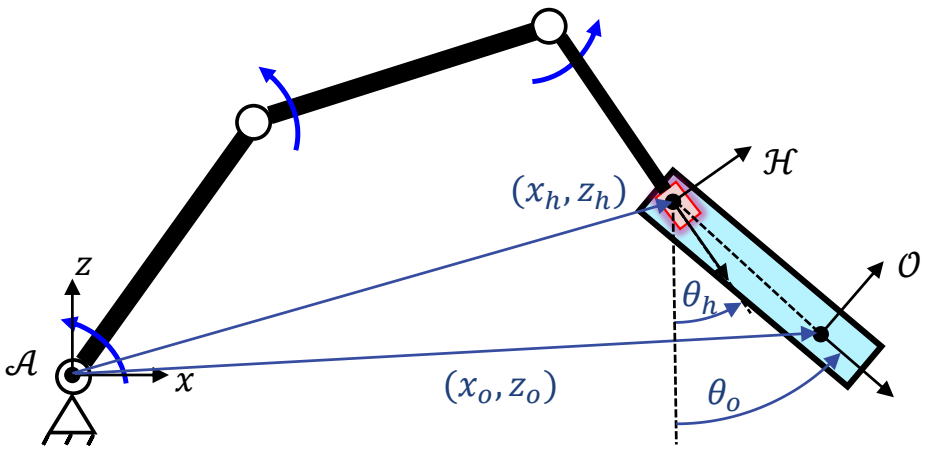


Figure 15: Major notations for flipping.

neglecting frictional interactions), which showed significantly larger errors compared to the learned model (∼3 cm vs. ∼15 cm error for throws with ∼50 cm horizontal displacement).

Compared to the literature that suffers from restricted landing poses, we aim to throw-flip objects to precise landing poses within a large and dense set of reachable outcomes. In addition, facing the dilemma between the inaccuracies of model-based planning and the high sample complexities of end-to-end learning, we aim to strike a balance by designing a learning system that seamlessly assimilates data with physical knowledge to accelerate learning. More specifically, our contributions include,

- **Control system design with impulse-momentum principle** to decouple displacement and rotation, resulting in drastically enlarged feasible landing poses.
- **Learning with data assimilation** integrates empirical data with flying dynamics, reducing sample complexity by 40% compared to pure data-driven learning.
- **Transfer learning to throw-flip a new object:** Reusing past data on in-hand object spinning when throw-flipping a new object under a Center of Mass (CoM) shift reduces sample complexity by 70%, compared to CoM-shift-agnostic methods.

This work is under review as "*Learning to Throw-Flip,*" *Liu, Da Costa, and Billard, submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2025* [32].

2.3.1 Method

Modeling throw-flipping A schematic of the throw-flip variables is illustrated in Fig. 15. The robot base frame $\mathscr{A}$ serves as world frame. Hand frame $\mathscr{H}$ is located at the center of the two fingers. Object frame $\mathscr{O}$ is located at its center of mass (CoM). In the absence of external disturbance, the throw-flip motion is planar. In the plane of motion, the coordinates of the hand frame w.r.t. the world frame reduce to three parameters: $\mathbf{q}^h = [x^h, z^h, \theta^h]^\top \in \mathbb{R}^3$, where x^h and z^h are the coordinates on the horizontal and vertical axes, and θ^h the orientation of the object at release time relative to the vertical axis. Hand twist at the center of the two fingers is expressed in the world frame $\mathscr{A}$: $\mathbf{v}^h = [v_x^h, v_z^h, \omega^h]^\top \in \mathbb{R}^3$.

Denote the joint position and velocity of an N-DoF robot manipulator as $\mathbf{q} \in \mathbb{R}^N$, $\dot{\mathbf{q}} \in \mathbb{R}^N$, respectively. Associate the end-effector (EEF) states in the world frame using standard forward kinematics and differential forward kinematics relationships:

$$\mathbf{q}^h = \mathbf{f}^h_{kin}(\mathbf{q}) : \mathbb{R}^N \to \mathbb{R}^3$$
$$\mathbf{v}^h = \mathbf{J}^h(\mathbf{q})\dot{\mathbf{q}} : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^3$$

The generalized coordinates of the object/bar are denoted by $\mathbf{q}^o = [x^o, z^o, \theta^o]^\top \in \mathbb{R}^3$. Note that θ^o is unwrapped from $[0, 2\pi]$ in order to differentiate different number of flips during its free flying. Likewise, object twist at CoM is expressed in the world frame $\mathscr{A}$ and is denoted by $\mathbf{v}^o = [v^o_x, v^o_z, \omega^o]^\top \in \mathbb{R}^3$.

For indoor throwing of solid objects, air drag can be neglected. The free-flying dynamics is only subject to gravitational force, resulting in the object acceleration $\mathbf{a}^o := [a^o_x, a^o_z, \alpha^o]^\top = [0, -G, 0]^\top$, where $G = 9.81$ m/s^2 is the gravitational acceleration. For such projectile dynamics, the landing pose can be obtained in closed form given the release state $(\mathbf{q}^o, \mathbf{v}^o)$. The flying duration t_{fly} can be calculated as $t^{fly} = \left(\dot{z}^o + \sqrt{\dot{z}^{o2} + 2G(z^o - z^{land})}\right)/G$. Then, $x^{land} = x^o + v^o_x t^{fly}$, $\theta^{land} = \theta^o + \omega^o_x t^{fly}$. Without loss of generality, in this work, we assume that the landing height is always 0 in the robot frame and define the landing pose at height 0 as $\mathbf{q}^{land} = (x^{land}, \theta^{land})$. We define the projectile flying flowmap g, with $\mathbf{q}^{land} = g(\mathbf{q}^o, \mathbf{v}^0)$, to compute the landing pose at 0 height given release state.

Define the relative coordinates of object frame $\mathscr{O}$ w.r.t. hand frame $\mathscr{H}$, $\mathbf{q}^r = [x^r, z^r, \theta^r]^\top := \mathbf{q}^o - \mathbf{q}^h \in \mathbb{R}^3$. Then for the contact point $\mathscr{C}$ on the object specified by a relative vector $\mathbf{q}^r$, the contact point twist $\mathbf{v}^c$ and object twist $\mathbf{v}^o$ are related through:

$$\mathbf{v}^c = \begin{bmatrix} v^c_x \\ v^c_z \\ w^c \end{bmatrix} = \begin{bmatrix} v^o_x + w^o z^r \\ v^o_z - w^o x^r \\ w^o \end{bmatrix} \tag{26}$$

The relative velocity $\mathbf{v}^r$ is defined as the difference between the velocity of the *contact point* $\mathbf{v}^c$ and the velocity of the hand $\mathbf{v}^h$, i.e. $\mathbf{v}^r := \mathbf{v}^c - \mathbf{v}^h$.

Impulse-momentum-based control design For manipulators with revolute joints, the coupling between the linear motion and the angular motion of the end-effectpr significantly limits the feasible space of the landing poses of thrown objects. To address this challange, we propose leveraging the temporal hinge effect during the gripper-opening window, as illustrated in the schematic of Fig. 16. The temporal hinge can be demonstrated through a simple dropping experiment with our fingers: pinch-grasp a pen at one end, hold it horizontally, and gradually open the fingers—the pen will first pivot in-hand and then detach. During the transient release window, the finger-bar contact behaves like an underactuated joint, making the robot-bar system an (N+1)-DoF pendubot for an N-DoF manipulator. In this scenario, if the robot undergoes drastic deceleration during the temporal hinge window, the impulse-momentum principle allows part of the robot's angular momentum to be transmitted to the bar through the frictional interface, accelerating the object's angular velocity. The amount of hinge acceleration can be controlled by the magnitude of braking during this window. As a result, we obtain one additional steerable DoF, independent of the nominal throwing state.

3-parameter family of flip motions As sketched in Section 2.3.1, the two-phase flip motion family proceeds as follows:

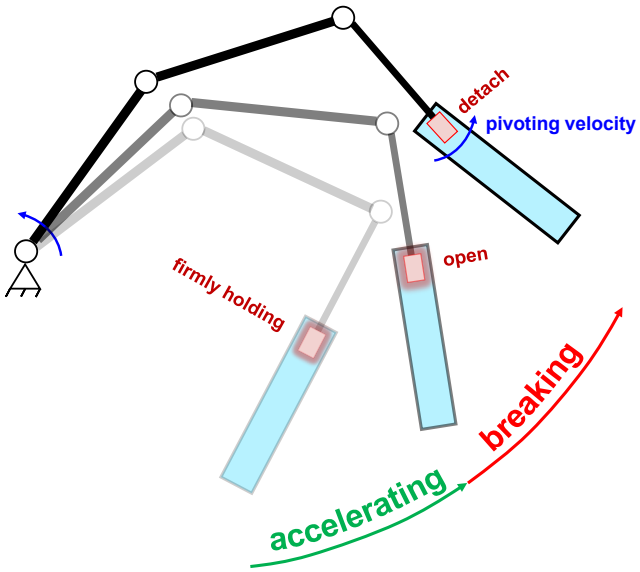


Figure 16: Schematic of flip motion. The robot firmly grasps the bar and accelerates it to a high-energy, high-velocity state, then enters a rapid decelerating-braking phase. During braking, the robot's gripper begins to open with decreasing normal force. Approximately 50 ms later, the normal force vanishes completely, and the bar enters free flight. Leveraging the impulse-momentum principle, the bar's rotation is accelerated by the pivoting velocity.

