Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MASTERTHESIS

Evaluation of Robotic Systems for Throwing Tasks

vorgelegt von

Lukas Sommerhalder

MIN-Fakultät
Fachbereich Informatik
TAMS
Studiengang: Wirtschaftsinformatik
Matrikelnummer: 7785880
Abgabedatum: 21.03.2026
Erstgutachter: Prof. Dr. Jianwei Zhang
Zweitgutachter: Dr. Norman Hendrich

Preface

I would like to sincerely thank the TAMS Group at the University of Hamburg for their invaluable support throughout this research. Their expertise, guidance, and generous access to robotic equipment were instrumental, and the collaborative and welcoming environment they created greatly enriched my work. I am especially grateful to my primary advisers, Norman Hendrich and Michael Görner, whose foundational research, insightful guidance, and deep knowledge provided an essential springboard, shaping the skills and understanding that made this thesis possible.

Lukas Sommerhalder
Hamburg, Germany
21.03.2026

Abstract

This research presents an investigation of robotic throwing, with a focus on delta robots and articulated robotic arms. In contrast to conventional pick-and-place operations, throwing can significantly extend the operational range while also reducing task execution time.

The focus of this work is to evaluate the potential of delta robots and articulated robotic arms for robust throwing of grasped objects.

Regarding delta robots, the integration of a final section into the throwing process that follows the object's flight trajectory during release is investigated.

For articulated robotic arms, where such a motion may be infeasible due to kinematic and dynamic constraints, an algorithm is developed to identify throwing configuration with robustness to release uncertainty with constant joint velocities during release.

By combining analytical modeling, algorithm design, and experimental validation, this work contributes to more reliable robotic throwing strategies and improves the understanding of dynamic manipulation.

Contents

Preface i

Abstract iii

Contents v

List of Figures vii

List of Tables ix

1 Introduction 1
- 1.1 Background and Motivation . . . 1
- 1.2 Problem Statement . . . 1
- 1.3 Research Objectives and Contributions . . . 2
- 1.4 Thesis Organization . . . 3

2 Related Work 5
- 2.1 Non-Prehensile Throwing . . . 5
- 2.2 Prehensile Throwing with Delta Robots . . . 6
- 2.3 Prehensile Throwing with Articulated Robotic Arms . . . 6
- 2.4 Expanding Horizons: Mobile Platforms and Heavy Machinery . . . 8
- 2.5 Summary and State of the Art . . . 8

3 Prehensile Throwing with Delta Robots 9
- 3.1 Delta Robot Kinematics and Dynamics . . . 10
- 3.2 Task Setting . . . 16
- 3.3 Problem Formulation . . . 17
- 3.4 Proposed Algorithm for Robust Delta Robot Throwing . . . 17
- 3.5 Robustness Analysis and Timing . . . 22
- 3.6 Summary . . . 31
- 3.7 Future Work . . . 31

4 Prehensile Throwing with Robotic Arms **33**
4.1 Kinematic and Dynamic Challenges in Robotic Arm Throwing 33
4.2 Problem Formulation 35
4.3 Algorithm for Robust Constant-Velocity Throws 36
4.4 Simulation and Experimental Results 40
4.5 Summary 52
4.6 Future Work 52

5 Summary and Future Work **55**

Bibliography **57**

A Glossary **61**
A.1 Prehensile throwing with Delta Robots 61
A.2 Prehensile throwing with Robotic Arms 64

B Appendix **67**
B.1 Visualization of Convergence and Throw Robustness 67
B.2 Performance Evaluation of ARTS 76
B.3 Supplementary Figures from Real-World Experiments 77

C Use of Generative AI **79**

List of Figures

3.1 Experimental setup. 10
3.2 Delta robot schematic. 12
3.3 Delta robot inverse kinematics. 14
3.4 Cartesian Trajectory. 19
3.5 Cartesian Velocity. 19
3.6 Cartesian Acceleration 20
3.7 Accuracy visualization. 27
3.8 Gripper normal force during release. 28
3.9 Joint positions and velocities. 29
3.10 Joint efforts. 30

4.1 UR5 real and simulated experimental platform. 34
4.2 Configuration progression for fixed target. 42
4.3 Landing error in release interval for fixed target with ARTS configuration. 43
4.4 Visualization of robust throwing configurations. 44
4.5 Landing error in release interval for random configuration. 45
4.6 Simulated throw trajectories I. 48
4.7 Simulated throw trajectories II. 49
4.8 Objects used to assess throwing performance. 50
4.9 Overlay of throws with golf ball. 51
4.10 Overlay of throws using distinct objects. 52

B.1 Second ARTS progression for fixed target. 68
B.2 Third ARTS progression for fixed target. 69
B.3 Fourth ARTS progression for fixed target. 70
B.4 Fifth ARTS progression for fixed target. 71
B.5 Second ARTS Configuration: Landing error during release interval for fixed target. 72
B.6 Third ARTS Configuration: Landing error during release interval for fixed target. . 72
B.7 Fourth ARTS Configuration: Landing error during release interval for fixed target. 73
B.8 Fifth ARTS Configuration: Landing error during release interval for fixed target. . 73
B.9 Second random configuration: Landing error during release interval for fixed target. 74

B.10 Third random configuration: Landing error during release interval for fixed target. . 74
B.11 Fourth random configuration: Landing error during release interval for fixed target. 75
B.12 Fifth random configuration: Landing error during release interval for fixed target. . 75
B.13 Overlays of throws for Targets 1–3 using various objects. 78

List of Tables

3.1 Target positions. 24
3.2 Experimental parameter settings for each object and jerk condition. 25
3.3 Average horizontal distance to each target. 25
3.4 Accuracy results for all targets. 25
3.5 Maximum positional error for all targets. 26
3.6 Throw duration per target. 28
3.7 Peak joint loads for all targets. 30

4.1 Comparison for fixed target. 46
4.2 Comparison for random targets. 46
4.3 Target positions. 51
4.4 Throwing success results. 51

B.1 Performance of ARTS for randomly generated targets. 77
B.2 Performance of random search for randomly generated targets. 77

Chapter 1

Introduction

Throwing objects is a common skill, which can be widely observed in human activities as sports, hunting, and diverse manipulations tasks. To successfully throw, most of these contexts require precise coordination, timing, and controlled force to achieve a high accuracy.

Even so it might appear simple, throwing remains a challenging task for robotic systems. It is highly dynamic and requires precise coordination between kinematics and dynamics, and in the case of throwing a grasped object also an accurate release timing. Thus to achieve a consistent and controllable projectile motion it requires careful trajectory planning and robustness to uncertainty.

Applications of robotic throwing include sports such as basketball and baseball, entertainment like juggling, as well as industrial material handling for sorting and packaging.

The development and improvement of reliable and efficient robotic throwing strategies provide an interesting area of research within the field of robotics with wide applicability.

1.1 Background and Motivation

Interaction with the environment is a fundamental aspect of robotics. It enables robots to manipulate objects, move through their surroundings, and perform increasingly complex tasks. Among the many capabilities explored in robotics, throwing is an interesting and promising skill. Beyond extending a robot's operational reach, throwing also bears potential to outperform traditional pick-and-place (PaP) operations by reducing execution time, mechanical wear, and energy consumption.

This combination of extended reach and improved efficiency provides a strong motivation for developing robust and repeatable robotic throwing strategies.

1.2 Problem Statement

Throwing requires a precisely timed motion to reach a desired velocity of an object at a specific position, followed by a release that ensures the resulting projectile reaches a specified target. Unlike static placement tasks, success depends not only on the end-effector position, but also on the object's velocity at the instant of release. Robotic throwing motions are commonly categorized

based on whether the robot establishes a grasp on the object at release.[1] With non-prehensile throwing the object remains loosely coupled around the release position, typically relying on impacts, pushes, flicks, or guided contacts. Prehensile throwing, in contrast, involves firmly grasping the object till releasing it in a controlled manner. From a task perspective, particularly prehensile throwing can be seen as a dynamic extension of conventional PaP manipulation: the robot still picks the object and follows a planned trajectory, but instead of placing it, the object is released dynamically to reach a target. This shows the conceptual continuity with PaP while introducing additional challenges especially associated with dynamic release as an instant detachment of the object from the gripper at the release position might not be possible on most systems.

Prehensile throwing allows the robot to control the object's pose including orientation up to the release point, and provides higher flexibility to adapt to different object geometries, trajectories, and release strategies. However, this advantage comes at the cost of high sensitivity to release dynamics where even small deviations can lead to significant trajectory errors as shown in [Liu and Billard, 2024]. On the other side non-prehensile throwing avoids grasp-related complications, but introduces unactuated degrees of freedom and highly sensitive contact dynamics, making execution strongly dependent on contact geometry, timing, and impact transitions.

Throwing differs from conventional manipulation tasks in that the object must not only reach a target position, but also achieve a precise velocity at release. This requirement, combined with high execution speeds, short motion durations, and strongly nonlinear dynamics, makes accurate modeling and robust control particularly challenging. Uncertainties in object properties, contact conditions, and actuator behavior further exacerbate these difficulties, rendering reliable and repeatable robotic throwing poses a demanding problem for general-purpose robotic systems.

1.3 Research Objectives and Contributions

Existing research on robotic throwing spans a wide range of approaches, including kinematic trajectory planning, dynamic modeling of projectile motion, sensor-based feedback control using vision or inertial measurements, and the design of specialized manipulators optimized for throwing.

Despite these advances, achieving consistent robustness across varying conditions, such as changes in object mass distribution, surface properties, and target location—remains an open challenge. Many proposed methods demonstrate promising performance in controlled settings; however, their effectiveness often degrades under uncertainty or if high accuracy is required.

This thesis provides a structured overview of the state of the art in robotic throwing, synthesizing existing work on trajectory generation, dynamic modeling, control strategies, and experimental evaluation. The objective is to consolidate current knowledge, clarify key challenges, and identify promising directions for improving robustness and repeatability in robotic throwing systems.

Motivated by the widespread adoption of PaP operations in industry—and the potential to extend these workflows via pick-and-throw (PaT) paradigms—this work focuses on evaluating

[1] This distinction is also used, for example, in [Liu and Billard, 2026].

robotic throwing on platforms representative of industrial practice, specifically general-purpose manipulators equipped with standard grippers handling grasped objects. [2]

Recognizing the central role of release dynamics in prehensile throwing accuracy, this thesis focuses on improving the robustness of the release phase. A key contribution is the exploitation of the inherent kinematic and dynamic properties of delta robots for throwing tasks.

By enabling the end-effector to closely follow the object's initial flight trajectory during release, delta robots can significantly reduce sensitivity to timing errors. This property leads to improved throwing precision and highlights the potential of delta robots to support a transition from conventional pick-and-place operations toward pick-and-throw paradigms in industrial applications.

In addition, this thesis addresses robotic throwing with articulated manipulators, for which approaches analogous to those used with delta robots are typically infeasible due to increased kinematic complexity and control challenges.

To address this limitation, a novel algorithm is proposed that systematically identifies stable manipulator configurations conducive to repeatable and successful throws at constant joint velocity during release. The approach employs a heuristic search strategy to manage the sensitivity and nonlinear dynamics inherent in throwing motions.

Experimental results demonstrate the existence of such favorable configurations on a 6-DoF robotic arm and show that the proposed method provides a practical means of improving throwing consistency and success rates. Beyond its immediate practical benefits, the algorithm establishes a foundation for future research on robust throwing strategies for articulated manipulators.

Overall, this thesis contributes both a consolidation of existing knowledge and new methods for enhancing robotic throwing, thereby supporting broader and more robust real-world deployment.

1.4 Thesis Organization

This thesis is structured as follows. First, related work in robotic throwing is reviewed. Both prehensile and non-prehensile approaches are discussed, with particular emphasis on the challenges posed by noisy and uncertain gripper release dynamics in prehensile throwing.

Next, the delta robot platform is introduced. Its forward and inverse kinematics are presented in detail, followed by the development of a dynamic model for the MuJoCo simulator. This modeling effort addresses the limited availability of openly accessible simulation tools for this class of robots. Building on this foundation, the proposed throwing method for the delta robot is presented and evaluated in simulation.

Subsequently, the experimental setup based on an articulated robotic arm is described, using a UR5 manipulator as a representative platform. An algorithm for generating robust constant-velocity throws is introduced, and simulation as well as real-world experiments on a UR5 robot are presented.

Finally, the concluding chapter summarizes the main findings of the thesis, discusses the limitations of the proposed methods, and outlines directions for future work aimed at improving the robustness, adaptability, and practical deployment of robotic throwing systems.

[2] In [Urrea and Kern, 2025], a consistent growth of robotics deployment in industry is indicated.

Chapter 2

Related Work

This chapter reviews prior research on robotic throwing, with a particular focus on articulated robotic arms and delta robots, as well as recent efforts to equip mobile platforms—such as legged robots [Ma et al., 2025] and underactuated excavation machines [Werner et al., 2024] with throwing capabilities.

These research directions illustrate both the diversity of robotic throwing paradigms and the increasing interest in extending these skills beyond controlled laboratory environments.

2.1 Non-Prehensile Throwing

Early research on robotic throwing predominantly focused on non-prehensile manipulation strategies.

In [Aboaf et al., 1988], a learning-based approach is proposed for ball throwing using a catapult-like mechanism, as well as a planar two-joint system for ball juggling. In [Senoo et al., 2008], a human-inspired throwing motion is implemented on a robotic arm, where the ball is initially grasped using three fingers and then transitions to a loosely controlled two-finger contact before release. Building on the throwing approach in [Senoo et al., 2008], the work in [Kizaki and Namiki, 2012] adapts the motion for juggling tasks by incorporating a catching phase to enable repeated throws. More recent work in [Ploeger et al., 2021] employs reinforcement learning together with a funnel-shaped end-effector to capture and accelerate the ball for juggling. The same setup, but with two arms and only in simulation, has been used in [Ploeger and Peters, 2022] to juggle with up to 17 balls.

For non-prehensile throwing, several studies have explored funnel-shaped or catapult-style interfaces which, when combined with appropriate throwing trajectories, passively constrain object motion during acceleration. These designs enable smooth momentum transfer without requiring dexterous grasping. However, such approaches typically assume specific object geometries or contact conditions, limiting their applicability to arbitrary objects in unstructured environments.