Acceleration Given a desired nominal throwing state in joint space $(\mathbf{q}_d, \dot{\mathbf{q}}_d) \in \mathbb{R}^{2N}$, a dynamically feasible accelerating trajectory is generated to connect the pre-throw pose with the nominal throwing state. This trajectory is executed via a standard impedance controller.

Starting from a fixed initial nominal throwing state $(\mathbf{q}_s, \dot{\mathbf{q}}_s) \in \mathbb{R}^{2N}$, ξ_d is obtained by adjusting two scalar parameters, "pitch" γ and "speed" s, as follows,

$$\mathbf{q}_d = \mathbf{q}_s + \frac{\gamma}{N} I_{N\times 1}, \quad \dot{\mathbf{q}}_d = s\dot{\mathbf{q}}_s$$

Effectively, "pitch" γ and "speed" s regulate the direction and the magnitude of the finger's linear velocity.

Brake After bypassing the end of the trajectory, the target position is set as $\mathbf{q}_d$ and the target velocity is set to $\mathbf{0}$ for the impedance controller. The magnitude of the brake is regulated by the "damping" D parameter of the impedance controller.

In summary, the family of robot flip motions is indexed by ("pitch" γ, "speed" s, "damping" D). Hence, we define the flip command $\mathbf{u} := (\gamma, s, D) \in \mathbb{R}^3$.

Learning and Adaptation We aim to solve the *inverse problem* of finding appropriate control commands for desired throwing outcomes through iterative learning. The rationale behind the data-driven approach is the absence of accurate *forward models* to describe the physical process of throw-flipping. This is not only due to the complex and intricate dynamic patch friction between the finger pad and the object, but also due to the lack of accurate robot dynamics models when operating in the highly dynamic regime of impulse-momentum braking.

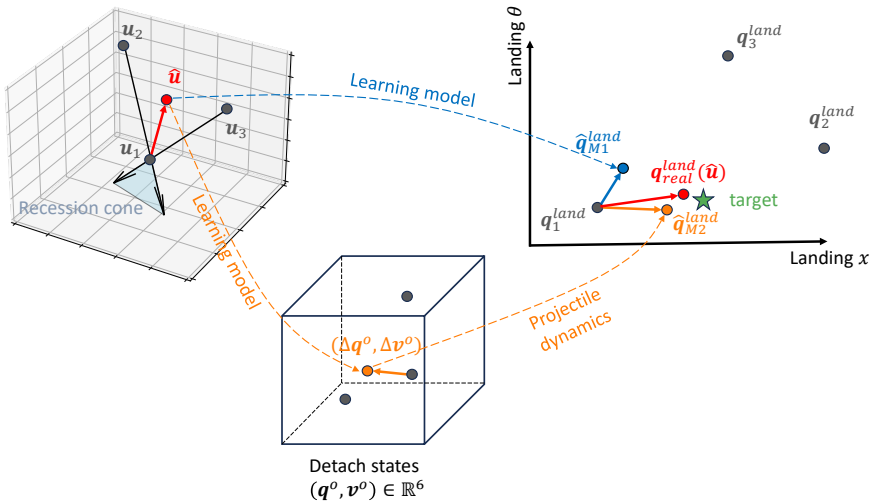


Figure 17: Schematic of the two forward models to predict the landing pose of a new command $\hat{\mathbf{u}}$.

Our learning to throw-flip setup is as follows: Given an initial support dataset $\mathcal{X}_0 = \{(\mathbf{u}, \mathbf{q}^h, \mathbf{q}^r, \mathbf{v}^h, \mathbf{v}^r, \mathbf{q}^o, \mathbf{v}^o, \mathbf{q}^{land})\}$ representing the initial observations of the physical process, the task is to iteratively refine control commands that progressively throw-flips the object closer to an unseen target landing pose as new data is gathered throughout the process. For a learning and adaptation system, two crucial design choices must be considered:

- **Forward model:** Given the dataset of past experiences, how to build a forward model to *predict* the throwing outcome of a new command?
- **Iterative command adaptation:** How to effectively adapt the command based on the updated forward model? As an autonomous decision-maker, what actions should be taken when the process gets stuck?

Next, we describe our iterative learning design in detail.

Forward Model Despite the absence of accurate release dynamics models and robot dynamics models, there is still rich geometrical and physical knowledge about the throw-fliping process at our disposal, as presented in Section 2.3.1. Can this knowledge be utilized to accelerate learning? To answer this question, we design and compare the following two models: as illustrated in Fig. 17,

- **Model 1: end-to-end** learns a locally linear map from the 3D command $\mathbf{u}$ to the 2D landing pose directly.
- **Model 2: projectile-based** learns a locally linear map from 3D command to the 6D object's detach state $(\mathbf{q}^o, \mathbf{v}^o)$ and then obtains the predicted landing pose using the projectile flying flowmap $g(\mathbf{q}^o, \mathbf{v}^o)$. Compared to **Model 1**, this model encodes physical knowledge of flying dynamics to assist learning.

Mathematically, we define the normalized error as,

$$e(\mathbf{q}^{\mathbf{land}}, \mathbf{q}^{\mathbf{target}}) = \left\| \left[\frac{x^{land} - x^{target}}{\epsilon_x}, \frac{\theta^{land} - \theta^{target}}{\epsilon_\theta} \right] \right\|_2$$

Algorithm 1: Iterative Learning with Cone Search

Input: Support dataset $\mathcal{X}$, Target pose $\mathbf{q}^{target}$, Max iterations T, Error thresholds $\epsilon_x, \epsilon_\theta$, #Command trials N

for $t = 1$ *to* T **do**
  Identify three nearest neighbors in $\mathcal{X}$: $(\mathbf{u}_i, \mathbf{q}_i^o, \mathbf{v}_i^o, \mathbf{q}_i^{land})$, $i \in \{1,2,3\}$;
  foreach (α_1, α_2) *in mesh grid* **do**
    Compute $\hat{\mathbf{q}}^{land}$ using Eq. 27 or Eq. 28 ;
    Compute error $e(\hat{\mathbf{q}}^{land}, \mathbf{q}^{target})$;
  Find best command $\mathbf{u}^t(\alpha_1, \alpha_2)$ minimizing e Execute $\mathbf{u}^t$, observe N trials, compute mean landing pose $\bar{\mathbf{q}}_t^{land}$
  Update dataset: $\mathcal{X} \leftarrow \mathcal{X} \cup (\mathbf{u}^t, \bar{\mathbf{q}}_t^{land})$
  if *All N trials land within thresholds* **then**
    return $\mathbf{u}_t$ (Success);
  if *learning stagnates (error does not improve)* **then**
    Select new neighbors to escape local minimum;
return best found $\mathbf{u}$;

where $\epsilon_x, \epsilon_\theta$ are the target pose thresholds for position and orientation, respectively. From a dataset $\mathcal{X}$, we identify the three nearest neighbors based on the normalized error, denoted as $(\mathbf{u}_i, \mathbf{q}_i^o, \mathbf{v}_i^o, \mathbf{q}_i^{land}), i \in 1,2,3$, ranked by their normalized error relative to the target pose. Define the delta command as $\Delta\mathbf{u}(\alpha_1, \alpha_2) = \sum_{i=1,2} \alpha_i(\mathbf{u}_{i+1} - \mathbf{u}_1)$, where $(\alpha_1, \alpha_2) \in \mathbb{R}^2$ is the recession cone coordinate we are searching for. Then the two *forward models* predict the landing poses of a new command $\mathbf{u} = \mathbf{u}_1 + \Delta\mathbf{u}(\alpha_1, \alpha_2)$ as follows:

$$\hat{\mathbf{q}}_{M1}^{land}(\alpha_1, \alpha_2) = \mathbf{q}_1^{land} + \sum_{i=1,2} \alpha_i(\mathbf{q}_{i+1}^{land} - \mathbf{q}_1^{land}) \tag{27}$$

$$\hat{\mathbf{q}}_{M2}^{land}(\alpha_1, \alpha_2) = g(\hat{\mathbf{q}}^o, \hat{\mathbf{v}}^o) = g(\mathbf{q}_1^o + \sum_{i=1,2} \alpha_i(\mathbf{q}_{i+1}^o - \mathbf{q}_1^o), \mathbf{v}_1^o + \sum_{i=1,2} \alpha_i(\mathbf{v}_{i+1}^o - \mathbf{v}_1^o)) \tag{28}$$

Model Inversion with On-Manifold Cone Search In essence, our iterative learning problem is a zeroth-order optimization/root-finding problem, where only sparse forward model evaluation is available. At each iteration, given the current best command ($\mathbf{u}_1$) for throw-flipping to the desired target, we aim to determine the delta change of the current best command ($\Delta\mathbf{u}$) to throw closer. We propose a fast adaptation method using on-manifold exhaustive search in the local recession cone. At each iteration, the robot samples a dense mesh of candidate feasible commands parametrized by the recession cone coordinate (α_1, α_2). It then selects the best candidate that minimizes the *predicted error* to execute and adds the new observations to the dataset. Progressively, the algorithm refines the throwing command using more accurate forward models around the target in the accumulated dataset. The pseudocode describing the procedure is summarized in Algorithm 1.

Transfer Learning under CoM shift To throw a new object with a different and known CoM, instead of gathering a support set from scratch and then starting iterative learning, we propose a method to transfer experience from throwing other objects to accelerate the learning process for the new object.

Assuming that the in-hand sliding-spinning twist remains unchanged after a CoM shift Δh, according to Eq. 26, we obtain the predicted landing pose of the new object by shifting the detach state $(\mathbf{q}^o, \mathbf{v}^o)$ as follows:

$$\hat{\mathbf{q}}^o(\Delta h) = \mathbf{q}^o + \Delta h[\sin\theta^o, -\cos\theta^o, 0]^\top$$
$$\hat{\mathbf{v}}^o(\Delta h) = \mathbf{v}^o + \omega^o \Delta h[\cos\theta^o, \sin\theta^o, 0]^\top$$

Integrating $(\mathbf{q}^o, \mathbf{v}^o)$ with projectile dynamics yields the transferred support set $\hat{\mathcal{X}}$. The learning procedure, denoted as **Model 3: transfer learning**, is as follows,

- **1st Iteration:** Select the closest command within the transferred support set $\hat{\mathcal{X}}$, denoted as $\mathbf{u}_1$. After executing $\mathbf{u}_1$, obtain data $(\mathbf{q}_1^o, \mathbf{v}_1^o, \mathbf{q}_1^{land})$.
- **2nd Iteration:** Select the 3 closest data entries within $\hat{\mathcal{X}}$, denoted as $\{(\hat{\mathbf{u}}_i, \hat{\mathbf{q}}_i^o, \hat{\mathbf{v}}_i^o, \hat{\mathbf{q}}_i^{land})\}, i \in 1, 2, 3$.Then we compute the best delta command $\Delta\mathbf{u}$, such that $\mathbf{u}_2 = \mathbf{u}_1 + \Delta\mathbf{u}$ yields closest landing pose $\hat{\mathbf{q}}_2^{land}$ predicted by the 3 selected data. In particular, let $\Delta\mathbf{u}(\alpha_1, \alpha_2) = \sum_{i=1,2} \alpha_i(\hat{\mathbf{u}}_{i+1} - \hat{\mathbf{u}}_1)$, then

$$\hat{\mathbf{q}}_2^{land} = g(\hat{\mathbf{q}}_2^o, \hat{\mathbf{v}}_2^o) = g(\mathbf{q}_1^o + \sum_{i=1,2} \alpha_i(\hat{\mathbf{q}}_{i+1}^o - \hat{\mathbf{q}}_1^o), \mathbf{v}_1^o + \sum_{i=1,2} \alpha_i(\hat{\mathbf{v}}_{i+1}^o - \hat{\mathbf{v}}_1^o))$$

- **3rd Iteration:** Select the 1st, 2nd and 4th closest data entries within $\hat{\mathcal{X}}$, compute the best delta command $\Delta\mathbf{u}$, such that $\mathbf{u}_3 = \mathbf{u}_2 + \Delta\mathbf{u}_2$ yields the closest predicted landing pose $\hat{\mathbf{q}}_3^{land}$, calculated in the same way as in the 2nd Iteration. $\hat{\mathbf{q}}_3^{land}$ is calculated the same way as in the 2nd iteration,
- **From the 4th Iteration Onward:** Start normal iterative learning using a new support set, consisting of the data obtained from the first three iterations.

2.3.2 Experiment

Hardware setup The throwing experiments are conducted using a 7-DOF Franka Emika Panda manipulator mounted with a Robotiq 2F-85 parallel gripper. The thrown object is a 3D-printed bar attached with markers, whose mass distribution can be configured by shifting the payload position within the bar.

Control design verification We first verify that our impulse-momentum-based control design can yield a large set of landing poses using only 3 control parameters (pitch" γ, speed" s, "damping" D), ensuring that this family of throwing motions includes commands capable of flipping the bar with large rotations.

To validate the control design, we conducted throwing experiments for a grid of $3^3 = 27$ control commands (3 pitch" γ, 3 speed" s, 3 "damping" D). Each command is executed 5 times to account for intrinsic randomness in the flipping process, arising from tiny variations in the grasp pose, stochastic friction between the finger and the bar [**?**], and the unrepeatable motor control of the Panda robot. A metal payload of 150 gram is configured in the second slot of the bar, making its CoM be 12 cm from the grasp point. The resulting population of $27 \times 5 = 135$ landing poses is illustrated in Fig. 18. By focusing on one parameter/channel at a time, we can observe the effect of individual control parameters:

- **"pitch"** γ**:** $\bigcirc \rightarrow \square \rightarrow \triangle$, increasing "pitch" steers the robot from a "forward" throw to an "upward" throw, resulting in a decreased landing distance, and an increased landing angle.

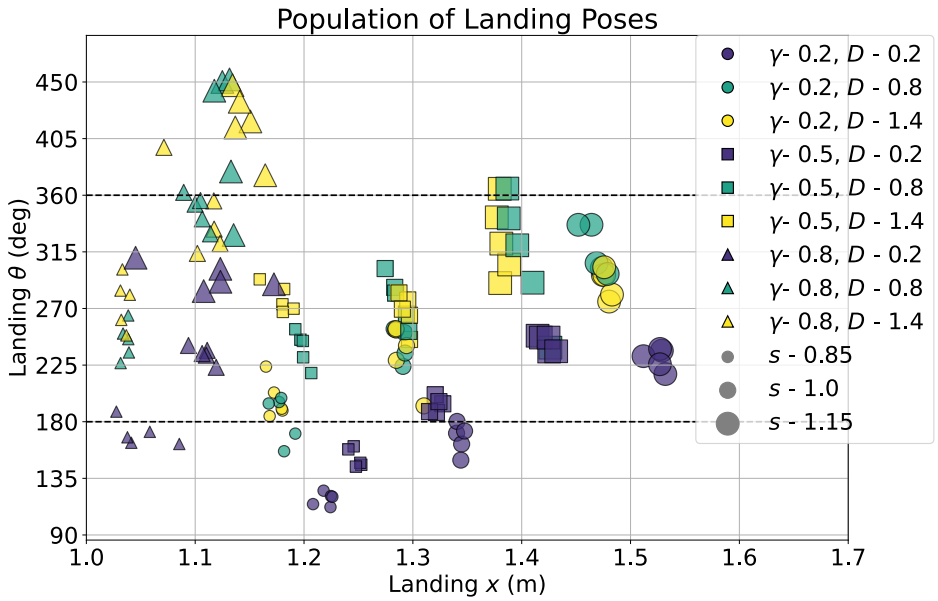


Figure 18: Initial population of landing poses generated from impulse-momentum-based control using a mesh of $3 \times 3 \times 3 = 27$ commands. Each command is executed 5 times, yielding 135 throws.

- **"speed" s:** ○ → ○ → ○, increasing the "speed" raises the linear velocity and also causes a parasitic increase in angular velocity, making the landing distance and landing angle increase simultaneously.
- **"damping" D:** (●, ■, ▲) purple landing poses obtained with the lowest damping are sparse and cannot achieve a full flip (360 degree), highlighting the limitations of purely velocity-based control. Introducing "damping" significantly expands the set of feasible landing poses. ● → ● → ●, increasing damping effectively increases the landing angle while slightly reducing landing distance.

Overall, our impulse-momentum-based control design significantly expands the reachable space of landing poses compared to the conventional velocity-based control design, notably enabling full 360-degree flips.

Learning to throw-flip: model comparison We report on a comparative analysis of throw-flip performance using end-to-end learning (model 1) as opposed to projectile-dynamics-based learning (model 2). Given the inherent stochasticity in the throwing process, as demonstrated by the variance of landing poses in Fig. 18, we set the target tolerance to (±5 cm, ±45 degrees). To evaluate the capability of the learning and adaptation system in throw-flipping to unseen poses, we provide only four initial commands that form a simplex in the 3D command space, offering a minimal non-degenerate representation of the forward model. Four unseen target landing poses are specified: (a) (1.2, 180), (b) (1.2, 360), (c) (1.4, 180), (d) (1.4, 360). The mean landing poses associated with the four commands in the support simplex, along with the target poses and their tolerances, are illustrated in Fig. 19. To account for the stochasticity in the throwing process, each command is executed three times per iteration to obtain the mean outcome. Each learning path is conducted for five iterations, resulting in 15 throwing trials per learning path.

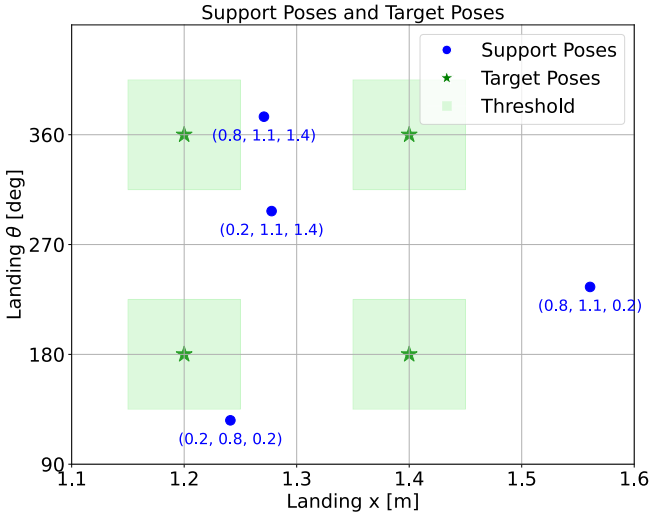


Figure 19: Mean landing poses for the four commands in the support simplex and the four unseen target poses.