2.2 Prehensile Throwing with Delta Robots

In [Raptopoulos et al., 2020], a PaT approach is evaluated in comparison to PaP using a Delta robot in an urban waste-sorting scenario. An ABB IRB 360 equipped with a suction gripper is employed, and the results indicate a reduced cycle time when replacing PaP with a PaT approach. In [Hassan et al., 2022], a methodology for determining minimum-time S-curve throwing trajectories for a T3KR parallel robot is presented and compared against [Raptopoulos et al., 2020] and PaP. While shorter cycle times are reported for both comparisons, the achieved throwing accuracy and release dynamics are not evaluated.

Another sorting scenario using a Delta robot is presented in [Louette et al., 2024], where reinforcement learning is employed to improve throwing trajectories and release timing for sorting scrap metal parts into bins.

Although these works demonstrate the feasibility of prehensile throwing with Delta robots, several limitations remain. In [Raptopoulos et al., 2020], the throwing trajectory generation is not described in detail, and neither the bin size nor the achieved accuracy are reported. Similarly, in [Hassan et al., 2022], neither the target size nor the achieved success rate is explicitly specified. Finally, the evaluation in [Louette et al., 2024] considers relatively large target areas (25×70 cm) and reports a success rate below 90% in simulation. Moreover, release dynamics are not explicitly modeled or analyzed in these works, with the exception of [Louette et al., 2024], which accounts only for the full gripper opening time and does not provide a detailed characterization of the release process.

To address these limitations, a robust baseline for prehensile throwing with Delta robots is presented in Chapter 3.

2.3 Prehensile Throwing with Articulated Robotic Arms

Robotic throwing with robotic arms has been studied extensively over the past decades, with approaches ranging from analytical motion planning to data-driven learning methods. Early work largely focused on planar or low-degree-of-freedom manipulators and targeted landing positions rather than full landing poses. More recent research has expanded the operational design domain by considering dexterous throwing motions, a wider variety of objects, and increasingly unstructured environments.

Sintov and Shapiro [Sintov and Shapiro, 2015] propose a stochastic offline motion planning algorithm for robotic object throwing under full kinodynamic constraints. Their method parameterizes the throwing motion using a high-dimensional, time-invariant vector that defines the end-effector trajectory up to the release point, followed by ballistic free-flight of the object. Kinodynamic, workspace, and actuator constraints are formulated as time-varying feasible subspaces in the parameter space, and a stochastic sampling-based search is used to identify feasible and optimal throw trajectories with probabilistic completeness guarantees. The approach is validated on a planar 3R manipulator and demonstrates accurate throws under tight constraints. However, the method is designed for offline planning and is demonstrated only on low-degree-of-freedom systems, limiting its applicability to high-speed or highly dexterous industrial manipulators.

Learning-based approaches address some of the modeling limitations inherent to analytical planning. In [Kim et al., 2021], a Quality-Diversity (QD) framework is applied to a Baxter robot to learn a diverse repertoire of throwing behaviors. By densely exploring the joint-space action domain and storing state–action–outcome mappings, the method produces multiple viable solutions for the same target, improving robustness to execution variability. However, such approaches typically require extensive exploration and do not explicitly enforce dynamic feasibility.

Hybrid learning-and-model-based systems have shown particular promise. TossingBot [Zeng et al., 2020] jointly learns grasping and throwing from RGB-D observations in a self-supervised manner, combining a physics-based ballistic model with a neural network that predicts grasp-conditioned residuals on the release velocity. This formulation enables generalization to a wide range of objects while compensating for unmodeled effects, such as grasp offsets and aerodynamic disturbances, achieving a success rate of 88.6% on a 25×15 cm target bin for seen objects. However, achieving this level of performance required thousands of trial-and-error experiments on the real robot.

TossNet [Chen et al., 2024] relies on proprioceptive sensing to learn an autoregressive model that predicts object landing outcomes from joint motion and wrist force/torque measurements. While this approach demonstrates strong generalization and avoids reliance on external sensors, the resulting throwing motions remain limited in rotational diversity, constraining achievable landing poses.

Beyond stationary targets, throwing toward moving targets introduces additional challenges in timing, feasibility, and online adaptation. In [Liu et al., 2022], a fast feasibility-based framework is proposed in which object flight dynamics are encoded using a learned Backward Reachable Tube, while robot kinematic constraints are handled via an offline velocity hedgehog structure. This decoupling enables sub-millisecond online queries and rapid replanning under disturbances. Building on this idea, [Liu and Billard, 2024] explicitly addresses release uncertainty caused by gripper dynamics by approximating corrective accelerations during gripper opening. This approach significantly improves hit rates and remains adaptively suited for moving targets and robot-system disturbances due to its fast computation.

To achieve both a desired landing position and orientation, [Liu et al., 2025] varies three parameters—velocity, pitch, and damping (acceleration)—to identify correlations that can be used to control the throw outcome.

Overall, existing approaches reveal a clear trade-off between model-based methods, which offer explicit feasibility guarantees but suffer from modeling inaccuracies and computational cost, and learning-based methods, which handle complex dynamics but require large amounts of data and often lack formal guarantees. In particular, accurate modeling and control of the release process in prehensile throwing remains a key challenge, especially for dexterous manipulators operating at high speed. [1]

The work in [Liu et al., 2022] presents a fast method for finding feasible throwing configurations. In combination with [Liu and Billard, 2024], it also addresses release uncertainty. However, the required accelerations during release place significant demands on robotic hard-

[1] Y. Liu and A. Billard address the analysis and modelling of the release process in [Liu and Billard, 2026].

ware and trajectory tracking, which may reduce throwing performance and impose high hardware requirements.

2.4 Expanding Horizons: Mobile Platforms and Heavy Machinery

Beyond fixed-base manipulators, robotic throwing has increasingly been explored on mobile platforms. Quadruped robots, in particular, have been investigated for whole-body throwing tasks that require coordinated motion planning, balance maintenance, and dynamic stability during the release phase. These systems introduce additional constraints related to locomotion, whole-body dynamics, and contact stability. An approach for throwing with quadruped robots is presented in [Ma et al., 2025], which also demonstrates the successful integration of the tube acceleration method from [Liu and Billard, 2024] to improve robustness against release uncertainty.

Another direction involves underactuated excavation machines equipped with throwing capabilities for material handling tasks [Werner et al., 2024]. These applications extend robotic throwing to large-scale, highly unstructured environments.

2.5 Summary and State of the Art

The literature shows a progression from early non-prehensile approaches to prehensile throwing systems that emphasize robustness, object generality, and adaptive control. While non-prehensile throwing benefits from simpler release dynamics, it is often limited to specific object classes and highly controlled settings.

Prehensile throwing allows greater control over projectile position, velocity, orientation, and angular momentum, but introduces significant challenges related to gripper release dynamics and high-speed nonlinear motion. Recent learning-based and hybrid approaches have improved robustness, yet accurate modeling of release physics and consistent evaluation across diverse scenarios remain open problems.

These gaps motivate the approaches presented in this thesis, which focus on enhancing release robustness and overall throwing performance on widely used robotic platforms.

Chapter 3

Prehensile Throwing with Delta Robots

This chapter introduces *prehensile throwing* as a dynamic manipulation strategy for delta robots. Prehensile throwing is motivated by comparison with standard PaP manipulation. Conventional PaP is limited to the directly reachable workspace, requiring the robot to follow a continuous motion from the pickup to the placement location. In contrast, dynamic throws exploit ballistic trajectories to transport objects beyond the static reach of the robot and likely more quickly, thus extending the effective workspace and increasing throughput.[1]

This comparison highlights a clear trade-off. While PaP provides predictable behavior and simpler control, it naturally limits both cycle time and spatial reach. Prehensile throwing, on the other hand, separates object transport from continuous actuation during flight, allowing for higher dynamic performance—but it also makes the system more sensitive to release timing, model errors, and accuracy. As a result, achieving high-speed manipulation with controlled dynamics requires carefully combining well-designed trajectories, and precise release control.

To evaluate throwing with delta robots in this chapter, first the relevant robot characteristics, including kinematics, dynamics, and operational limits as well as the implementation in the simulation are introduced.

Then the task setup is outlined by describing the payload, outlining the assumptions for grasping, specifying the release conditions, and defining the performance metrics used to evaluate success. These metrics focus on how accurately the object lands, how repeatable the throws are, the peak loads on the actuators, and the overall duration of the throw.

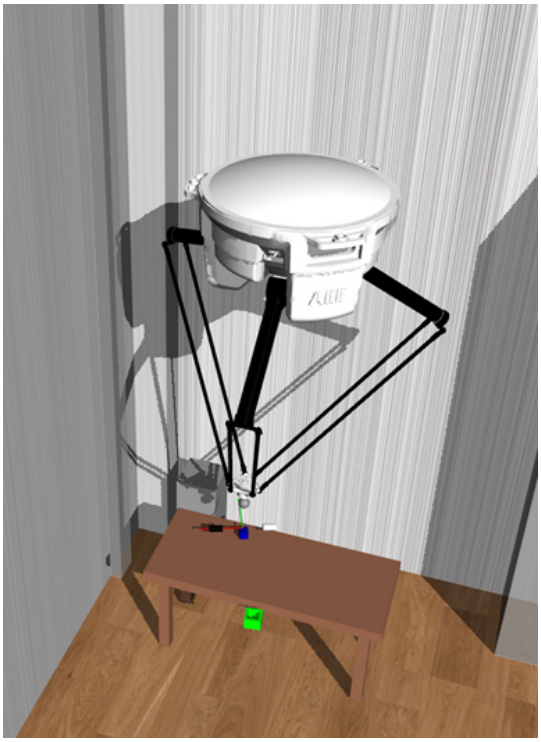
To achieve robust throwing results, a method for generating trajectories that are resilient to release uncertainty is derived. The full control pipeline is presented, covering everything from trajectory generation to release timing.

Finally, the proposed framework is evaluated in MuJoCo simulation [Todorov et al., 2012], the results are analyzed with respect to accuracy, robustness, and actuator demands, and the findings are summarized.

[1] The potential of dynamic throwing with delta robots has already been demonstrated in prior work, e.g., [Raptopoulos et al., 2020, Hassan et al., 2022, Louette et al., 2024].



(a) ABB IRB 360 FlexPicker



(b) MuJoCo simulation environment

Figure 3.1: Experimental setup used in this work. (a) ABB IRB 360 FlexPicker [ABB Robotics, 2026]. This robot is a smaller variant of the model used in this study. (b) MuJoCo simulation environment used for evaluation. The blue cube represents one of the objects to be thrown, and the green box indicates the target location. Objects are randomly placed on the table. The rod mechanism for rotation is not explicitly modeled. However, the corresponding joint is implemented.

3.1 Delta Robot Kinematics and Dynamics

3.1.1 Overview

The *Delta Robot* [Rey and Clavel, 1999] is a parallel manipulator widely used in industry for high-speed PaP operations. Its classical architecture consists of three identical kinematic chains connecting a fixed base to a mobile platform. Each chain consists of an *actuated upper arm* and a *passive parallelogram linkage*, which constrains the orientation of the mobile platform. Consequently, the end effector exhibits purely translational motion, while maintaining a fixed orientation.

The original delta robot design further includes a rotational joint at the moving platform, providing an additional degree of freedom about the vertical (z) axis. Over the years, several architectural variations have emerged, including:

- Configurations with more than three kinematic chains,

- Designs employing linear actuators in place of rotary upper arms,
- Mechanisms introducing additional joints between the platform and end effector to decouple translational and rotational motion.

In this work a simulated *ABB IRB 360* [ABB Robotics, 2026] with four degrees of freedom is considered, as shown in Fig. 3.1. The IRB 360 closely follows the classical delta architecture and is widely deployed in industrial environments. As a parallel manipulator, it offers several structural advantages over serial robotic arms:

- Actuators mounted on the fixed base keep the moving structure lightweight, resulting in low effective inertia and very high end-effector accelerations.
- Closed-chain topology provides high structural stiffness and favorable load distribution.

These characteristics yield excellent dynamic performance and repeatability, making the platform particularly suitable for tasks requiring rapid motion, high acceleration, and precise positioning.

As described above typical delta robot uses three rotary motors at the base to control the angular positions of the upper arms. The resulting end-effector motion emerges from geometric constraints imposed by the passive linkages connecting the arms to the mobile platform. In the IRB 360, an additional actuator—also base-mounted— is mechanically coupled to the end effector via a rod mechanism, enabling controlled rotation about the vertical axis. This fourth degree of freedom allows tool orientation without compromising high-speed translational capabilities.

This section presents the kinematic and dynamic modeling of the delta robot, including *inverse kinematics*, *forward kinematics*, and *Jacobian computation*.[2] The modeling approach follows Clavel's original geometric construction [Rey and Clavel, 1999].[3] Forward and inverse kinematics are implemented explicitly, as widely used robot description formats such as MJCF and URDF offer only limited support for modeling closed-chain mechanisms.[4]

3.1.2 Geometric Model

The delta robot consists of three identical kinematic chains connecting a fixed base to a moving end-effector platform. Each chain comprises:

- An *upper arm* of length L_u, actuated by a revolute joint at the base.
- A *lower arm* of length L_l forming a parallelogram linkage with the moving platform.

The parallelogram linkage constrains the orientation of the end effector, ensuring purely translational motion. The actuated joints are evenly spaced at 120° around the vertical Z-axis. The base and end-effector platform are modeled as circles with radii r_b and r_p, respectively.

A base coordinate frame is defined at the center of the base, with:

[2] For the throwing method developed later in this chapter, only inverse kinematics are strictly required. Forward kinematics and Jacobian computations are included for completeness and validation.

[3] Subsequent extensions include e.g. methods based on conformal geometric algebra (CGA) [Hadfield et al., 2020].

[4] In contrast, the Simulation Description Format (SDF) natively supports closed-chain structures, enabling libraries such as Pinocchio to compute forward and inverse kinematics as well as rigid-body dynamics.