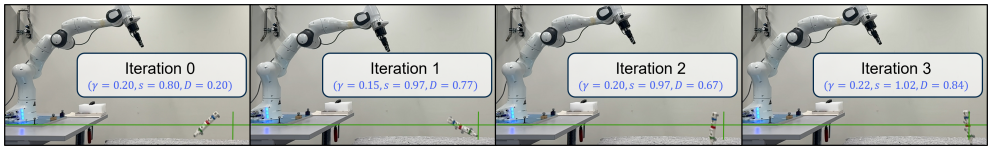


Figure 20: A sample learning process for target pose (1.4, 180) with Model 2. Iterative 0 represents the closest landing pose within the support set. The green vertical bar represents the desired landing pose.

Results are summarized in Table 8 and Fig. 21. A sample learning process for the target (1.4, 180) with Model 2 is illustrated in Fig. 20.

	Error	First Iter.		Min.		#Iters Enter Thres.	
Target	Initial	M1	M2	M1	M2	M1	M2
(a)	1.58	3.37	**1.04**	1.58	**0.45**	Failed	**1**
(b)	1.33	2.02	**1.94**	1.33	**0.83**	Failed	**5**
(c)	3.12	2.27	**2.25**	0.79	**0.52**	4	4
(d)	2.88	1.97	**1.24**	**0.40**	0.68	2	**1**
Average	2.23	2.41	**1.62**	1.02	**0.62**	>4.5	**2.75**

Table 8: Summary of normalized error and the number of iterations required to reach the target threshold for the four target poses. 'M1' - Model 1, 'M2' - Model 2.

Normalized errors: The normalized error for Model 2 after the first iteration is consistently lower than that of Model 1, indicating that incorporating physical knowledge of nonlinear projectile dynamics explains away part of the nonlinearity in the command-outcome map. On average, Model 2 achieves 40% smaller error than Model 1.
Iterative improvement: For Model 2, the iterative learning procedure progressively brings the landing pose closer to the target. This is evidenced by the fact that the minimum error is always achieved after the second iteration, despite some error oscillation.

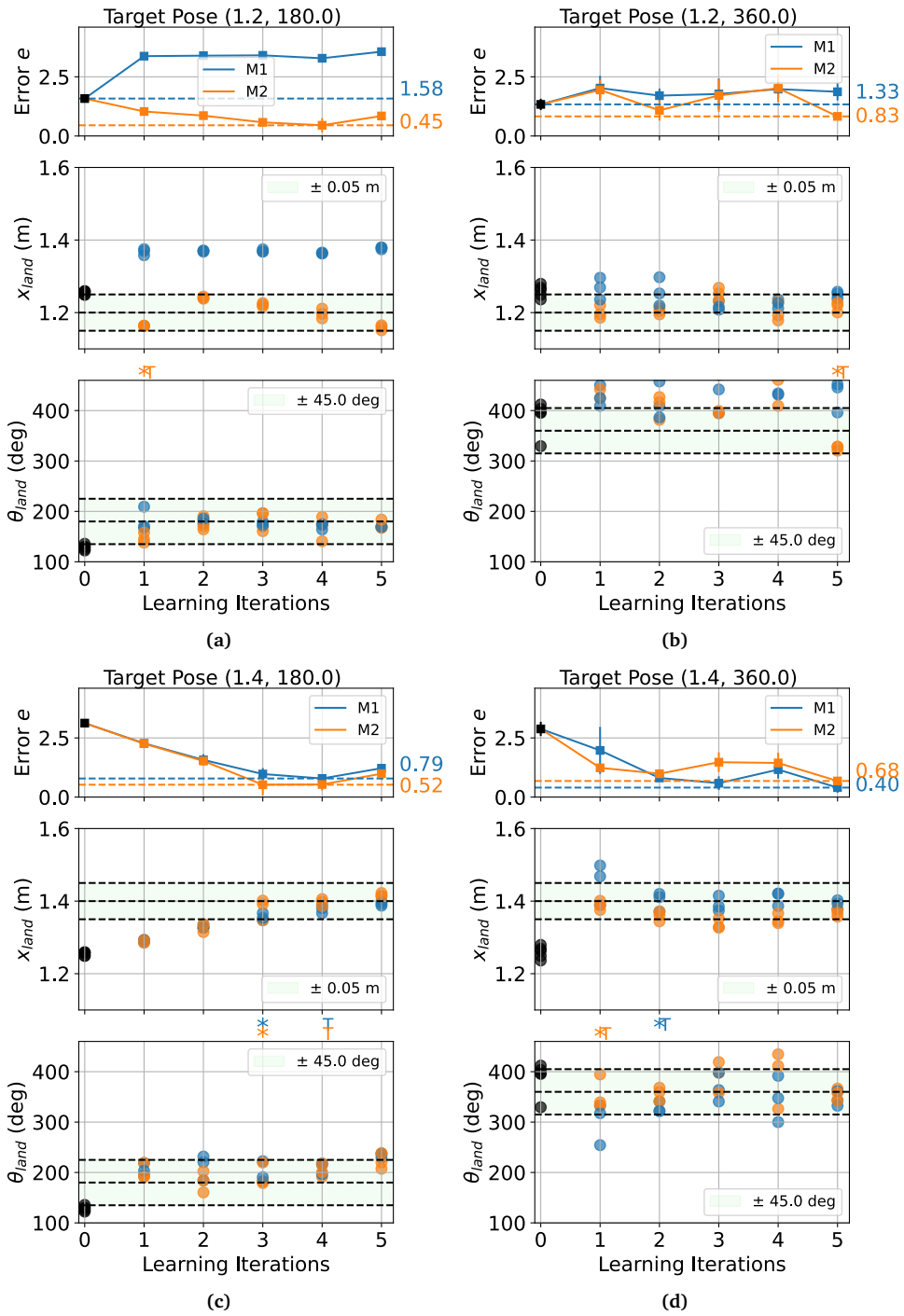


Figure 21: Summary of iterative learning of four unseen target poses. Symbol '*' represents the first iteration where the landing pose falls within the position and orientation threshold for **at least 2 out of 3 trials**. Symbol 'T' represents the first iteration where the landing pose falls within the position and orientation threshold for **all the 3 trials**.

First iteration entering threshold: Model 2, which incorporates projectile dynamics, reaches the target threshold in fewer iterations than Model 1 for all four target poses.
Error correction: Our system design demonstrates the ability to escape local minima. In particular, for the fourth iteration of (Target (b), Model 2) and the fourth iteration of (Target (d), Model 1), the landing pose exhibits greater error than in previous iterations, indicating that the neighbor-constructed local model was inaccurate. However, a new set of nearest neighbors is selected thanks to the re-selection routine, allowing the system to correct the large error in the next iteration.

Transfer learning under CoM shift In this experiment, we demonstrate **Model 3: transfer learning** can accelerate iterative learning for new objects under CoM shift. The transfer learning method follows the procedure described in Section III.G. Compared to the previously thrown bar, we shift the metal payload from the middle to the furthest position away from the tip, effectively changing the object's CoM offset from 12 cm to 18 cm.

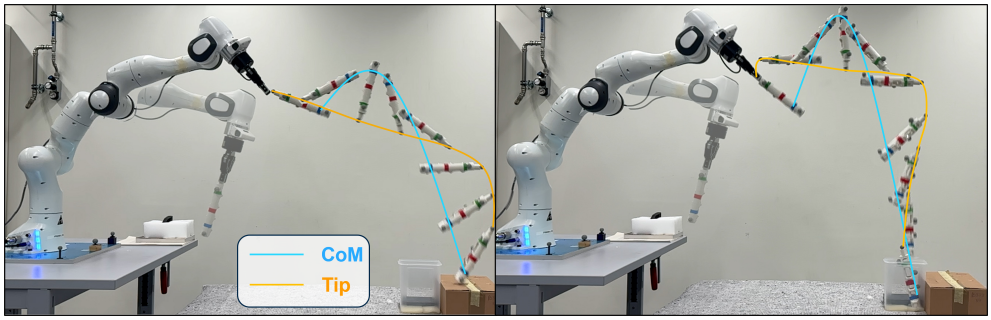


Figure 22: Transfer learning under CoM shift. Left: the command to throw-flip the original bar is invalid for the CoM-shifted bar. Right: A new valid command is found after transfer learning within few iterations.

Fig. 22 illustrates the effect of increased landing position under CoM shift and the result of transfer learning. The learning progress is summarized in Fig. 23. For comparison, we also include a baseline approach that does not incorporate physical knowledge of CoM shift. In this baseline, the support set simplex is gathered using four commands, followed by standard iterative learning with Model 2.
First iteration entering threshold: Strikingly, transfer learning with Model 3 reaches a 2/3 success rate after two iterations and 3/3 success rate after three iterations, even before entering the normal iterative learning process. In contrast, Model 2 requires a total of 5 iterations to reach 2/3 success rate, and failed to achieve 3/3 success rate after 9 iterations. This demonstrates Model 3's ability to effectively *reuse* past experience when throwing unseen objects, leveraging physical knowledge for faster adaptation. **Iterative improvement:** After 6 iterations, Model 3 achieves a significantly lower normalized error than Model 2 (0.3 vs. 0.88). This improvement is likely due to the fact that the data population gathered by Model 3 is much closer to the target (as indicated by the smaller normalized errors in the first three iterations) compared to the standard support simplex used in Model 2. Consequently, the local approximation quality in Model 3 remains higher throughout learning, leading to better decisions during the iterative process.

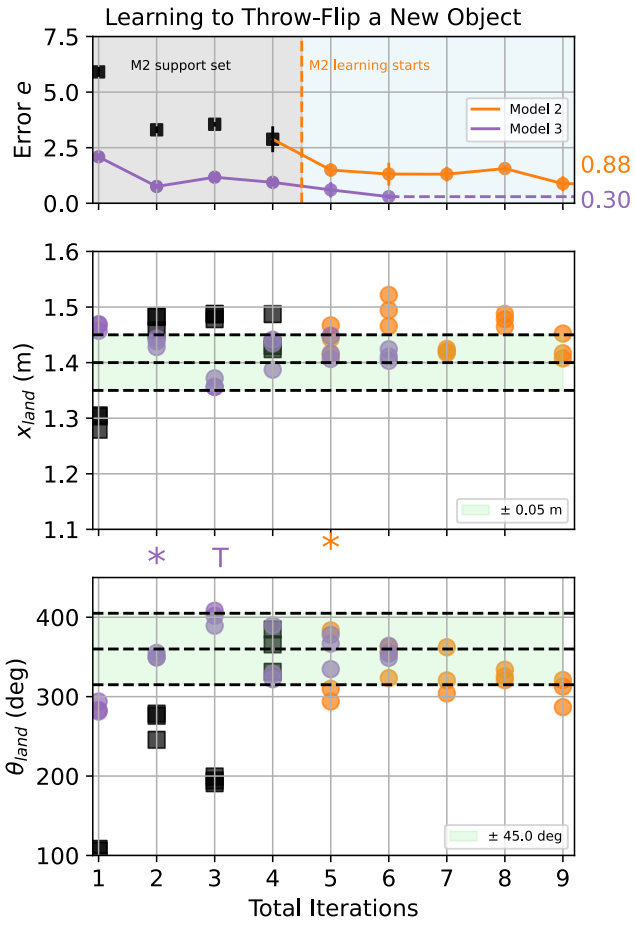


Figure 23: Summary of transfer learning. Symbol '*' represents the first iteration where the landing pose falls within the position and orientation threshold for **at least 2 out of 3 trials**. Symbol 'T' represents the first iteration where the landing pose falls within the position and orientation threshold for **all the 3 trials**.

3 Throwing (UNIPI)

3.1 Adaptive control for throwing

To move robotics manipulators into daily living situations, one has to guarantee a safe physical interaction with the environment and humans. For this purpose, most of the control algorithms used in these situations are based on impedance control [52, 20]. This technique enables the assignment of a desired compliant behavior to the manipulator through the design of an appropriate control law, permitting it to interact with the environment without the generation of large contact forces. Several works were done exploiting this type of strategy [52, 3]. However, in these controllers, errors in the dynamic model of the manipulator could lead to a tracking error on the desired motion, especially in highly dynamic tasks such as throwing.

One of the traditional control algorithms in the literature that reaches asymptotic stability under parameter uncertainties is the adaptive computed torque control [13, 50,

16, 14], which consists in a feedback linearization with the joint torque as system input. However, for the parameters update the law includes the estimated mass matrix inverse and the joint accelerations, which could lead to numerical problems such as the ill-conditioning of the estimated mass matrix or the necessity of the double derivation of the joint position measurements to obtain acceleration. Furthermore, the solution of the Lyapunov equation of the linearized model of the closed-loop system is required.

To overcome this problem, we propose a novel adaptive law for the computed torque algorithm that does not require the aforementioned information. We formulate the control law both in the joint and in the Cartesian domain, to cover all the possible reference trajectory types. In this section, we briefly recap both the theoretical formulation and the experimental validation. For more details, we refer the interested reader to *"Low-Gains Adaptive Computed Torque Control for High Dynamic Tasks," Simonini, Baracca, Cavaliere, Bicchi, and Salaris, submitted to IEEE Access, 2025* [51].

3.1.1 Control law formulation

Joint space controller: Let us consider the well-known dynamic model of a rigid robot manipulator described by Lagrange equation (see [21]):

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau\,. \tag{29}$$

As pointed out in [22], (29) is linear in the inertia parameters $\pi \in \mathbb{R}^p$ (link masses, first and second moments of inertia), and can be hence rewritten as

$$\tau = Y(q,\dot{q},\ddot{q})\pi, \tag{30}$$

where $Y \in \mathbb{R}^{n\times p}$ is the regressor matrix and p_d is the number of parameters.

Let us define $\hat{\pi} \in \mathbb{R}^p$ as an estimate of π, and let $\big(\hat{M}(\cdot),\hat{C}(\cdot),\hat{G}(\cdot)\big)$ be the dynamic matrices evaluated in $\hat{\pi}$. Let us define $q_d(t) \in \mathbb{R}^n \times \mathbb{R}$ as the desired trajectory. Finally, let then $\tilde{\pi} \triangleq \pi - \hat{\pi}$ be the parameters' error and $e(t) = q_d(t) - q(t)$ be the tracking error.

The classical computed torque control [22] provides a law that feedback linearises the nonlinear system and requires the knowledge of the dynamic model to generate the feed-forward and feedback torque action

$$\tau = \hat{M}(q)\ddot{q}_d + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q) + k_v\dot{e} + k_p e, \tag{31}$$

where k_v, k_p are positive definite gain matrices. Substituting (31) in (29), the error dynamics becomes

$$\hat{M}(q)\ddot{e} + k_v\dot{e} + k_p e = \tilde{M}(q)\ddot{q} + \tilde{C}(q,\dot{q})\dot{q} + \tilde{G}(q). \tag{32}$$

Under the assumption of a perfect knowledge of the dynamic model, i.e., $\hat{\pi} \equiv \pi$, the error follows a mass-spring-damper dynamics with linear stiffness and damping, i.e.,

$$M(q)\ddot{e} + k_v\dot{e} + k_p e = 0,$$

with guarantees of asymptotic stability due to positive definiteness of $M(q), k_v, k_p$. In case of parameters uncertainties, i.e., if $\hat{\pi} \neq \pi$, (32) can be written as

$$\ddot{e} + \hat{M}^{-1}(q)\big(k_v\dot{e} + k_p e\big) = \hat{M}^{-1}(q)Y(q,\dot{q},\ddot{q})\tilde{\pi}.$$

A possible solution is to consider a new state $x = (e^\top, \dot{e}^\top)^\top$, which represents the tracking error and its derivative, and rewrite (29) in state-space form as follows

$$\begin{cases} \dot{x} = Ax + B\hat{M}^{-1}(q)Y(q,\dot{q},\ddot{q})\tilde{\pi} \\ \dot{\tilde{\pi}} = -\dot{\hat{\pi}} \end{cases}$$

where $A \in \mathbb{R}^{2n\times 2n}, B \in \mathbb{R}^{2n\times 2n}$ are the dynamic and input matrices of a minimal state space realization of the error system. By choosing the parameters' update law as

$$\dot{\hat{\pi}} = R^{-1}Y^{\top}(q,\dot{q},\ddot{q})\hat{M}^{-\top}(q)B^{\top}Px \tag{33}$$

where $R \in \mathbb{R}^{p\times p}$ is a gain matrix and $P \in \mathbb{R}^{n\times n}$ is the solution of the Lyapunov equation, it is possible to show the asymptotic stability of the tracking error (we refer the interested reader to [13] for the theoretical proof). With this law, it is possible to assign the desired stiffness k_p and damping k_v to the manipulator. However, while the theoretical background is strong, the technical implementation could have different problems. In fact, the parameters' update law requires the inversion of the estimated mass matrix $\hat{M}$ (whose could be ill-conditioned) and the actual joint acceleration $\ddot{q}$ (which is not directly measurable and has to be obtained through differentiation). Furthermore, it requires also the solution P of the Lyapunov equation, which is related to the controller gains and has to be computed each time they change.

To overcome these problems, considering the system (29), we defined the parameters update law as

$$u_{\pi} = R_p\Big(Y_r^{\top}\dot{e} + \gamma Y^{\top}R_t(\tau - Y\hat{\pi})\Big), \tag{34}$$

with $R_p \in \mathbb{R}^{p\times p}, R_t \in \mathbb{R}^{n\times n}$ positive definite matrices and $\gamma > 0$. It is important to note that $Y_r(q,\dot{q},\dot{q}_r,\ddot{q}_r)$ is the Slotine-Li regressor, which is a generalization of the classical one $Y(q,\dot{q},\ddot{q})$. Indeed, substituting $(\dot{q}_r,\ddot{q}_r)$ with $(\dot{q},\ddot{q})$ the two matrices are identical.

Task-space controller: Despite the most simple way to control a manipulator is to generate the motion in the joint space, in terms of task planning the easiest approach is to compute the reference trajectories in the Cartesian domain. Nowadays, it is common to have manipulators with more than 6 degrees of freedom. In this case, the controller has to be able to handle the redundancy of the system. Considering the system (29), defining $(\cdot)^{+}$ as the right pseudo-inverse operator, i.e., $A^{+} \triangleq A^{\top}(AA^{\top})^{-1}$, the system (29) can be written in the task space by substituting $\dot{q} = J^{+}\dot{\xi} + N\dot{a}$, and $\ddot{q} = J^{+}\big(\ddot{\xi} + \dot{J}\dot{q}\big) + N\ddot{a}$. The variables $\dot{a}, \ddot{a}$ are the joint velocity and acceleration that do not affect the motion of the end-effector, given that they are projected in the null of the Cartesian space by the null projector $N = I - J^{+}J$.