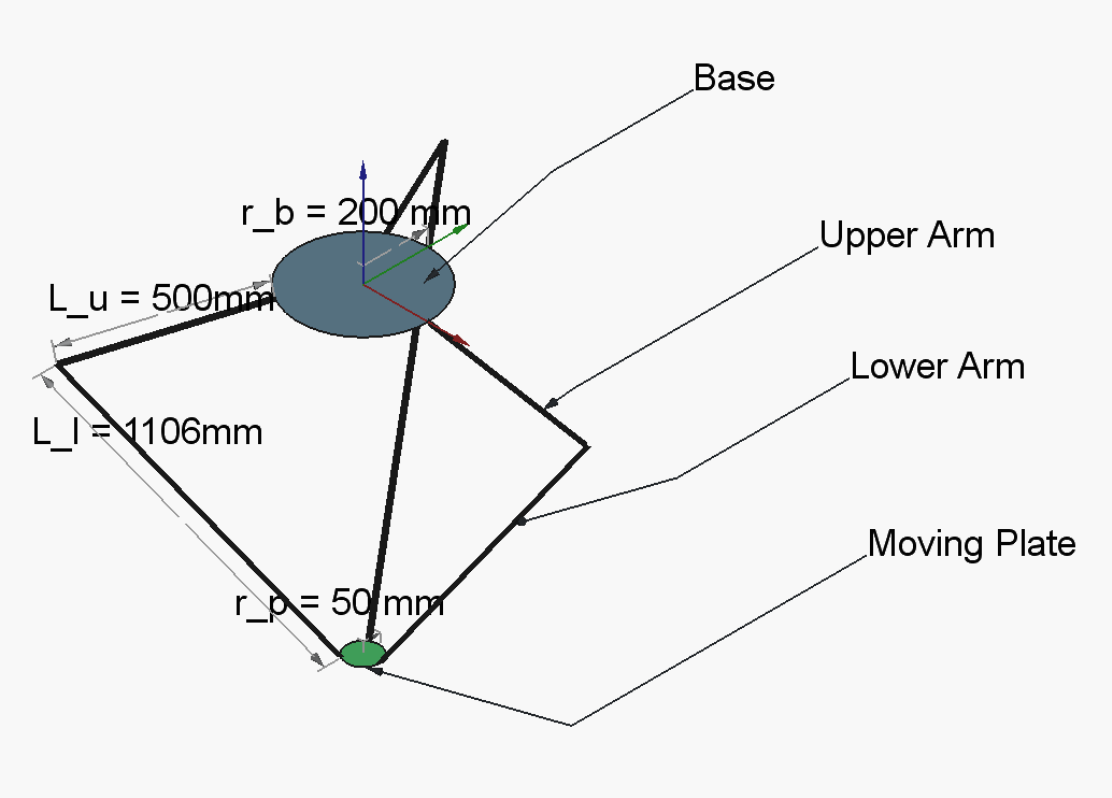


Figure 3.2: Schematic of the delta robot showing geometric parameters used in the simulation model. The parallelogram structure of the lower arms is omitted for clarity.

- Z-axis pointing upwards,
- X-axis aligned with the first kinematic chains.

The joint limits for the upper arms are set to $[-50°, 85°]$,[5] where an angle of $0°$ corresponds to the arm being aligned with the xy-plane of the base. Positive angles increase as the respective arm moves downward.

The geometric parameters and workspace dimensions used in the simulation model are based on the ABB IRB 360 specifications [ABB Robotics, 2026].

Fig. 3.2 illustrates the delta robot geometry, highlighting the arrangement of the base, upper and lower arms, and the end-effector platform.

3.1.3 Inverse Kinematics

The inverse kinematics problem consists of computing the upper-arm angles relative to the base,

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]^T,$$

[5] As the specifications [ABB Robotics, 2026] only provide Cartesian workspace data, the joint constraints were estimated.

required to achieve a desired end-effector position

$$\mathbf{p} = [x, y, z]^T.$$

Owing to the symmetry of the delta robot, the inverse kinematics can be solved independently for each kinematic chain. By rotating the respective upper arm into the XZ-plane[6], the problem reduces to determining the intersection of two circles.

- A circle centered at
$$\mathbf{C}_1 = (r_b, 0), \qquad r_1 = L_u,$$
representing the locus of the upper arm.

- A circle centered at
$$\mathbf{C}_2 = (x + r_p, z), \qquad r_2 = \sqrt{L_l^2 - y^2},$$
representing the projected reachable locus of the lower arm in the XZ-plane.

If the circles do not intersect, no feasible joint configuration exists. A visualization of this geometric construction is shown in Fig. 3.3.

For valid configurations, the intersection point (x_i, z_i) is computed using standard circle–circle intersection formulas. The corresponding motor angle is then obtained as

$$\theta_i = \arctan 2(-z_i,\ x_i - r_b)$$

This procedure is repeated for the remaining two chains after rotating the base by $\pm 120^\circ$ about the Z-axis.

3.1.4 Forward Kinematics

Forward kinematics computes the end-effector position $\mathbf{p}$ from a given set of joint angles $\boldsymbol{\theta}$. First, the positions of the three elbow joints are computed in the base frame. For the first kinematic chain, this yields

$$\mathbf{e}_1 = \begin{bmatrix} L_u \cos\theta_1 + r_b - r_p \\ 0 \\ -L_u \sin\theta_1 \end{bmatrix}.$$

The elbow positions of the remaining chains, $\mathbf{e}_2$ and $\mathbf{e}_3$, are obtained by rotating this expression about the Z-axis by $+120^\circ$ and -120°, respectively.

The end-effector corresponds to the intersection point of three spheres centered at $\mathbf{e}_i$ with radius L_l (the lower arm length). This intersection is computed using a standard trilateration formulation. First, an orthonormal basis is constructed:

$$\mathbf{e}_x = \frac{\mathbf{e}_2 - \mathbf{e}_1}{\|\mathbf{e}_2 - \mathbf{e}_1\|}, \qquad i = \mathbf{e}_x^T(\mathbf{e}_3 - \mathbf{e}_1),$$

[6] In this setup, the first arm is already aligned with the XZ-plane.

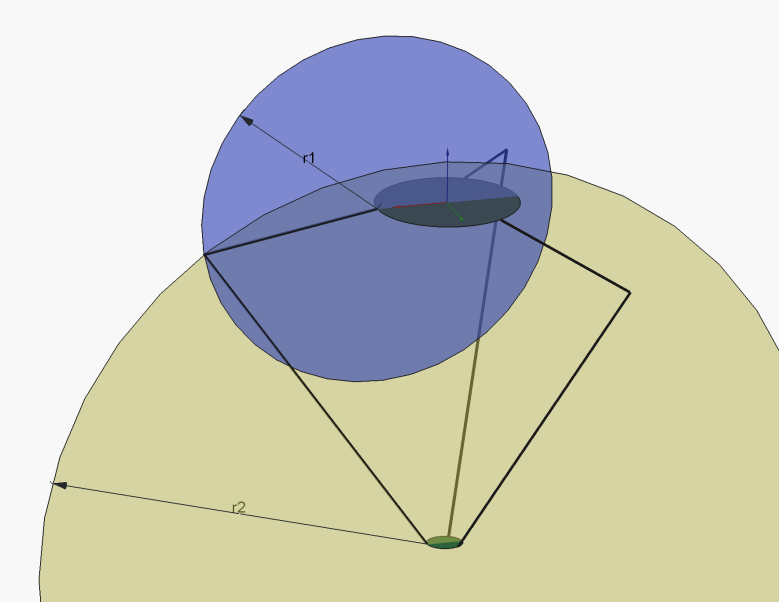


Figure 3.3: Geometric construction used for inverse kinematics computation. The circles illustrate the solution for a single kinematic chain; the remaining two joints are obtained by rotating the base frame accordingly and applying the same procedure.

$$\mathbf{e}_y = \frac{\mathbf{e}_3 - \mathbf{e}_1 - i\mathbf{e}_x}{\|\mathbf{e}_3 - \mathbf{e}_1 - i\mathbf{e}_x\|}, \qquad \mathbf{e}_z = \mathbf{e}_x \times \mathbf{e}_y.$$

Let

$$d = \|\mathbf{e}_2 - \mathbf{e}_1\|, \qquad j = \mathbf{e}_y^T(\mathbf{e}_3 - \mathbf{e}_1).$$

The coordinates of the intersection point in this local basis are

$$\xi = \frac{d}{2}, \qquad \eta = \frac{i^2 + j^2 - 2i\xi}{2j}, \qquad \zeta = \sqrt{L_l^2 - \xi^2 - \eta^2}.$$

The end-effector position in the base frame is then

$$\mathbf{p} = \mathbf{e}_1 + \xi\mathbf{e}_x + \eta\mathbf{e}_y + \zeta\mathbf{e}_z.$$

The positive sign of ζ selects the physically valid downward-facing configuration, and the resulting end-effector position is transformed into the world frame using the robot's base pose.

3.1.5 Numerical Jacobian

To relate joint velocities to end-effector linear velocities, the Jacobian matrix is computed numerically using central finite differences. For each joint variable q_i, a small perturbation ε is applied:

$$\mathbf{J}_i = \frac{\mathbf{p}(q_i + \varepsilon) - \mathbf{p}(q_i - \varepsilon)}{2\varepsilon},$$

where $\mathbf{p}(\cdot)$ denotes the forward kinematics mapping.

Since only translational motion is considered, the Jacobian reduces to a 3×3 matrix. The resulting Jacobian is expressed in the robot base coordinate frame and is used for velocity control and differential motion analysis.

3.1.6 Workspace and Singularity Analysis

In this work, the workspace of the delta robot isn't explicitly modeled. Instead, workspace constraints are implicitly enforced through joint limits and by validating that the cartesian limits specified in [ABB Robotics, 2026] aren't violated in the throwing tasks described in Section 3.5.3.

Singular configurations, in which the Jacobian matrix loses rank and may result in loss of control authority or unbounded joint velocities, are not explicitly analyzed. However selected joint limits and task definition are designed to prevent the robot from operating near such singularities during the considered motions.

3.1.7 Dynamic Modelling

An explicit analytical dynamic model of the delta robot is not derived in this work. Instead, the dynamic behavior of the ABB IRB 360 delta robot is modeled through simulation in the MuJoCo [Todorov et al., 2012] physics engine. MuJoCo provides a physically consistent environment that accounts for inertial, gravitational, and contact effects, enabling realistic dynamic behavior without requiring explicit equations of motion. For the simulation, the openly accessible CAD files from [ABB Robotics, 2026], for the IRB 360 X-1600 FlexPicker robot, were first converted to STL format with FreeCAD. Then the STL files were refined with automated mesh-processing tools in Meshcat, followed by manual correction in Blender. These meshes were then used to describe the robot in MJCF format for the MuJoCo simulation.[7]

The robot is controlled using a proportional–derivative (PD) effort position controller that tracks desired joint positions. The official IRB 360 specification specifies a maximum Cartesian acceleration of 150 $\mathrm{m/s^2}$. For all evaluation tasks, trajectories are generated such that the resulting end-effector accelerations remain well below this limit, ensuring safe and physically realistic motion within the simulated environment.

The gripper is modeled using a sphere and two cylindrical bodies representing the fingers. This simplified gripper is fixed centrally to the moving plate of the robot and points downward, with a distance of 20 cm between the moving plate and the axis connecting the fingers.[8] The gripper opening and closing is simulated as an effort-controlled cylinder, the second cylinder is constrained through a kinematic constraint, both connected to the sphere as prismatic joints. The controller is designed such that the normal force exerted on the finger contact surfaces can

[7] The repository manipulator-mujoco [Ian Chuang, 2026] was used as a basis for the MuJoCo setup. The model of the environment was converted from URDF of TAMS-Group [2025] to MJCF format using urdf2mjcf from [K-Scale Labs, 2026].

[8] The gripper is shown in 3.1b. The black and white cylinders simulate fingers.

be specified to grasp an object, and opening and releasing an object is achieved by linearly decreasing this normal force exerted by the finger contact surfaces.

The MuJoCo setup for the delta robot is shown in Fig. 3.1b.

3.1.8 Summary

The modeling of the delta robot used in this project was described in this chapter. A parallel manipulator that connects a stationary base to a mobile platform via three identical kinematic chains, enabling free motion in cartesian space while keeping orientation constant. Through an additional joint connected to the moving platform, rotation along the vertical axis can be achieved to orient the finger of the gripper.

To provide the functional similarity of a standard 2F gripper while offering an efficient and simplified representation for simulation purposes, a custom two-finger gripper was used, modeled with two cylinders as fingers.

The tasks to evaluate throwing perfomance are defined as such, so that generated trajectories remain feasible within the robot's joint limits, even though the workspace is not explicitly modeled. Similarly, it is not anticipated that singular configurations will occur within the designated task and joint ranges.

The robot's dynamic behavior is captured through simulation in MuJoCo, allowing for quantitative evaluation of joint torques, forces, and other dynamic quantities during task execution as well as realistic, physically consistent system behavior.

3.2 Task Setting

A prehensile throwing cycle is considered in which an object is grasped, accelerated while maintaining a stable grasp, and released such that it reaches a specified target position after a free-flight phase. The task is defined by a desired landing position and constrained by robot feasibility (workspace, joint limits, velocity/acceleration bounds, and actuator limits) as well as grasp feasibility (no slip and no loss of contact prior to release).

Payload and grasp assumptions During the acceleration phase, the object is assumed to be firmly grasped by the end effector, with the contact maintained by a controlled normal force of the gripper towards the grasped object. Two different payloads are considered in the evaluation, a cube with CoM at grasp position, as well as a box with CoM apart from grasp position (detailed description in Section 3.5.2); aerodynamic effects are neglected.

Motion Generation and Release Let t_0 denote the time at which the object is firmly grasped and ready to be accelerated, and let t_e denote the time at which the robot has completed the throwing motion and is free to return to grasp the next object. A time-parameterized end-effector trajectory is generated over the interval $[t_0, t_e]$ such that the robot reaches a desired release state at time $t_r \in [t_0, t_e]$.

At time t_r, the gripper opening is initiated, and the normal force between the fingers and the object is reduced over a finite release interval Δt_r. After contact is fully lost at time $t_r + \Delta t_r$, the

object enters free flight. From this moment onward, the object motion is modeled using ballistic dynamics under gravity and neglected air drag.

The object position for $t \geq t_r + \Delta t_r$ is given by

$$\mathbf{p}(t) = \mathbf{p}(t_r + \Delta t_r) + \mathbf{v}(t_r + \Delta t_r)\big(t - (t_r + \Delta t_r)\big) + \tfrac{1}{2}\mathbf{g}\big(t - (t_r + \Delta t_r)\big)^2,$$

where $\mathbf{p}(t_r + \Delta t_r)$ and $\mathbf{v}(t_r + \Delta t_r)$ denote the object CoM position and velocity at the instant of full release. The vectors

$$\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}, \qquad \mathbf{v}(t) = \begin{bmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{bmatrix}$$

represent the Cartesian position and velocity of the object center of mass.

The gravity vector is

$$\mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ -g_0 \end{bmatrix}, \qquad g_0 = 9.81 \text{ m/s}^2.$$

Performance Metrics The following criteria are used to evaluate a throw:

1. **Accuracy:** The deviation between the desired target position $\mathbf{p}_t$ and the actual landing position of the thrown object.
2. **Robustness to Gripper Delay:** The sensitivity of the throwing outcome to variations in gripper release timing, evaluated implicitly by testing different payloads and release durations.
3. **Throw Duration:** The duration of the throwing motion, defined as $t_e - t_0$.
4. **Peak Loads:** Maximum joint torques encountered during the throwing motion.

3.3 Problem Formulation

The objective is to compute a dynamically feasible throwing motion for a delta robot such that a grasped object reaches a specified target position. A secondary implicit objective is to minimize the transfer time from an initial position to the target, with the goal of achieving shorter cycle times than a conventional PaP strategy within the reachable workspace. The generated motion must satisfy the robot's kinematic and dynamic constraints, as well as timing constraints imposed by the gripper opening mechanism. In addition, the formulation accounts for uncertainty at release, primarily arising from gripper–payload release dynamics.

3.4 Proposed Algorithm for Robust Delta Robot Throwing

This section describes a robust trajectory-generation algorithm for prehensile throwing with a delta robot. The proposed method delivers a grasped payload to a desired target position while

allowing the object to be released during active robot motion. A key contribution of this work is the explicit modeling of the finite gripper release interval, Δt_r, during which gravitational acceleration acts on the payload as the gripping force is gradually reduced.

In prior work on relta robot throwing, this effect has typically been considered only implicitly, for example through data-driven or learning-based approaches [Raptopoulos et al., 2020, Hassan et al., 2022, Louette et al., 2024]. In contrast, its impact has been identified as significant and explicitly modeled in studies on articulated robotic arms [Liu and Billard, 2024].

The proposed algorithm generates a throwing trajectory that explicitly accounts for the varying non-zero gripper opening time, thereby improving robustness to uncertainties in release timing. The approach is structured into two consecutive phases:

1. *Throw Initialization Phase:* The robot moves the grasped object to a suitable release state, defined by a target position and desired velocity, over the interval $[t_0, t_a]$.

2. *Release Phase:* The gripper opening is executed over the finite interval Δt_r, while the robot continues its motion over $[t_a, t_e]$ to compensate for the delayed loss of contact and ensure accurate object release.

3.4.1 Method

The proposed method first computes a *nominal release state*, defined by the end-effector position and velocity that would achieve the desired landing under idealized release conditions. In this work, the nominal release state is obtained by constraining the end-effector motion to a straight-line trajectory in Cartesian space from the grasp to the release position, with a jerk-limited bang–bang profile. Here, *idealized release* refers to an instantaneous gripper opening followed by ballistic payload motion under gravity, neglecting aerodynamic drag.

To account for the finite and potentially uncertain gripper opening time, the robot trajectory is continued beyond the nominal release instant as such that, at the actual moment of payload separation, the object attains the same position and velocity as it would have under idealized free-flight conditions with instantaneous release at the nominal release state.

Phase I: Nominal Release State Computation

Given a target position $\mathbf{p}_t$ in Cartesian space and an initial end-effector position $\mathbf{p}_0$, a nominal release position $\mathbf{p}_r$ and release velocity $\mathbf{v}_r$ are computed using a simplified projectile model. The model assumes instantaneous object release, constant gravitational acceleration g_0, and negligible aerodynamic effects.

Jerk-limited acceleration phase During Phase I, the end-effector motion is constrained to a straight-line acceleration trajectory in Cartesian space. The blue section in Fig. 3.4 illustrates the resulting Phase 1 trajectory to throw an object grasped at the green dot towards the black dot.

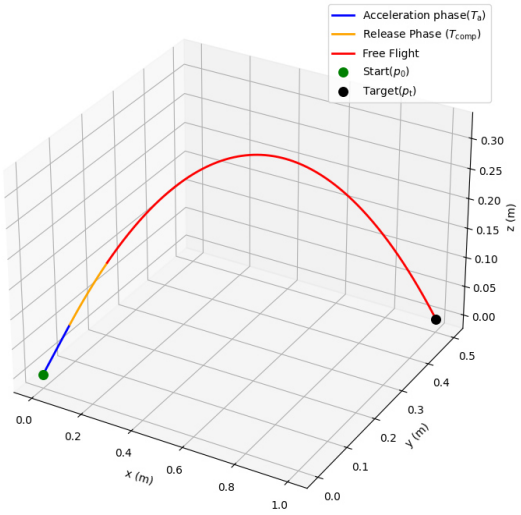


Figure 3.4: Object center-of-mass (CoM) Cartesian trajectory over Phase I, Phase II, and free-flight motion.

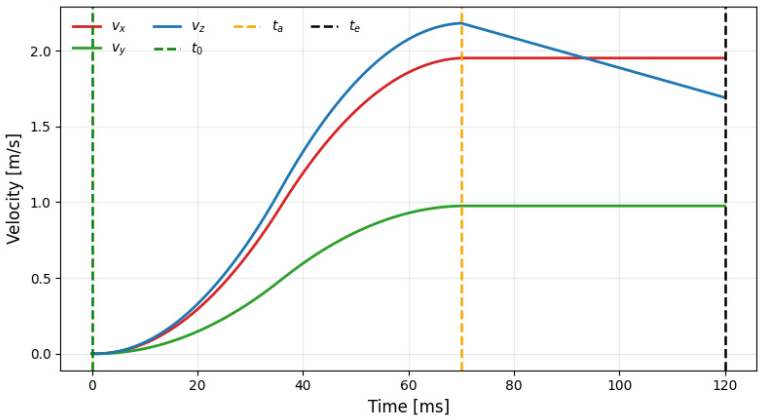


Figure 3.5: Object center-of-mass (CoM) Cartesian velocity profile during throwing, for the same target and parameters as in Fig. 3.4.

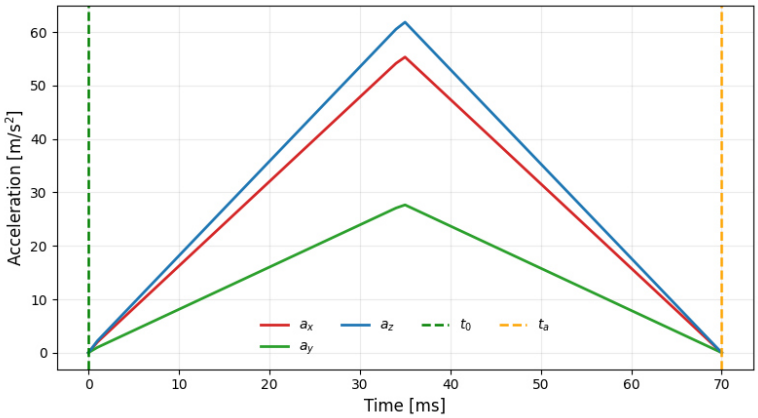


Figure 3.6: Object center-of-mass (CoM) Cartesian acceleration profile during Phase I, for the same target and parameters as in Fig. 3.4.

To calculate the trajectory let $\hat{\mathbf{d}}_h$ denote the unit vector pointing from the initial end-effector position $\mathbf{p}_0$ toward the target $\mathbf{p}_t$ in the horizontal plane,

$$\hat{\mathbf{d}}_h = \frac{[\Delta_x,\, \Delta_y,\, 0]^\top}{\sqrt{\Delta_x^2 + \Delta_y^2}}, \qquad \boldsymbol{\Delta} = \mathbf{p}_t - \mathbf{p}_0.$$

An elevation angle θ defines the three-dimensional throwing direction

$$\hat{\mathbf{d}}(\theta) = \begin{bmatrix} \hat{d}_{h,x} \cos\theta \\ \hat{d}_{h,y} \cos\theta \\ \sin\theta \end{bmatrix}.$$

The end-effector acceleration magnitude follows a jerk-limited bang–bang profile with maximum jerk $j_{\max}$. Let T_j denote the duration of each jerk phase and let $T_a = 2T_j$ be the total duration of the acceleration phase. In Phase I, the *idealized release* is assumed to occur at the end of the acceleration phase, i.e., $t_r = t_a = T_a$ when the initial time is set to $t_0 = 0$. [9]

The Cartesian acceleration is defined as

$$\mathbf{a}(t) = a_{\text{mag}}(t)\, \hat{\mathbf{d}}(\theta), \qquad t \in [0, T_a],$$

where the acceleration magnitude is given by

$$a_{\text{mag}}(t) = \begin{cases} j_{\max}\, t, & 0 \le t \le T_j, \\ j_{\max}\, (T_a - t), & T_j < t \le T_a. \end{cases}$$

An initial elevation angle θ is selected (here 45°) and incrementally increased if no feasible solution is found. For each candidate angle, the jerk time T_j is determined using a bisection

[9] In practice, an additional delay may be introduced to debounce the controller before gripper opening is initiated, resulting in $t_r > t_a$, as discussed below.

search [Quarteroni et al., 2007]. Given a candidate T_j, numerical integration of the acceleration profile yields a release state $(\mathbf{p}_r, \mathbf{v}_r)$, after which the payload follows ballistic flight.

The corresponding flight time t_f is obtained from the vertical motion constraint

$$(p_{r,z} - p_{t,z}) + v_{r,z} t_f - \tfrac{1}{2} g t_f^2 = 0,$$

where the physically valid (non-negative) root is selected. The bisection objective is defined by matching the horizontal distance to the target, computed as the sum of the horizontal displacement during the acceleration phase and the horizontal displacement during the ballistic flight.

Once the nominal release state $(\mathbf{p}_r, \mathbf{v}_r)$ is obtained, the corresponding acceleration trajectory is discretized using a fixed time step Δt. In the implementation, this is achieved by integrating the jerk-limited acceleration profile backward in time from $(\mathbf{p}_r, \mathbf{v}_r)$ toward $\mathbf{p}_0$ using a fourth-order Runge–Kutta (RK4) [Butcher, 2016] scheme. The backward integration is terminated when the position error satisfies

$$\|\mathbf{p} - \mathbf{p}_0\| < 10^{-4}.$$

The resulting number of discrete steps N_a defines the effective acceleration duration, i.e., $t_a \approx N_a \Delta t$. The corresponding joint configurations $\mathbf{q}_k = \mathbf{q}(t_k)$ are then obtained via inverse kinematics as shown in Section 3.1.3 at the discrete time instants $t_k = k\Delta t$, with $k \in \{1, \ldots, N_a\}$. [10] The calculated cartesian velocities in Phase I are shown in Fig. 3.5, and the according accelerations in Fig. 3.6. In both figures, Phase I spans from the green dashed line to the vertical yellow dashed line.

Phase II: Release Compensation Trajectory

In practice, the gripper opening time is non-zero, introducing a delay between the time t_a and the actual payload separation. Phase II compensates for this delay by extending the nominal Cartesian trajectory beyond t_a for an additional duration

$$T_{\text{comp}} = t_e - t_a.$$

The end of Phase II is chosen such that the robot remains in motion until the object has fully separated, i.e.,

$$t_e \geq t_r + \Delta t_r.$$

Optionally, a transition delay Δt_d may be introduced between the phases, before the gripper opening is initiated, to allow the robot to adapt after the high-effort acceleration phase, such that

$$t_r \geq t_a + \Delta t_d.$$

To calculate the trajecory of Phase II, let $\mathbf{p}_r$ denote the nominal release position computed in Phase I (Section 3.4.1), and let $\mathbf{v}_r$ be the corresponding nominal release velocity. The compensation trajectory is then initialized at this nominal release state as

[10] Since $\mathbf{p}$ and $\mathbf{v}$ are defined in Cartesian space and refer to the object's center of mass (CoM), the offset between the end effector and the CoM must be taken into account for inverse kinematics.

$$\mathbf{p}_0^{\text{ext}} = \mathbf{p}_r, \qquad \mathbf{v}_0^{\text{ext}} = \mathbf{v}_r,$$

and is propagated for

$$N_c = \left\lceil \frac{T_{\text{comp}}}{\Delta t} \right\rceil$$

discrete steps using forward Euler integration.

During the extension, the horizontal velocity is held constant while the vertical component is updated under gravity:

$$\mathbf{p}_{k+1,xy}^{\text{ext}} = \mathbf{p}_{k,xy}^{\text{ext}} + \mathbf{v}_{r,xy}\,\Delta t, \qquad v_{k+1,z}^{\text{ext}} = v_{k,z}^{\text{ext}} + g\,\Delta t, \qquad p_{k+1,z}^{\text{ext}} = p_{k,z}^{\text{ext}} + v_{k+1,z}^{\text{ext}}\,\Delta t,$$

for $k \in \{0, \ldots, N_c - 1\}$. The resulting extension is appended to the Phase I trajectory.

As a result, the payload experiences approximately ballistic motion throughout the uncertain release interval, which improves robustness to variations in the gripper opening time.

In Fig. 3.4, the yellow section shows the trajectory of Phase II, while the remaining red section shows the free flight of the object toward the target. The calculated Cartesian velocities and accelerations of Phase II are shown along with Phase I in Figs. 3.5 and 3.6. Phase II spans from the yellow dashed line to the black dashed line.

3.5 Robustness Analysis and Timing

To assess the effectiveness of the proposed two-phase throwing strategy, a series of simulations were conducted using a delta robot model, as described in Section 3.1. For this the simulated experimental setup introduced in Section 3.5.2 is described more detailed below. The performance is evaluated using the metrics introduced in Section 3.2 and formally defined in the following section.

3.5.1 Evaluation Metrics

The throwing performance is quantified using the following metrics:

1. **Accuracy:** The Euclidean distance in the horizontal plane between the desired target position and the actual landing position of the thrown object. Both the initial position $\mathbf{p}_0$ and the target position $\mathbf{p}_t$ are defined relative to the object's center of mass (CoM). The landing position $\mathbf{p}_{\text{landing},xy}$ is defined as the CoM location at the instant of collision with the horizontal target plane.[11]

$$E_{\text{pos}} = \left\| \mathbf{p}_{\text{landing},xy} - \mathbf{p}_{t,xy} \right\|.$$

2. **Robustness to gripper delay:** The sensitivity of the landing error to variations in the gripper release timing. Robustness is evaluated indirectly by varying the release interval

[11] The height of the horizontal target plane is chosen such that the CoM reaches the target height at the instant of collision. This definition assumes negligible object rotation during the throwing motion.

Δt_r, which represents the time required to ensure safe separation between the gripper and the object. No explicit robustness metric is computed, as the proposed trajectory generation method guarantees convergence to the nominal release state independently of the exact release timing, provided that separation occurs before t_e.