Consider the system (29), define a desired trajectory on the Cartesian-space $\xi_d(t) \in \mathrm{SE}(3)\times\mathbb{R}$ and let $e_{\xi} = \xi_d - \xi$ be the end-effector error. Using the control law

$$\tau = \hat{M}\alpha + \hat{C}\beta + \hat{D} + \hat{G} + k_v(\beta - \dot{q}) + J^{\top}k_{\xi_p}e_{\xi}, \tag{35}$$

where $\alpha \triangleq J^{+}\big(\ddot{\xi}_d - \dot{J}\dot{q}\big) + N\ddot{a}_d$, $\beta \triangleq J^{+}\dot{\xi}_d + N\dot{a}_d$ are desired velocity and acceleration respectively, and $k_{\xi_p} \in \mathbb{R}^{6\times 6}$ is the gain matrix for the error in the Cartesian space, in conjunction with the parameters update law

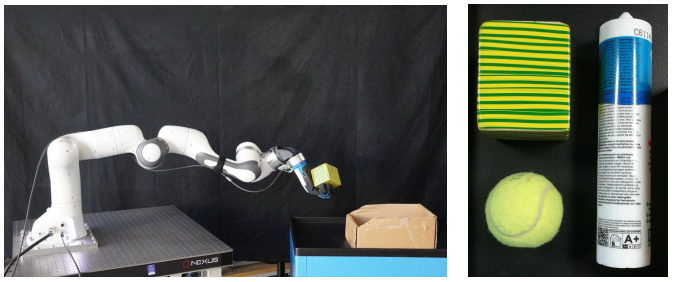
$$u_{\pi} = R_p\Big(Y_r^{\top}(q,\dot{q},\beta,\alpha)\big(\beta - \dot{q}\big) + \gamma Y^{\top}(\cdot)R_t\big(\tau - Y(\cdot)\hat{\pi}\big)\Big), \tag{36}$$

with $R_p \in \mathbb{R}^{p\times p}, R_t \in \mathbb{R}^{n\times n}$ positive definite matrices and $\gamma > 0$, the closed loop system is stable and the end-effector follows the desired trajectory, i.e.,

$$\lim_{t\to\infty} \big\|e_{\xi}\big\| = 0.$$

3.1.2 Validation in throwing task

This section presents the validation part related to the launch task, as an example of a dynamic task where an accurate knowledge of the dynamic model has an important effect



(a) Setup. **(b)** Objects.

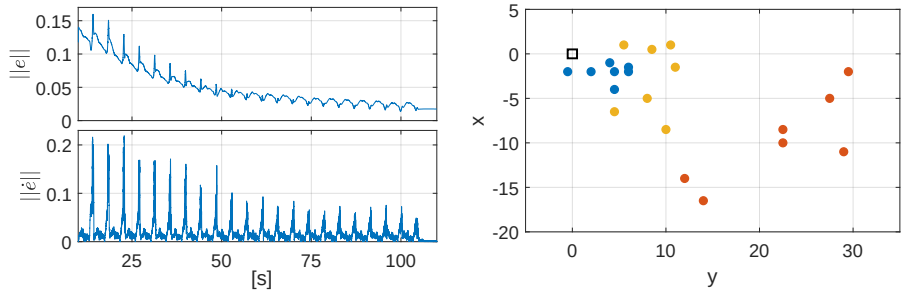
Figure 24: Experimental setup and objects used in the throws. The objects are a wooden box (221 g) in the top right, a heavy tennis ball (427 g), and a silicon tube (445 g).

in terms of its success rate. For the description of the whole validation experiments, we refer the interested reader to [51]. In our case, the controller has no prior information about the inertial properties of the object, and it has to learn by itself. In this type of task, the object has to reach a desired landing position $p_d = (x_d, y_d, z_d)$. For this purpose, the gripper must release the object in a precise Cartesian pose and with a precise velocity. Considering the projectile motion, the object movement is planar on the plane defined by the end-effector velocity at the release time and the gravity acceleration vector $\vec{g}$, which in our reference system is aligned with the z axes.

In this work, we assume that the object release time is instantaneous. In this way, the trajectory followed by the robot before the release does not affect the throw. We choose a minimum-jerk trajectory to go from the initial end-effector position $\xi_0 \in \mathbb{R}^3$ to the release point $\xi_r \in \mathbb{R}^3$ and velocity $\dot{\xi}_r \in \mathbb{R}^3$. The choice of this type of trajectory to perform the task was done to have smooth movements (i.e., continuous acceleration and limited jerk). The orientation is considered fixed to have no angular velocity and to simplify the hand opening. It is worth noting that the solution proposed in this work to perform the throwing is not necessarily the optimal one. However, the focus of this part is on testing the improvement introduced by our controller in dynamic tasks. The integration of this approach with optimal throwing motion will be part of future works.

The experimental setup consists of a Franka Emika Panda equipped with a SoftHand as the gripper. The manipulator, starting from a predefined initial position, has to throw the grasped object to the desired goal. The dynamic model of the manipulator takes into account the inertia of the SoftHand, but not the one of the grasped object. This choice was made to emulate the situation where the manipulator needs to launch an unknown object picked using a soft under-actuated gripper. In this case, even if the system knows exactly the inertia of the object by itself, the structure of the gripper does not permit us to know how the object is grasped and to modify accurately the dynamic model of the robot.

Precise throwing: With this setup, we tested three different controllers: our adaptive computed torque with low gains ($k_p \approx 30N/m$), the classical non-adaptive computed torque with low gains ($k_p \approx 30N/m$), and the classical non-adaptive computed torque with high gains ($k_p \approx 300N/m$). The choice of performing tests for the non-adaptive controller with two levels of gains was made because, usually, the simplest way to make the system less sensible to inertial model errors is to increase the stiffness of the manipulator. However, this solution leads to higher interaction forces with the environment, decreasing the safety of the system. With this test, we want to prove that our approach can achieve better performances even while keeping a low stiffness profile. The gain k_v was chosen in each case to obtain a critical or overdamped damping behavior in order to avoid oscillations.



(a) Tracking error. **(b)** Landing point distribution.

Figure 25: Throwing experiments. In 25a position and velocity tracking errors are plotted. To correct the inertia parameter errors, the manipulator repeats several times the throwing trajectory without releasing the object and then performs the launch. In Fig. 25b the landing point distribution is plotted. The black rectangle is the desired landing point. The landing points are referred to: (**adaptive low gains**, **non-adaptive low gains**, **non-adaptive high gains**)

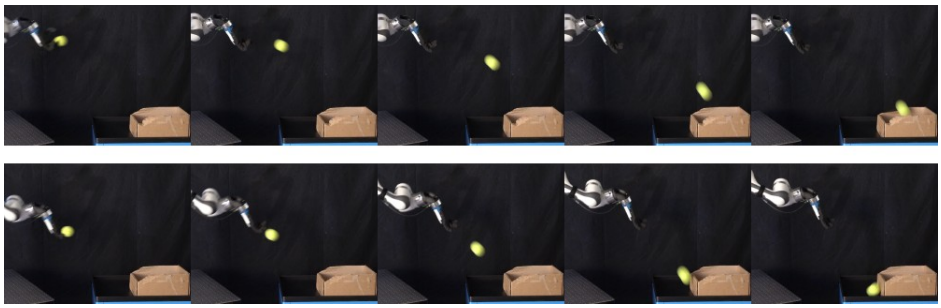


Figure 26: Throwing sequence of the robot. The heavy tennis ball reaches the box only in the case in which adaptive control is used.

Figure 27: Throwing with the elastic wrist. The robot movement exploits the elasticity to improve throwing.

As introduced before, the motion performed in this set of tests consists of the minimum jerk trajectory which connects the defined starting point to the desired releasing point to launch to the desired target. As an object to throw, we use a silicon tube (Fig. 24b) with a weight of 445 g. For the adaptive controller, before performing the throwing, the manipulator repeats 4 times the desired trajectory to permit the update law to correct the dynamic model. It is worth noting that this strategy is not the optimal one to estimate the inertial parameters. In the literature, different works address this problem [47, 7]. However, the optimal trajectory generation for inertial parameter estimation is not the main focus of this work and requires a separate discussion.

To evaluate the performances, we measured the dispersion of the position reached by the object after throwing with respect to the desired target. In Fig. 25b, we can observe the results obtained by the three controllers tested, proving that our approach outperforms the others. In Fig. 25a instead we can see the convergence of the position and velocity errors during the "learning phase" before the throw.

Throwing in a box: To make an additional test we made a second phase of throws, in which the purpose was to land into a box placed over the robot's natural workspace. The experimental setup is shown in Fig. 24a. For these tests, we compare the adaptive and non-adaptive controllers with two new objects. The objects chosen are a wooden box and a heavy tennis ball, represented in Fig. 24b. In these tests, we notice that the experiments with the non-adaptive controllers do not reach the box because they cannot compensate for the errors due to the weight of the grasped object. Fig. 26 shows a photo sequence with the heavy tennis ball.