3. **Throw duration:** The total time required to execute the throwing motion:

$$T_{\text{throw}} = t_e - t_0.$$

The acceleration duration T_a is reported separately. Since $T_{\text{comp}} = T_{\text{throw}} - T_a$, the durations of both phases are characterized independently.

4. **Peak loads:** The maximum joint torque magnitude observed during the acceleration phase:

$$\tau_{\max}^{\text{acc}} = \max_{t \in [t_0,\, t_a]} \max_{i \in \{1,2,3\}} |\tau_i(t)|,$$

where $\tau_i(t)$ denotes the torque applied at upper-arm joint i of the delta robot.

To separately characterize the significantly lower actuation effort during the subsequent phase, the peak torque is also reported over the interval following the deceleration period, i.e., $[t_a + \Delta t_d,\, t_e]$.[12]

$$\tau_{\max}^{\text{comp}} = \max_{t \in [t_a + \Delta t_d,\, t_e]} \max_{i \in \{1,2,3\}} |\tau_i(t)|.$$

This distinction highlights the reduction in peak loads once the initial high-acceleration phase is concluded.

3.5.2 Experimental Setup

All simulations were conducted using the delta robot model described in Section 3.1. The robot executed throws toward targets located at varying horizontal distances, all at a fixed height equal to the initial vertical position of the object, corresponding to the surface of a table of size 1.05×0.45 m. Targets were modeled as horizontal planes centered at the desired target locations and aligned with the table surface, from which objects were randomly placed[13] and grasped. The robot was centered above the table, with the horizontal base plate located 1.5 m above the table surface.

Both the initial object position $\mathbf{p}_0$ and the target position $\mathbf{p}_t$ were defined relative to the object's center of mass (CoM). Accordingly, object-specific offsets between the grasp point and the CoM were incorporated into the inverse kinematics to ensure consistent release conditions across objects.

The landing position was recorded as the object's CoM at the instant of contact with the horizontal ground plane of the target plane.

[12] The transition interval of duration Δt_d is excluded from this metric due to potentially noisy torque measurements during the phase transition.

[13] Objects were placed within an area of 1.0×0.4 m.

Nominal release states were computed using the Phase I procedure, while the Phase II compensation trajectory was applied long enough to ensure that object separation occurred before t_e.

The resulting joint-space trajectories were executed in MuJoCo using a position–effort PD controller, with an update rate of 1000 Hz for the simulation. Accordingly the trajectory described in the previous Section 3.4.1 where generated with $\Delta t = 1$ ms.

Actuator torques were computed using the full MuJoCo dynamics model, including mass and inertia properties obtained from CAD meshes via mass-equivalent filling.

The gripper was simulated as an effort-controlled left finger, modeled as a cylinder and constrained to a symmetric right finger through a kinematic constraint. For each trial, a nominal grasp normal force was specified, and during release the normal force was reduced approximately linearly inside the interval $\mathcal{I}_r$ (see Fig. 3.8). As such the release duration emerged implicitly from the gripper controller dynamics and the selected grasp force.

3.5.3 Results

The proposed two-phase throwing strategy was evaluated in simulation using the experimental setup described in Section 3.5.2 and the performance metrics defined in Section 3.5.1. Six distinct target locations at varying horizontal positions were considered. All experiments were performed separately for two different objects with distinct geometric and inertial properties:

- **Cube:** A 4cm cube with uniformly distributed mass of 0.1 kg. The cube was grasped at its geometric center, coinciding with its center of mass.
- **Block:** A $4 \times 4 \times 15$ cm rectangular block with a total mass of 0.1 kg. The block's CoM is located 2.5 cm above its bottom face. It was grasped in an upright orientation with the grasp point located 2.5 cm below the top face.

The six target positions used in the experiments are summarized in Table 3.1.

Table 3.1: Target positions (x, y) in meters.

Target	x (m)	y (m)
1	0.3	-0.5
2	0.3	0.0
3	0.3	0.5
4	1.0	-0.5
5	1.0	0.0
6	1.0	0.5

For the cube, the maximum jerk constraint $j_{\max}$ was varied to evaluate robustness under different acceleration profiles. The block was evaluated only under the low-jerk condition because high-jerk throws caused unstable grasps and slippage. To maintain stable grasping for the block, the grasp force was increased to 30 N and the effective release interval Δt_r was extended accordingly, the parameters are shown in Table 3.2.

Table 3.2: Experimental parameter settings for each object and jerk condition.

Condition	$j_{\max}$ (m/s^3)	F_{grasp} (N)	Δt_d (ms)	Δt_r (ms)
Cube (low jerk)	250	10	10	40
Cube (high jerk)	2500	10	10	40
Block (low jerk)	250	15	10	60

Table 3.3 reports the average horizontal distance to each target, computed from the target positions in Table 3.1. This distance is used to interpret timing and accuracy results across targets.

Table 3.3: Average horizontal distance to each target, measured from the random object placement locations over 150 experiments per target.

Target	Distance (m)
1	0.600
2	0.415
3	0.628
4	1.160
5	1.053
6	1.147

Each experimental condition was repeated 50 times to obtain statistically meaningful results; the resulting throwing outcomes are shown in Fig. 3.7.

Throwing Accuracy

Table 3.4 reports the mean and standard deviation of the positional error E_{pos} across all target locations, object types, and jerk conditions.

Table 3.4: Accuracy results (mean ± std) for all targets.

Target	Cube (low jerk)	Cube (high jerk)	Block (low jerk)
1	0.012 ± 0.011	0.026 ± 0.017	0.023 ± 0.008
2	0.019 ± 0.011	0.024 ± 0.010	0.019 ± 0.004
3	0.015 ± 0.012	0.028 ± 0.017	0.024 ± 0.010
4	0.017 ± 0.008	0.028 ± 0.021	0.017 ± 0.010
5	0.023 ± 0.007	0.037 ± 0.014	0.016 ± 0.007
6	0.018 ± 0.008	0.039 ± 0.015	0.017 ± 0.012

The proposed method achieves consistently low landing errors for all targets and both objects. Across repeated trials, the variance in landing position remains small, indicating stable

and repeatable behavior. These results demonstrate that the Phase II compensation trajectory effectively enforces convergence to the nominal release state, rendering the throwing outcome largely insensitive to the exact release instant, provided that object separation occurs before t_e.

Table 3.5 presents the maximum positional error observed across all trials. The maximum error remains below 0.09 m for all conditions, with the largest deviations occurring under the high-jerk cube condition.

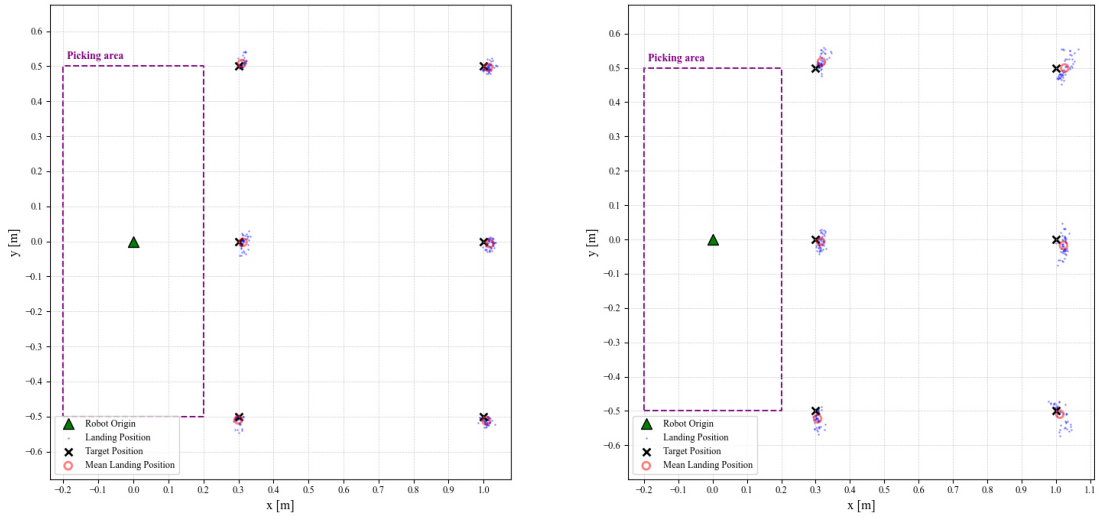
Table 3.5: Maximum positional error $E_{\text{pos,max}}$ for all targets.

Target	Cube (low jerk)	Cube (high jerk)	Block (low jerk)
1	0.047	0.068	0.044
2	0.041	0.044	0.029
3	0.047	0.064	0.044
4	0.038	0.074	0.044
5	0.035	0.075	0.035
6	0.039	0.085	0.046

Nevertheless, both mean and maximum errors increase under higher jerk and when the center of mass is offset from the grasp point. It should be noted that the simulated gripper may exhibit unrealistic behavior, which can introduce additional noise into the results.

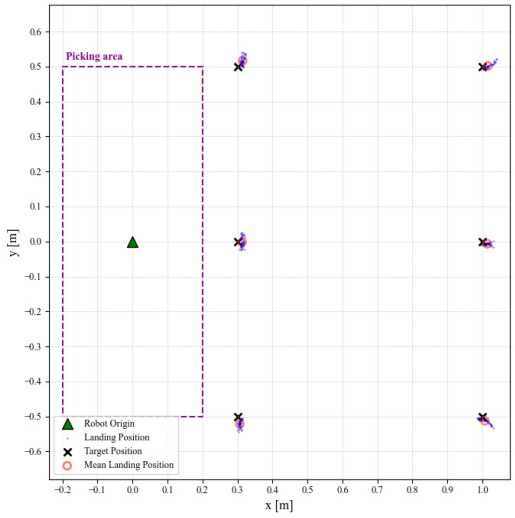
Robustness to Gripper Release Timing

Robustness to gripper delay was evaluated by implicitly varying the effective release interval Δt_r and by changing the grasp force. The linearly decreasing normal force during release is shown in Fig. 3.8. Across all trials, no significant correlation between release timing and landing error was observed, supporting the design objective that the object reaches a consistent release state independent of the precise release timing.



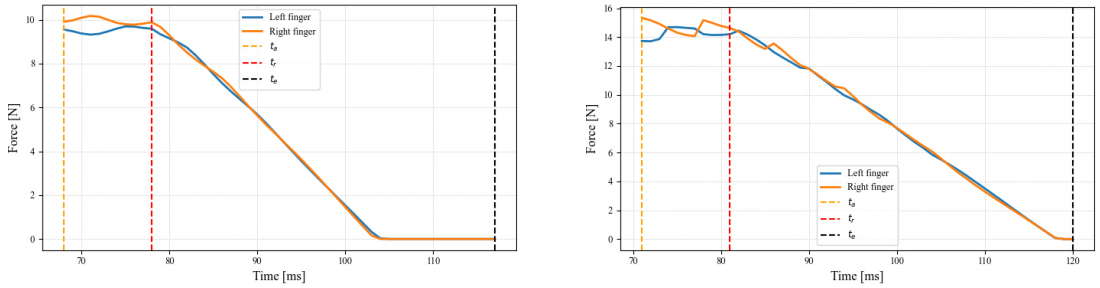
(a) Cube, low jerk

(b) Cube, high jerk



(c) Box, low jerk

Figure 3.7: Comparison of throwing accuracy for different object types and jerk settings. (a) Cube with low jerk, (b) cube with high jerk, and (c) box with low jerk. Black crosses indicate target positions, blue dots represent individual landing positions, red circles denote mean landing positions, and the green triangle marks the robot origin.



(a) Cube, commanded grasp force 10 N.

(b) Stick, commanded grasp force 15 N.

Figure 3.8: Measured gripper normal force during Phase II of the throwing motion. (a) Cube with a commanded grasp force of 10 N. (b) Stick with a commanded grasp force of 15 N.

Throw Duration

Table 3.6 summarizes the mean throw duration T_{throw} and the acceleration duration t_a for each target, object type, and jerk condition.

Table 3.6: Throw duration per target (mean ± std) in ms.

Target	Cube (low jerk)		Cube (high jerk)		Block (low jerk)	
	T_{throw}	T_a	T_{throw}	T_a	T_{throw}	T_a
1	220	170 ± 19	110	60 ± 6	247	177 ± 20
2	208	158 ± 15	103	53 ± 4	230	160 ± 11
3	222	172 ± 23	110	60 ± 7	247	177 ± 19
4	260	210 ± 8	120	70 ± 3	281	211 ± 6
5	255	205 ± 6	118	68 ± 2	274	204 ± 6
6	259	209 ± 8	120	70 ± 3	280	210 ± 8

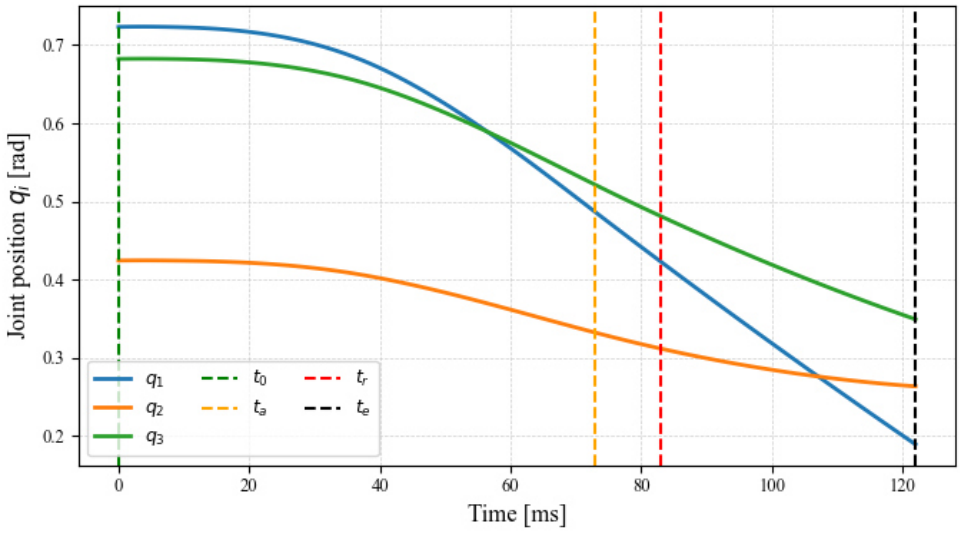
The total throw duration T_{throw} depends on the target distance, the maximum jerk, and the chosen timing parameters Δt_d and Δt_r. The acceleration duration t_a is primarily determined by the maximum jerk and the target distance, while the total duration is given by

$$T_{\text{throw}} = t_a + \Delta t_d + \Delta t_r.$$

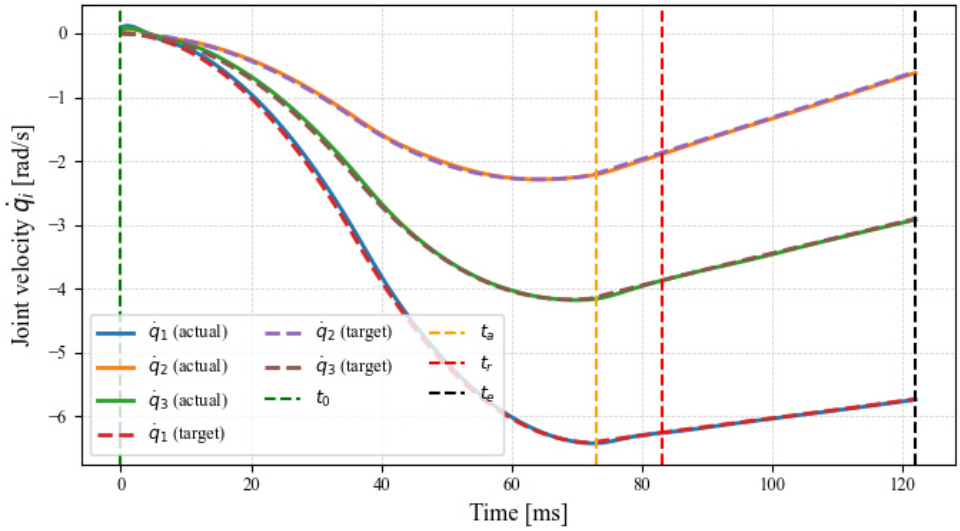
Peak Joint Loads

Table 3.7 reports the peak joint torque magnitudes during the acceleration phase, $\tau_{\text{max}}^{\text{acc}}$, and during the compensation phase, $\tau_{\text{max}}^{\text{comp}}$, for all targets, objects, and cube jerk conditions.