3.2 Throwing with elastic wrist

At the Maker-Faire Rome held from October 25 to 27, 2024, the University of Pisa demonstrated a task involving the use of a Franka Emika Panda manipulator equipped with a variable stiffness actuator (VSA) wrist. The task concerned throwing lightweight paper boxes into a larger target box, utilizing the VSA wrist's ability to store and release energy. This wrist, featuring two motors to control the equilibrium angle and stiffness, allowed the system to exploit compliant behavior for dynamic manipulation. During each throw, the stiffness and equilibrium angle remained fixed to simplify control while leveraging the wrist's energy-storing properties.

The manipulator's trajectory was designed to store energy in the springs of the VSA wrist during the arm's acceleration phase and release it during deceleration, enhancing the velocity and accuracy of the throw. This approach highlighted the advantages of using a compliant actuator to improve throwing dynamics compared to rigid joints. Future developments will focus on optimizing the arm's trajectory with advanced algorithms and dynamically adjusting the wrist's parameters to further enhance performance. An image captured during the event shows the manipulator in action (Fig. 27), illustrating the throw of the Franka Emika Panda arm, equipped with the VSA wrist. A video can be found at this *link*[1]

4 Throwing (TUM)

4.1 Embedding of throwing capabilities via system features

Optimizing for the best throwing performance can be approached from several sides. From one side, the focus was on increasing the capabilities of the hardware in order to provide the best manipulability while featuring BSA joints [43] that have been in focus for TUM development. On the other side, efforts were placed on modeling, control, and introduction of familiar interfaces (similar to Franka) that should foster collaboration between partners in order to test the capabilities of the newly developed system.

Main hardware feature for the elastic system is the addition of BSA as an Integrated actuator. Previous BSA actuator ($\mathbb{BSA}^a$) was implemented using our configurable testbed [17]. However, due to the serial architecture of the actuator, its excessive mass, length and, by extension, its power-to-weight ratio make it unsuitable for complex robot morphologies beyond particular research cases [42]. Integrated design has allowed largely to mitigate all previously mentioned concerns. Prototype represetns a 3 DoF robot with two rigid joints and one BSA joint at the end. More description regarding integrated platform can be found in deliverable **D1.5**.

Beside the form factor, important hardware performance features include custom cage spring design, revised electronics, featuring faster sampling and PWM control (increased to 16 kHz and 80 kHz, respectively), more slaves for SPI communication (SPI based rotor position sensing for more robust readings, etc.). Main Control Architecture has been extended with ROS interfaces and C++ UDP based controller. Refer to Deliverable **D1.5** for more details.

4.2 Validating and benchmarking throwing capabilities (with EPFL)

To validate the hardware platform and benchmark its throwing capabilities in preparation for future assessments of the novel BSAs, we collaborated with EPFL during the MS3 milestone integration in June 2024 in Munich. This collaboration involved integrating their software stack for throwing with our low-level control interfaces through ROS. As a result, we achieved the first throwing experiments with our platform. In the throwing experiments, the setup is as follows,

Gripper and Thrown Object
As demonstrated in Section 2.1 by EPFL, prehensile throwing using grippers and anthropomorphic hands is subject to release uncertainties. To isolate the throwing uncertainties caused by release variability and focus solely on the behavior and performance of our

[1] https://drive.google.com/drive/folders/1nQ2HFcd0ZK6pHvzJ-tlJQBqaRqbVoRKv?usp=drive_link

control stack, we used a SCHUNK magnetic gripper. The thrown object was a keyring, which offers a high level of repeatability in object release.

Throwing Motion Generation
While TUM is still developing optimal control algorithms to leverage elasticity, we treated our platform as a conventional robot with SEAs for this validation. This allowed us to reuse EPFL's existing algorithms and software stack, providing a strong benchmark to facilitate future studies. Utilizing EPFL's previous algorithm on throwing planning [33] reported in D4.5, we obtained nominal throwing configurations in the joint state space using the D-H parameters of our arm. The throwing trajectory was then generated using an open-source trajectory generator, Ruckig [4], given the initial grasping state and the nominal throwing state. Finally, the throwing trajectory is tracked by our velocity controller.

Throwing Result
Qualitatively, the platform is able to repeatedly throw the keyring into a target box with dimensions of 50 cm × 40 cm (smaller than the Darko target box, which measures 70 cm × 40 cm) located 2.2 m away from the base, achieving a 100% success rate. This demonstrates the high level of control accuracy of our hardware platform in the throwing task. A video showing single throw experiment can be found at this *link*[1].

4.3 Optimal control for elasticity-aware throwing

Robot modeling: with motor, spring, link-side configuration and the state are defined as θ, ψ, $\mathbf{q}$, and $\xi := (\psi, \mathbf{q})$, the equation of motion of the system can be expressed as

$$\begin{gathered}\underbrace{\begin{bmatrix}\boldsymbol{B}_\psi & \mathbf{0}\\ \mathbf{0} & \boldsymbol{M}(\mathbf{q})\end{bmatrix}}_{=:\boldsymbol{\Pi}(\xi)}\ddot{\xi}+\underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{h}(\mathbf{q},\dot{\mathbf{q}})\end{bmatrix}}_{=:\eta(\xi,\dot{\xi})}+\underbrace{\begin{bmatrix}\boldsymbol{K}(\theta-\psi)\\ \mathbf{0}\end{bmatrix}}_{=:\tau(\xi)}+\tau_f=\tau_c,\\ \mathbf{l}_p(\dot{\xi})=\mathbf{0}.\end{gathered} \tag{37}$$

where $\boldsymbol{B}_\psi$ and $\boldsymbol{M}(\mathbf{q})$ are the spring and link-side inertia matrices, $\mathbf{h}(\mathbf{q},\dot{\mathbf{q}})$ is the link-side nonlinear bias term including Coriolis, centrifugal and gravitational terms, $\mathbf{K}$ is a stiffness matrix, τ_f and τ_c are friction and clutch torques, respectively. Clutches prevent relative motion between frames. Thus, for each active clutch, the corresponding relative velocities (concatenated in the vector $\mathbf{l}_p(\dot{\xi})$) are set to zero.

For preliminary testing, model parameters have been largely taken from CAD. Later on, the parameter identification procedure based on the [29] has been adopted. Parameter regrouping was used for the optimal representation of dynamics for identification. Identification trajectories has been characterized by minimal condition number. For identified linear parameter combinations, feasible parameter search has been performed using the Riemannian gradient method. Utilizing clutching modes, additionally, it was possible to precisely estimate combined inertia of clutch and spring. In coupled mode, by relating position and torques, spring stiffness was identified.

Two approaches have been used to generate the throwing motion. The first approach is utilizing the elasticities and the full system model (including the hybrid dynamics transitions). The OC problem for End-effector speed maximization has been defined, as in [17]. To deal with hybrid modes, we select a switching sequence a priori. Subsequently, we solve a multiphase optimization problem, with the phase duration T_p being one of the

[1] https://drive.google.com/file/d/126pju$_X$ZNzQSJ4ZWFf−mMgHU1AMWC5TK/view?usp=sharing

decision variables.

$$\begin{aligned} \min_{\mathbf{x}(t),\mathbf{u}(t),T_p} \quad & \mathscr{J}(\mathbf{x}(t),\mathbf{u}(t)) \\ \text{s.t.} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}_p(\mathbf{x}(t),\mathbf{u}(t)), \;\; t \leq T_p \\ & \mathbf{x}_p^+(t) = \mathbf{g}_p(\mathbf{x}^-(t)), \;\; t = T_p \\ & \mathbf{x}(t) \in \mathscr{X}, \; \mathbf{u}(t) \in \mathscr{U}, \end{aligned} \tag{38}$$

We select BRK → SEA as the sequence for the actuator modes. Thus, the system can preload the spring while the other joints move. Then, the stored potential energy is converted to kinetic energy in a launch-like motion – rapidly accelerating the link forward. Despite a limited motor speed of 4.5 rad/s, the robot reached a final cartesian velocity of 4.1 m/s. This is already outperforming our previous prototype and rigidly actuated robots. Experimental validation has proven repeatability of the preformance and overall system reliability.

5 Conclusion and outlook

5.1 EPFL

Our primary future work focuses on the systematic experimental validation of learning to throw-flip, particularly the benefits of incorporating prior physical structures for improved sample complexity and transferability (e.g., utilizing past data to throw-flip new objects). In addition to the restrictive landing goals in throw-flipping, we are also interested in iterative learning to throw objects to a desired location with arbitrary grasping poses, especially in the context of major use cases in Darko that primarily require achieving a desired location irrespective of landing orientation.

5.2 UNIPI

The preliminary throwing tests performed with elastic elements in the kinematic chains could be beneficial in increasing the range of throwing of a classic manipulator. The next step will be to design an optimization algorithm capable of exploiting the elasticity introduced by the elastic wrist to obtain precise throws maximizing the distance reached.

The other point to be addressed is the shooting with the pneumatic tool. In the previous reports, we showed that our tool could be a feasible way to reach targets far from the manipulator. In the next months we will improve the system with a pneumatic model of the tool capable to increase the precision of the shot. This system will also be integrated with the picking routine to be performed with the manipulator to accomplish a complete cycle of the pick & throw routine.

5.3 TUM

Future efforts are focusing on implementing an efficient planning solution that is optimally using the clutches for various throwing targets. The plan is to take advantage of offline computed data at various throwing configurations. The algorithm should make decisions online based on the given target and current robot position, as well as ensure repeatability and reliability of the performance for this more general case.

References

[1] Robot juggling: implementation of memory-based learning. *IEEE Control Systems Magazine*, 14(1):57–71, 1994.

[2] Eric W Aboaf, Christopher G Atkeson, and David J Reinkensmeyer. Task-level robot learning: Ball throwing. 1987.

[3] Fares J Abu-Dakka and Matteo Saveriano. Variable impedance control and learning—a review. *Frontiers in Robotics and AI*, 7:590681, 2020.

[4] Lars Berscheid and Torsten Kröger. Jerk-limited real-time trajectory generation with arbitrary target states. *Robotics: Science and Systems XVII*, 2021.

[5] Michael Bombile and Aude Billard. Dual-arm control for coordinated fast grabbing and tossing of an object: Proposing a new approach. *IEEE Robotics & Automation Magazine*, 29(3):127–138, 2022.

[6] Michael Bombile and Aude Billard. Bimanual dynamic grabbing and tossing of objects onto a moving target. *Robotics and Autonomous Systems*, page 104481, 2023.

[7] Vincent Bonnet, Philippe Fraisse, André Crosnier, Maxime Gautier, Alejandro González, and Gentiane Venture. Optimal exciting dance for identifying inertial parameters of an anthropomorphic structure. *IEEE Transactions on Robotics*, 32(4):823–836, 2016.

[8] Alan R Champneys and Péter L Várkonyi. The painlevé paradox in contact mechanics. *IMA Journal of Applied Mathematics*, 81(3):538–588, 2016.

[9] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[10] Lipeng Chen, Weifeng Lu, Kun Zhang, Yizheng Zhang, Longfei Zhao, and Yu Zheng. Tossnet: Learning to accurately measure and predict robot throwing of arbitrary objects in real time with proprioceptive sensing. *IEEE Transactions on Robotics*, 2024.

[11] Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. Learning neural event functions for ordinary differential equations. *arXiv preprint arXiv:2011.03902*, 2020.

[12] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[13] J. Craig, Ping Hsu, and S. Sastry. Adaptive control of mechanical manipulators. In *1986 IEEE International Conference on Robotics and Automation Proceedings*, volume 3, pages 190–195.

[14] J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson/Prentice Hall, 2005.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[16] Rajeshree Deotalu and Shital Chiddarwar. Trajectory tracking of the manipulator using adaptive computed torque control. In *2020 IEEE International Conference for Innovation in Technology (INOCON)*, pages 1–6, 2020.

[17] Edmundo Pozo Fortunić, Mehmet C. Yildirim, Dennis Ossadnik, Abdalla Swikir, Saeed Abdolshah, and Sami Haddadin. Optimally controlling the timing of energy transfer in elastic joints: Experimental validation of the bi-stiffness actuation concept. *IEEE Robotics and Automation Letters*, 8(12):8106–8113, 2023.

[18] Yizhi Gai, Yukinori Kobayashi, Yohei Hoshino, and Takanori Emaru. Motion control of a ball throwing robot with a flexible robotic arm. *International Journal of Computer and Information Engineering*, 7(7):937–945, 2013.

[19] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. *Wear*, 143(2):307–330, 1991.

[20] Neville Hogan. Impedance control: An approach to manipulation: Part ii—implementation. *Journal of Dynamic Systems, Measurement, and Control*, 107(1):8–16, 1985.

[21] John Hollerbach, Wisama Khalil, and Maxime Gautier. *Dynamics*, pages 37–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[22] John Hollerbach, Wisama Khalil, and Maxime Gautier. *Motion Control*, pages 163–194. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[23] Robert D Howe and Mark R Cutkosky. Practical force-motion models for sliding manipulation. *The International Journal of Robotics Research*, 15(6):557–572, 1996.

[24] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[25] Maarten Johannes Jongeneel, Luuk Poort, Nathan van de Wouw, and Alessandro Saccon. Experimental validation of nonsmooth dynamics simulations for robotic tossing involving friction and impacts. 2023.

[26] Farshad Khadivar, Sthithparagya Gupta, Walid Amanhoud, and Aude Billard. Efficient configuration exploration in inverse dynamics acquisition of robotic manipulators. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1934–1941. IEEE, 2021.

[27] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 3 edition, 2002.

[28] Jens Kober, Katharina Muelling, and Jan Peters. Learning throwing and catching skills. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5167–5168. IEEE, 2012.

[29] Fernando Díaz Ledezma and Sami Haddadin. Ril: Riemannian incremental learning of the inertial properties of the robot body schema. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9354–9360, 2021.

[30] Yang Liu and Aude Billard. Tube acceleration: Robust dexterous throwing against release uncertainty. *IEEE Transactions on Robotics*, 40:2831–2849, 2024.

[31] Yang Liu and Aude Billard. On transient release dynamics in robot throwing: A sliding pivot model. *Submitted to IEEE Transactions on Robotics*, 2025. Under review.

[32] Yang Liu, Bruno Da Costa, and Aude Billard. Learning to throw-flip. In *Submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2025*, 2025. Under review.

[33] Yang Liu, Aradhana Nayak, and Aude Billard. A solution to adaptive mobile manipulator throwing. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1625–1632. IEEE, 2022.

[34] Kevin M Lynch and Matthew T Mason. Dynamic nonprehensile manipulation: Controllability, planning, and experiments. *The International Journal of Robotics Research*, 18(1):64–92, 1999.

[35] Matthew T Mason and Kevin M Lynch. Dynamic manipulation. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, volume 1, pages 152–159. IEEE, 1993.

[36] Hideyuki Miyashita, Tasuku Yamawaki, and Masahito Yashima. Control for throwing manipulation by one joint robot. In *2009 IEEE International Conference on Robotics and Automation*, pages 1273–1278. IEEE, 2009.

[37] Maxim Monastirsky, Osher Azulay, and Avishai Sintov. Learning to throw with a handful of samples using decision transformers. *IEEE Robotics and Automation Letters*, 8(2):576–583, 2022.

[38] Wataru Mori, Jun Ueda, and Tsukasa Ogasawara. 1-dof dynamic pitching robot that independently controls velocity, angular velocity, and direction of a ball: Contact models and motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 1655–1661. IEEE, 2009.

[39] Masafumi Okada, Shota Oniwa, and Wataru Hijikata. Robust throwing design based on dynamic sensitivity analysis. *Mechanical Engineering Journal*, 5(1):17–00442, 2018.

[40] Masafumi Okada, Alexander Pekarovskiy, and Martin Buss. Robust trajectory design for object throwing based on sensitivity for model uncertainties. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3089–3094. IEEE, 2015.

[41] Masafumi Okada and Takahiro Sekiguchi. Throwing motion design based on minimum sensitivity with respect to error covariance of robot dynamic parameters. *Mechanical Engineering Journal*, 8(1):20–00299, 2021.

[42] Dennis Ossadnik, , Vasilije Rakcevic, Mehmet C. Yildirim, Edmundo Pozo Fortunić, Hugo T. M. Kussaba, Abdalla Swikir, and Sami Haddadin. Optimal control for clutched-elastic robotic systems: A contact-implicit approach. In *2024 (ICRA))*, 2024.

[43] Dennis Ossadnik, Mehmet C. Yildirim, Fan Wu, Abdalla Swikir, Hugo T. M. Kussaba, Saeed Abdolshah, and Sami Haddadin. Bsa - bi-stiffness actuation for optimally exploiting intrinsic compliance and inertial coupling effects in elastic joint robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3536–3543, 2022.

[44] Alexander Pekarovskiy and Martin Buss. Optimal control goal manifolds for planar nonprehensile throwing. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4518–4524. IEEE, 2013.

[45] Lev Semenovich Pontryagin, EF Mishchenko, VG Boltyanskii, and RV Gamkrelidze. *The Mathematical Theory of Optimal Processes*. 1962.

[46] Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. *Operations Research*, 66(1):230–252, 2018.

[47] Paolo Salaris, Marco Cognetti, Riccardo Spica, and Paolo Robuffo Giordano. Online optimal perception-aware trajectory generation. *IEEE Transactions on Robotics*, 35(6):1307–1322, 2019.

[48] Stefan Schaal and Christopher G Atkeson. Open loop stable control strategies for robot juggling. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 913–918. IEEE, 1993.

[49] Taku Senoo, Akio Namiki, and Masatoshi Ishikawa. High-speed throwing motion based on kinetic chain approach. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3206–3211. IEEE, 2008.

[50] Wei-Wei Shang, Shuang Cong, and Yuan Ge. Adaptive computed torque control for a parallel manipulator with redundant actuation. *Robotica*, 30:457 – 466, 05 2012.

[51] Giorgio Simonini, Marco Baracca, Tommasio Valerio Cavaliere, Antonio Bicchi, and Paolo Salaris. Low-gains adaptive computed torque control for high dynamic tasks. *Submitted to IEEE Access*, 2025. Under review.

[52] Peng Song, Yueqing Yu, and Xuping Zhang. A tutorial survey and comparison of impedance control on robotic manipulation. *Robotica*, 37(5), 2019.

[53] Bart Van Parys and Negin Golrezaei. Optimal learning for structured bandits. *Management Science*, 70(6):3951–3998, 2024.

[54] Nicholas Xydas and Imin Kao. Modeling of contact mechanics and friction limit surfaces for soft fingers in robotics, with experimental results. *The International Journal of Robotics Research*, 18(9):941–950, 1999.

[55] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[56] Ahmed Zermane, Niels Dehio, and Abderrahmane Kheddar. Planning impact-driven logistic tasks. *IEEE Robotics and Automation Letters*, 2024.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017274