As expected, the highest torque demands occurred during the acceleration phase. Peak torques during the interval $[t_a + \Delta t_d, t_e]$ were significantly lower across all joints and target distances, validating the two-phase structure and confirming that the release phase can be executed under substantially reduced actuation effort.



(a) Joint positions



(b) Joint velocities

Figure 3.9: Joint positions (a) and velocities (b) during the throwing motion for Target 6 using the cube under high jerk conditions, shown from t_0 to t_e.

Table 3.7: Peak joint loads (mean ± std) for all targets.

Target	Cube (low jerk)	Cube (high jerk)	Block (low jerk)
	$\tau_{\max}^{\text{acc}}$ *(Nm)*		
1	42.9 ± 5.8	109.3 ± 7.9	41.5 ± 1.8
2	43.4 ± 2.8	117.3 ± 9.7	43.2 ± 1.4
3	43.6 ± 6.0	113.1 ± 10.2	41.0 ± 2.8
4	59.1 ± 4.8	162.4 ± 3.6	56.3 ± 1.2
5	59.9 ± 3.3	167.8 ± 1.2	57.7 ± 1.4
6	60.0 ± 5.3	163.1 ± 3.4	57.0 ± 1.9
	$\tau_{\max}^{\text{comp}}$ *(Nm)*		
1	4.7 ± 0.7	4.4 ± 0.8	5.2 ± 0.9
2	4.7 ± 0.7	4.0 ± 0.8	5.4 ± 0.7
3	4.6 ± 0.7	4.8 ± 0.7	5.1 ± 1.0
4	4.9 ± 1.0	6.3 ± 2.0	5.5 ± 1.7
5	4.4 ± 1.1	5.4 ± 0.9	5.4 ± 1.8
6	4.8 ± 1.2	6.8 ± 1.9	5.6 ± 2.0

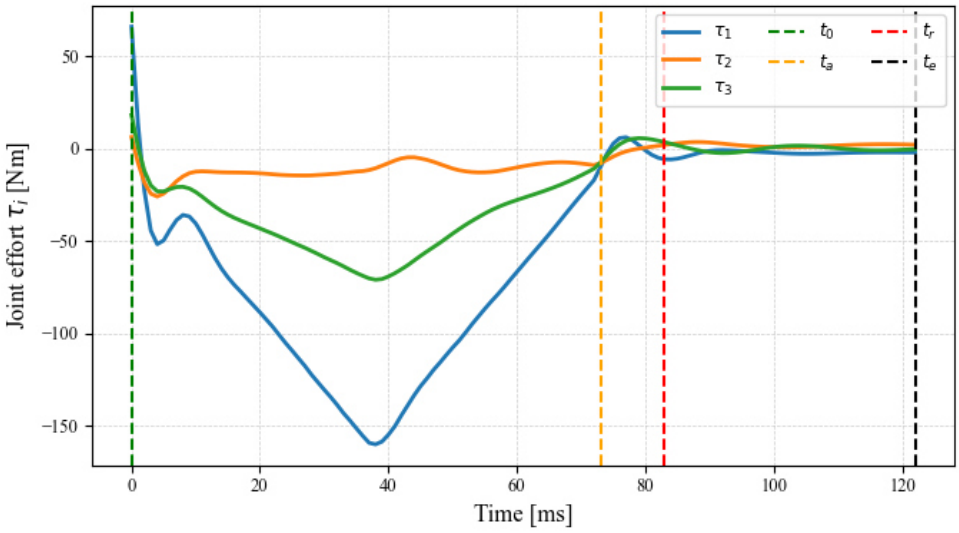


Figure 3.10: Joint efforts during the throwing motion for Target 6 using the cube under high jerk conditions, shown from t_0 to t_e.

3.6 Summary

Overall, the results demonstrate that the proposed two-phase throwing strategy achieves accurate and repeatable throws across multiple targets locations and object types. The method exhibits strong robustness to gripper release timing, and a clear reduction in actuator loads during release. Compared to PaP approach, the proposed strategy is promising to achieve significantly shorter execution times despite operating at lower accelerations.

The introduction of a low-effort post-acceleration phase likely reduces mechanical stress and mitigates the excitation of structural oscillations, thereby contributing to improved system longevity. In addition, the shorter cycle times and reduced actuation effort indicate a promising reduction in overall power consumption. These properties make the approach well suited for dynamic throwing tasks with delta robots and highlight the advantages of the delta robot architecture for easy to implement, robust, low-effort throwing trajectory execution, even under release uncertainty.

3.7 Future Work

Several promising directions remain for future investigation. First, the proposed method should be implemented and validated on real hardware to assess its performance under real actuation, sensing, and contact uncertainties. In particular, evaluation in an industrial PaP setting would enable a direct comparison between the proposed PaT method and conventional PaP workflows and help quantify potential improvements in cycle time and energy efficiency.

From a trajectory-generation perspective, Phase I could be modified to terminate with zero horizontal acceleration while maintaining a vertical acceleration of $-g_0$, potentially allowing the transition delay Δt_d to be reduced or eliminated. Furthermore, integrating the energy-aware optimization strategy proposed in [Hassan et al., 2022] may further reduce power consumption and execution time. An analytical characterization of why, where, and under which conditions the actuation effort in Phase II becomes negligible is currently missing and should be addressed in future work. In addition, the influence of the release position and velocity on the required Phase II effort has not yet been systematically evaluated.

The current approach assumes simple ballistic motion without aerodynamic effects. Future work could therefore explore more complex Phase II trajectories, for example by learning object-specific flight models using NeuralODEs, as indicated in [Liu and Billard, 2024]. Finally, while this work considers a two-finger gripper, extending the method to suction-based grippers represents another important direction for higher applicability.

Chapter 4

Prehensile Throwing with Robotic Arms

In the previous chapter, prehensile throwing was investigated using a delta robot, where the problem structure benefited from a highly constrained kinematic architecture, low moving inertia, and predominantly translational end-effector motion. These properties enabled simplified trajectory generation and facilitated the realization of robust throwing motions, allowing the target to be hit by following the ballistic curve with relatively low control effort during the release.

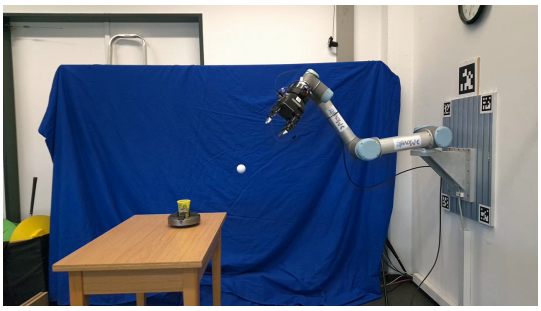
In contrast, extending prehensile throwing to articulated robotic arms introduces a fundamentally different set of challenges. Robotic arms typically possess higher degrees of freedom, larger and configuration-dependent inertial effects, and more complex kinematic mappings between joint space and end-effector motion. While this increased flexibility expands the space of achievable throwing motions, it significantly complicates both kinematic feasibility and dynamic consistency.

This chapter focuses on prehensile throwing with general robotic arms and examines how their kinematic redundancy and dynamic coupling can be explicitly considered to generate reliable release conditions. For this a method is presented to generate throws that remain feasible over a finite time window with constant joint velocities, thereby addressing uncertainty in the release timing. This approach highlights the key differences in planning and execution compared to the delta robot platform, where robust throws could be achieved using intuitive Cartesian trajectories and gravitational effects could be fully compensated over extended time horizons during release.

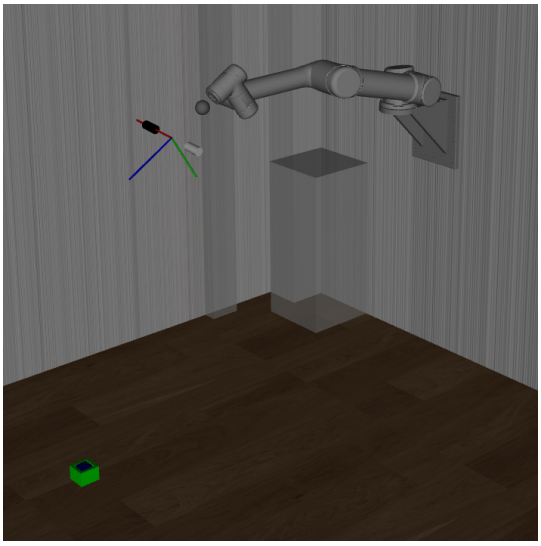
4.1 Kinematic and Dynamic Challenges in Robotic Arm Throwing

Achieving an end-effector motion that approximates a ballistic trajectory during release is non-trivial for articulated robotic arms, [1] in contrast to the delta robot as shown in Chapter 3. Due to the presence of multiple chained rotational joints, simple Cartesian paths generally do not correspond to smooth or dynamically feasible joint-space trajectories. Consequently, the joint space must be coordinated with the Cartesian space. To account for release uncertainty, not just

[1] This has been pointed out, for example, in Liu and Billard [2024].



(a) Real-world experimental setup with a wall-mounted UR5 and Robotiq 3F gripper. The Play-Doh cup on the table serves as the target for assessing throwing performance with the golf ball.



(b) MuJoCo simulation environment with simplified gripper model, target (green), and throwable object (blue). The corner block serves as the object spawning and pick-up location.

Figure 4.1: Real and simulated experimental platform used for evaluation. (a) Physical UR5; (b) corresponding MuJoCo simulation environment with the simplified gripper model, target, and throwable cube.

at a single position but over an interval and at high joint velocities. Additionally, joint constraints in position, velocity, and acceleration must be strictly respected throughout the motion.

One objective is to find a feasible release position and velocity in joint space, while respecting joint limits, workspace constraints, and kinematic feasibility. A difficulty arises from the redundancy and nonlinearity of robotic manipulators. Multiple joint configurations may correspond to the same end-effector pose. However, not all of these configurations are suitable for high-speed throwing due to proximity to joint limits or unfavorable Jacobian conditioning. Singular configurations are particularly problematic, as they can induce excessively large joint velocities when attempting to realize a desired end-effector velocity.

While such issues have been addressed in prior work (e.g., [Liu et al., 2022]), the requirement for robustness in prehensile throwing introduces an additional layer of complexity. Unlike impulsive release mechanisms, prehensile grasps are subject to release uncertainty: the gripper opening is neither instantaneous nor perfectly repeatable. As a result, the object may be released at slightly different times across repeated throws.

This uncertainty implies that the end-effector position and velocity must remain within acceptable bounds over a short continuous time window, rather than at a single idealized instant. In [Liu and Billard, 2024], they address this issue by introducing constant acceleration profiles near the release configuration . However, this conflicts with the proposed objective of maintaining constant joint velocities around the release phase, which might reduce tracking error and improve

performance.

Finally, kinematic feasibility must be preserved throughout the entire throwing motion. This includes ensuring smooth joint trajectories, avoiding self-collisions, and remaining within actuator limits. Together, these requirements significantly restrict the space of admissible throwing motions for articulated robotic arms.

Prehensile throwing with robotic arms presents a tightly coupled set of kinematic and dynamic challenges that fundamentally distinguish it from both quasi-static manipulation and delta robot throwing. On the kinematic side, chained rotational joints, kinematic redundancy, and configuration-dependent Jacobian properties complicate the generation of end-effector motions that reliably achieve the desired release position and velocity. These challenges are further exacerbated by uncertainty in the release process, which necessitates feasibility over a continuous time interval rather than at a single instant.

These constraints motivate the need for throwing strategies that explicitly account for joint-level feasibility, robustness to release timing uncertainty, and actuator limitations.

The following sections build on these observations and introduce an approach for generating dynamically and kinematically feasible throws of grasped objects with articulated robotic arms, with a focus on finding joint configurations with advantageous properties for prehensile throwing.

4.2 Problem Formulation

An articulated robotic manipulator with n degrees of freedom is considered, initially grasping a rigid object at a known end-effector pose. Let $\mathbf{q} \in \mathbb{R}^n$ denote the joint configuration, with $\dot{\mathbf{q}}$ representing the corresponding joint velocities. The end-effector pose is given by $\mathbf{x} = f(\mathbf{q})$, and its velocity by

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}},$$

where $\mathbf{J}(\mathbf{q})$ denotes the manipulator Jacobian.

Given a desired target position in the workspace, the throwing task is defined as reaching a release state from which the object follows a ballistic trajectory and hits the target under the influence of gravity. For a fixed target, this requirement induces a set of admissible end-effector release positions and velocities that satisfy the ballistic constraints.

Unlike idealized formulations that assume instantaneous release, the release process is modeled as occurring over a finite time interval with a noisy start time and duration due to gripper opening dynamics, timing uncertainty and object properties. Consequently, robust throwing performance requires identifying a release configuration for which, under constant joint velocities, the resulting end-effector states over a surrounding time interval still yield a high probability of successfully hitting the target.

The problem is therefore formulated as follows. Starting from an initial grasp configuration $\mathbf{q}_0$, a joint configuration $\mathbf{q}_r$ and a joint velocity $\dot{\mathbf{q}}_r$ must be determined such that:

- The corresponding end-effector position and velocity induce a ballistic object trajectory that reaches the target.

- When propagated forward from $\mathbf{q}_r$ at constant joint velocity $\dot{\mathbf{q}}_r$, the resulting end-effector positions and velocities over a finite time interval continue to result in a successful hit, if the object is released.
- Joint position, velocity, and acceleration limits are respected.
- Actuator torque limits and collision-avoidance constraints are satisfied.

Among all feasible release states $(\mathbf{q}_r, \dot{\mathbf{q}}_r)$, an additional objective is to select one that can be reached from the initial configuration $\mathbf{q}_0$ in minimal time.

This formulation focuses on identifying robust release states in joint space that enable accurate and repeatable throwing despite uncertainty in the release timing.

4.3 Algorithm for Robust Constant-Velocity Throws

This method computes dynamically feasible release states that achieve robust target hits within a specified release-time window $\mathcal{I}_r$ with duration Δt_r, under the assumption of constant joint velocities. Instead of identifying at a single precise release instant, the algorithm searches for release states that reduce landing error within a local temporal neighborhood, improving robustness to release uncertainty.

The procedure operates under a fixed computational budget T_{search}, combining randomized configuration sampling with iterative forward refinement and continuous evaluation of task-level robustness. Candidate release states are progressively improved based on target error and temporal sensitivity, converging toward dynamically consistent and execution-tolerant solutions within the allotted time. In the following sections this method is referred to as ARTS (Arm Robust Throw Search).

For forward and inverse kinematics as well as collision checking, the URDF model of the experimental setup [TAMS-Group, 2025] was adapted and Pinocchio [Carpentier et al., 2015–2018] was used to perform the corresponding kinematic and dynamic computations.

4.3.1 Configuration Sampling and Feasibility

First joint configurations $\mathbf{q}$ are sampled uniformly within their admissible position limits. For each sampled configuration, feasible joint velocities $\dot{\mathbf{q}}$ are determined via a discrete search over admissible pitch angles.

Let $\mathbf{p}_t$ denote the target position. Define d as the horizontal distance to the target and Δz as the vertical displacement. The pitch angle θ is varied over

$$\theta \in [\theta_{\min}, \theta_{\max}]$$

with increment $\Delta\theta$. For each θ, the required ballistic launch speed is

$$v_0^2 = \frac{g d^2}{2\cos^2\theta\,(d\tan\theta - \Delta z)},$$

valid only if $d\tan\theta - \Delta z > 0$.

The corresponding Cartesian end-effector velocity is

$$\mathbf{v}(\theta) = \begin{bmatrix} v_0 \cos\theta\, \hat{\mathbf{d}} \\ v_0 \sin\theta \end{bmatrix},$$

where $\hat{\mathbf{d}}$ is the horizontal unit vector toward the target.

Joint velocities are obtained using a damped least-squares pseudoinverse of the translational Jacobian $\mathbf{J}_v(\mathbf{q})$:

$$\dot{\mathbf{q}}(\theta) = \mathbf{J}_v^\top (\mathbf{J}_v \mathbf{J}_v^\top + \lambda^2 \mathbf{I})^{-1} \mathbf{v}(\theta),$$

with λ a damping factor and joint limits enforced after inversion. Among admissible pitch angles, the solution minimizing the maximum joint velocity magnitude is selected:

$$\dot{\mathbf{q}} = \arg\min_\theta \|\dot{\mathbf{q}}(\theta)\|_\infty.$$

Configurations without feasible solutions over the interval $[\theta_{\min}, \theta_{\max}]$ are discarded. For feasible release states $(\mathbf{q}, \dot{\mathbf{q}})$, a time-optimal constant-jerk S-curve trajectory is generated *backwards in time*, ensuring all joint velocities reach zero simultaneously while respecting position, velocity, acceleration, and jerk limits.

Collision checking along the trajectory ensures feasibility. Only candidates satisfying all kinematic, dynamic, and collision constraints proceed to refinement. [2]

4.3.2 Forward Feasibility and Refinement

This section introduces a method to modify feasible release states $(\mathbf{q}, \dot{\mathbf{q}})$, as obtained through the method described in Section 4.3.1, using forward propagation and midpoint-based reconstruction, with the goal of improving tolerance to release-time uncertainty within $\mathcal{I}_r$.

Forward Feasibility Given $(\mathbf{q}, \dot{\mathbf{q}})$, forward propagation under constant velocity for a time interval $\mathcal{I}_f$ with duration $\Delta t_f = \Delta t_r/2$ yields

$$\mathbf{q}_f = \mathbf{q} + \dot{\mathbf{q}}\, \Delta t_f.$$

At $\mathbf{q}_f$, for each pitch angle $\theta \in [\theta_{\min}, \theta_{\max}]$, the ballistic velocity $\mathbf{v}(\theta)$ is computed, and the associated joint velocity $\dot{\mathbf{q}}_f(\theta)$ is obtained via damped least-squares inversion of $\mathbf{J}_v(\mathbf{q}_f)$ with joint limits enforced, as in Section 4.3.1.

The velocity minimizing deviation from the initial configuration is selected:

$$\dot{\mathbf{q}}_f = \arg\min_\theta \|\dot{\mathbf{q}}_f(\theta) - \dot{\mathbf{q}}\|^2.$$

This preserves motion direction while maintaining ballistic feasibility. If no admissible solution exists, refinement stops and a new sample is selected.

[2] For ARTS, feasibility is checked against the configuration $(\mathbf{q}_r - \dot{\mathbf{q}}_r \Delta t_d, \dot{\mathbf{q}}_r)$, with $\mathbf{q}_r = \mathbf{q} - \dot{\mathbf{q}} \Delta t_f/2$, see also Section 4.3.4.

Algorithm 1 ARTS - Articulated Robotic Arm Robust Throw Search

Start timer
while elapsed time $< T_{\text{search}}$ **do**
  Sample joint configuration $\mathbf{q}$ within limits
  Compute feasible joint velocity $\dot{\mathbf{q}}$
  if no feasible velocity exists **then**
    continue
  end if
  Validate backward constant-jerk S-curve trajectory for collisions and kinematic/dynamic limits
  if any constraint violation is detected **then**
    continue
  end if
  Initialize best candidate: $c_{\min} \leftarrow \emptyset$
  Initialize minimal error: $e_{\min} \leftarrow \infty$
  for $i = 1$ to N_r **do**
    Forward propagate: $\mathbf{q}_f \leftarrow \mathbf{q} + \dot{\mathbf{q}}\,\Delta t_f$
    Compute forward joint velocity $\dot{\mathbf{q}}_f$ minimizing deviation from $\dot{\mathbf{q}}$
    if no feasible $\dot{\mathbf{q}}_f$ exists **then**
      break
    end if
    Compute midpoint state: $\bar{\mathbf{q}} \leftarrow (\mathbf{q} + \mathbf{q}_f)/2$, $\bar{\dot{\mathbf{q}}} \leftarrow (\dot{\mathbf{q}} + \dot{\mathbf{q}}_f)/2$
    Refine configuration: $\mathbf{q}_{\text{next}} \leftarrow \bar{\mathbf{q}} - 0.5\,\Delta t_f\,\bar{\dot{\mathbf{q}}}$
    Evaluate robustness metric: $e \leftarrow \epsilon(\mathbf{q}_{\text{next}}, \bar{\dot{\mathbf{q}}}) + \epsilon(\mathbf{q}_{\text{next}} + \bar{\dot{\mathbf{q}}}\,\Delta t_f, \bar{\dot{\mathbf{q}}})$
    if $e < e_{\min}$ **then**
      $c_{\min} \leftarrow (\mathbf{q}_{\text{next}}, \bar{\dot{\mathbf{q}}})$
      $e_{\min} \leftarrow e$
    end if
    Compute feasible joint velocity $\dot{\mathbf{q}}_{\text{next}}$ at $\mathbf{q}_{\text{next}}$
    Update current state: $\mathbf{q} \leftarrow \mathbf{q}_{\text{next}}, \quad \dot{\mathbf{q}} \leftarrow \dot{\mathbf{q}}_{\text{next}}$
  end for
  if $c_{\min} \neq \emptyset$ **then**
    $C \leftarrow C \cup \{c_{\min}\}$
  end if
end while
return C

Midpoint Reconstruction and Robustness Evaluation ARTS proceeds as follows:

1. **Midpoint Construction:** Compute the intermediate state

$$\bar{\mathbf{q}} = \frac{\mathbf{q} + \mathbf{q}_f}{2}, \quad \bar{\dot{\mathbf{q}}} = \frac{\dot{\mathbf{q}} + \dot{\mathbf{q}}_f}{2}.$$

2. **Refined Release State:** Step backward along the midpoint velocity

$$\mathbf{q}_{\text{next}} = \bar{\mathbf{q}} - 0.5\,\Delta t_f\,\bar{\dot{\mathbf{q}}}.$$

3. **Temporal Sensitivity Evaluation:** Quantify robustness under symmetric release-time perturbations

$$e = \epsilon(\mathbf{q}_{\text{next}}, \bar{\dot{\mathbf{q}}}) + \epsilon(\mathbf{q}_{\text{next}} + \bar{\dot{\mathbf{q}}}\,\Delta t_f, \bar{\dot{\mathbf{q}}}),$$

   where $\epsilon(\mathbf{q}, \dot{\mathbf{q}})$ denotes the landing error. The two terms correspond to early and delayed releases within $\mathcal{I}_f$.

Smaller values of e indicate release states with low landing position error at both ends of the interval, and correspond to robustness against release-time uncertainty, while larger values indicate higher sensitivity.

4.3.3 Iterative Refinement

The configuration $\mathbf{q}_{\text{next}}$ from midpoint reconstruction serves as input for the next refinement iteration (4.3.2). Iterations continue while the configuration remains feasible and the maximum number of steps N_r is not exceeded. Each candidate producing a smaller landing error than previously is replacing $c_{\min}$.

After N_r iterations for a given initial configuration, the configuration $c_{\min}$, which is the one with the smallest e, is added to the configuration set C, and a new feasible random configuration is drawn (see Section 4.3.1). The process continues until the total computational budget T_{search} is exhausted.

The set C consists of all randomly sampled states that were feasible and iteratively refined. It serves as the set of candidates from which a $(\mathbf{q}, \dot{\mathbf{q}})$ is selected to hit the defined target.

4.3.4 Implementation Details

The pitch angle θ is varied over $[\theta_{\min}, \theta_{\max}] = [-30.0°, 80.0°]$ with step size $\Delta\theta = 0.1°$.

The refinement procedure is performed for $N_r = 300$ iterations. The duration of the robust release interval is set to $\Delta t_r = 100$ ms, and the corresponding duration of the forward propagation interval is $\Delta t_f = 50$ ms.

Candidate release configurations $(\mathbf{q}, \dot{\mathbf{q}}) \in C$ are adjusted to account for the duration of Δt_r:

$$\mathbf{q}_r = \mathbf{q} - \frac{\Delta t_r - \Delta t_f}{2}\,\dot{\mathbf{q}}.$$

This backward shift along the velocity vector effectively centers the interval $\mathcal{I}_f$ within the full release interval $\mathcal{I}_r$. Since the algorithm explicitly evaluates errors only at the start and end of $\mathcal{I}_f$, the shift ensures that these checked points correspond to the centers of the two halves of the release interval. As a result, release-timing deviations are controlled and distributed evenly across the $\mathcal{I}_r$ interval.

For the S-curve acceleration trajectory, parameters were chosen for the UR5 robot, with a maximum jerk of 100 rad/s^3, a maximum acceleration of 6 rad/s^2, and a maximum velocity of 3 rad/s for all joints.[3]

To allow the robot to adapt after acceleration, the trajectory is generated such that the system reaches $\mathbf{q} - \dot{\mathbf{q}}\Delta t_d$ with joint velocity $\dot{\mathbf{q}}$. This allows the system to settle during Δt_d before the actual release interval begins. For both simulation and real-world experiments, $\Delta t_d = 24$ ms. Larger values would reduce the available leeway for acceleration, while smaller values may introduce noise due to insufficient settling time after acceleration.

4.4 Simulation and Experimental Results

The proposed algorithm ARTS was validated through theoretical performance evaluation, simulation, and real-world experiments. Both the simulation and the physical experiments were conducted using a wall-mounted 6-DoF UR5 robot, positioned approximately one meter above the ground near a corner of the workspace.

As in Chapter 3, simulations were performed using MuJoCo and the simplified gripper model. For the real-world setup, a Robotiq 3F gripper served as the end-effector. The experimental and simulation setups are shown in Fig. 4.1.

The joints of the UR5 are denoted by $\mathbf{q}_i$ for $i \in \{1, \ldots, 6\}$, starting with the shoulder pan joint and ending with the final wrist joint. The corresponding joint velocities are denoted by $\dot{\mathbf{q}}_i$.

To reach a desired joint position and velocity prior to release, Ruckig [Berscheid and Kroeger, 2021] is used to generate a time-optimal trajectory from the current joint configuration to a target state. [4] As the last joint of the UR5 arm does not affect translational motion, the gripper fingers were oriented such that the total force vector acting on the object to be thrown is aligned with the plane parallel to the gripper finger surfaces. This alignment was chosen to avoid additional release delay and deflection; however, it has not been explicitly evaluated in this work against alternative orientations, such as aligning this plane with the direction of motion as proposed in [Liu and Billard, 2024].

4.4.1 Theoretical Performance Evaluation

In this section, ARTS is evaluated and compared against simple random configuration sampling, first in detail for a fixed target and then across a range of random target locations.

[3] Maximum jerk and acceleration were estimated, whereas the maximum velocity is specified as 180 deg/s [ROS-Industrial, 2026]. Position limits were set with respect to the experimental setup.

[4] If the trajectory generated by Ruckig fails due to an unsuccessful collision check, an alternative approach would be to use the initial state of the S-curve acceleration trajectory from Section 4.3.1 as the target state for, e.g., MoveIt [Görner et al., 2019]. This approach also enables a clear separation between the backswing and the throwing phase with respect to maximum velocities and accelerations.

For each target, the algorithm is executed for a search duration of $T_{\text{search}} = 120$ s, using the parameters specified in Section 4.3.4.

As a baseline, random configurations are sampled and checked for feasibility. Unlike with ARTS, these random configurations do not undergo a iterative refinement procedure. For each feasible random configuration, the corresponding joint velocity $\dot{\mathbf{q}}$ required to hit the target is computed as described in Section 4.3.1, and the configuration is adjusted according to

$$\mathbf{q} = \mathbf{q}_{\text{rand}} - \frac{\Delta t_r}{2}\,\dot{\mathbf{q}},$$

and stored along with $\dot{\mathbf{q}}$. [5] This temporal shift centers the zero-error configuration within the release interval, which is expected to reduce the mean landing error.

Note that the error at the original random configuration is zero by construction, since the joint velocities are specifically computed to reach the target from $\mathbf{q}_{\text{rand}}$ at $\dot{\mathbf{q}}$.

To assess the performance, several metrics were considered. These include the number of feasible joint configurations identified during the search and the predicted landing errors over the release interval. Additionally, for the fixed target assessed in the following Section 4.4.1 the convergence behavior of ARTS is visualized to provide insight into the optimization dynamics and the corresponding evolution in joint space.

Detailed Inspection at a Fixed Target

A detailed inspection was conducted for a fixed target located at $[1.7, 0.0, 0.0]$ m. The target is positioned 1.7 m perpendicular to the wall on which the robot is mounted and lies at floor level. This setup allows a focused evaluation of the proposed algorithm, demonstrating how joint configurations are selected to minimize the landing error for a single, stationary target in front of the robot.

ARTS Over the 120 s search duration, ARTS identified six feasible configurations. For five of these, the performance metric was successfully reduced to values near zero, indicating robust throws.

The first randomly sampled feasible configuration that successfully converged is presented in detail. Fig. 4.2 illustrates the iterative improvement of the performance metric across the corresponding joint configurations, showing a reduction of the error metric as the refinement progresses.

It is important to note that $\mathbf{q}_6$ remains constant throughout the optimization. In this setup, the sixth joint does not contribute to translational motion because the center of mass of the grasped object is aligned with the rotation axis of the final joint. Consequently, variations in $\mathbf{q}_6$ do not affect the translational release position or velocity of the projectile.

Corresponding to the refinement process shown in Fig.4.2, Fig. 4.3 presents the landing error over the release interval $\mathcal{I}_r$ for the configuration with the smallest robustness metric e. The minimum landing error occurs at the release times where ϵ was minimized, specifically at

[5] To check the feasibility of reaching this state, the S-curve trajectory and check described in Section 4.3.1 were applied.

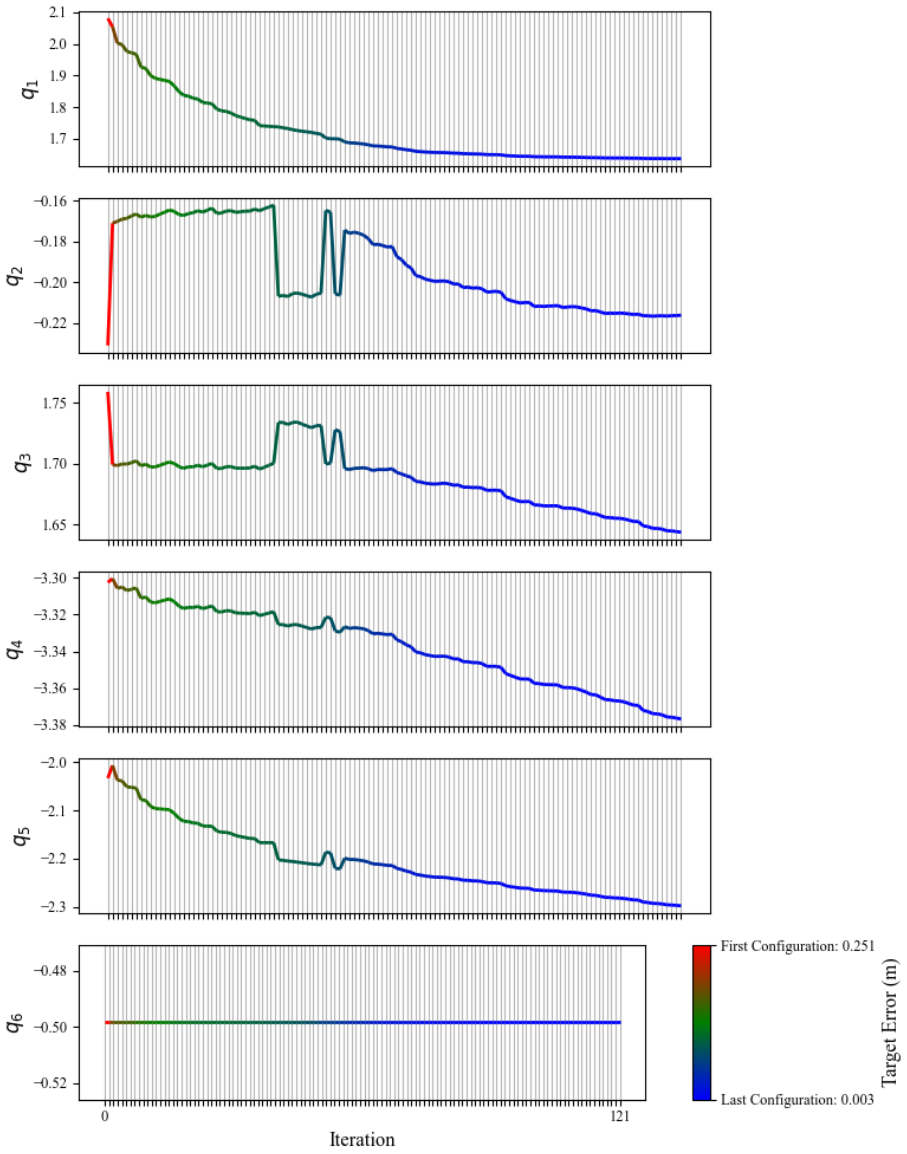


Figure 4.2: Configuration progression from the first to the 300th iteration for the fixed target at $[1.7, 0.0, 0.0]$ m. Vertical lines indicate iterations where the error e decreased compared to previous iterations.

25 ms and 75 ms. The error increases on either side of these points, exhibiting an approximately symmetric profile. However, it can be observed that the landing error remains consistently below 1 cm over the interval [20 ms, 80 ms], with the maximum error not exceeding 2.3 cm. Additional error profiles for the remaining four configurations found by ARTS are provided in Appendix B.2.

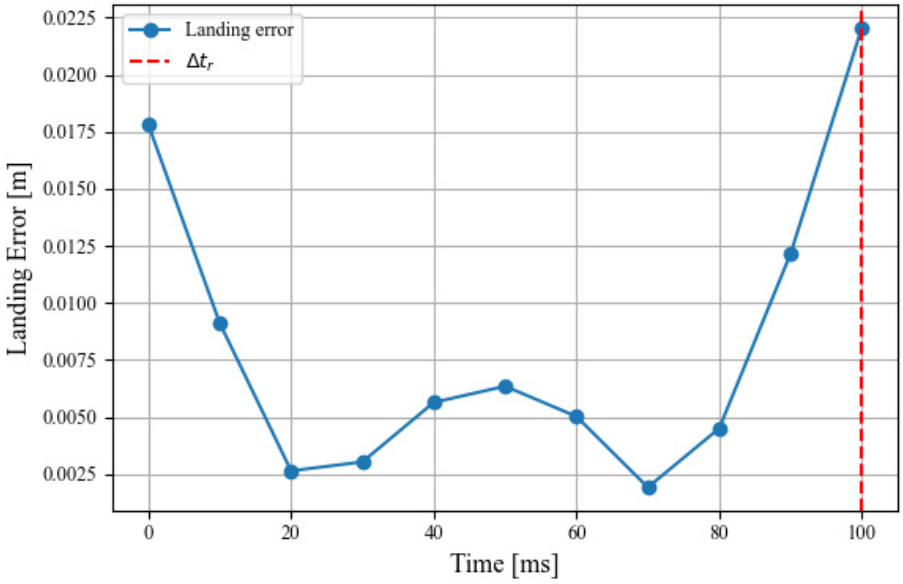


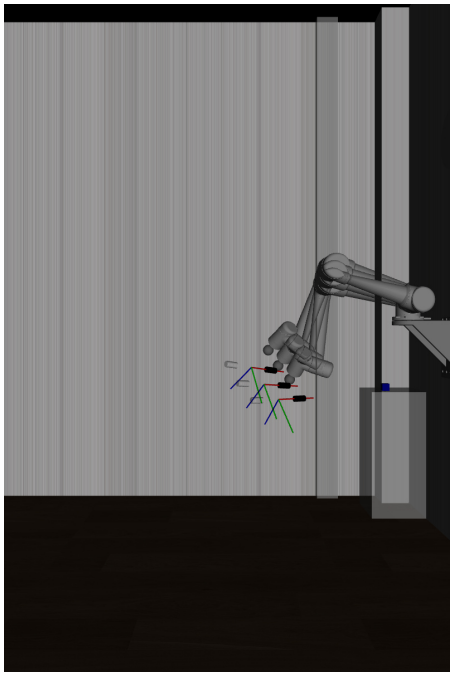
Figure 4.3: ARTS Configuration: Landing error of the thrown object during the release interval $\mathcal{I}_r$ for the fixed target at $[1.7, 0.0, 0.0]$ m. This corresponds to the first successfully converged configuration.

To illustrate first four of the six configurations found by ARTS, the corresponding release configuration $\mathbf{q}_r$ is visualized in MuJoCo in Fig. 4.4. The figure shows a composition of the release interval at 0 ms, 50 ms, and 100 ms. It can be observed that, although there appears to be low variability in the shoulder pan joint, the remaining joints exhibit greater variability.

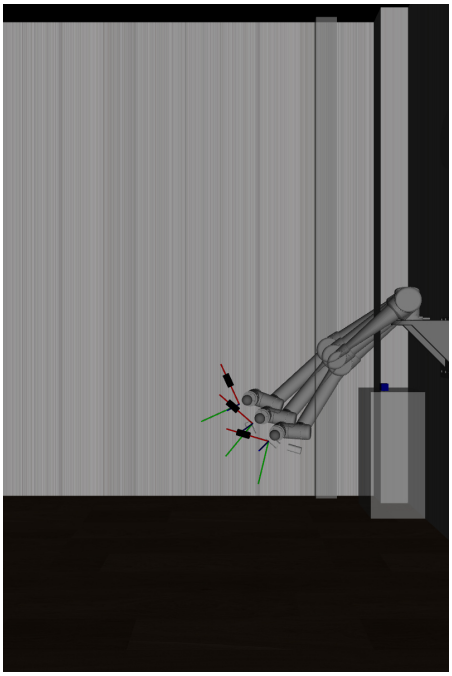
Random Configuration Sampling To assess the performance of pure random sampling relative to the proposed heuristic approach ARTS, configurations were randomly sampled over a search period of 120 s, as described in Section 4.4.1. Within this time frame, 121 feasible configurations were identified. The five configurations yielding the smallest mean landing errors were selected for detailed analysis.

The configuration with the smallest mean error, [6] obtained after a total search time of $T_{\text{search}} = 120$ s using simple random sampling is referred to as MMERC (Minimal Mean Error Random Configuration). Fig. 4.5 illustrates the landing error as a function of the release interval $\mathcal{I}_r$ for the MMERC.

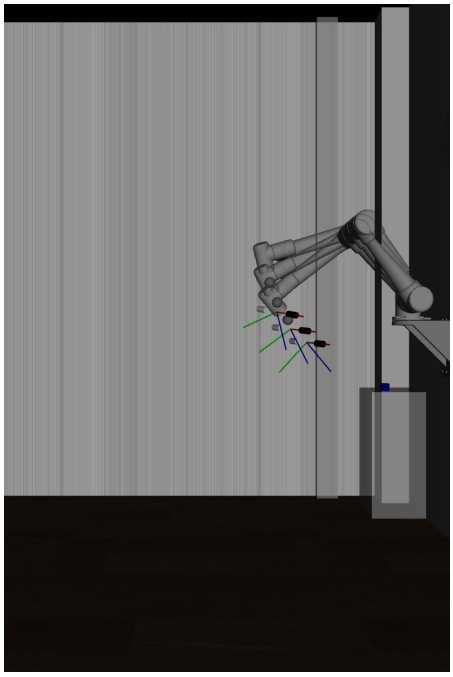
[6] The method to calculate the mean landing error is described below in Section 4.4.1.



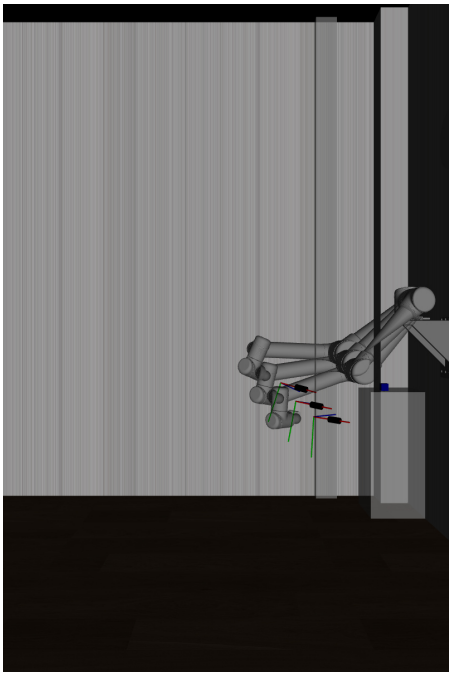
(a) Config. 1



(b) Config. 2



(c) Config. 3



(d) Config. 4

Figure 4.4: Four out of five throwing configurations identified during the 120 s ARTS search process for the target located at $[1.7, 0.0, 0.0]$ m, overlay during the release interval at 0ms, 50ms and 100ms.

The estimated landing error remains below 2 cm throughout the interval [20 ms, 80 ms] and it increases to more than 3 cm toward the boundaries of the release interval, indicating reduced robustness with respect to timing deviations.

The predicted landing errors for the remaining four configurations are presented in Appendix B.2.

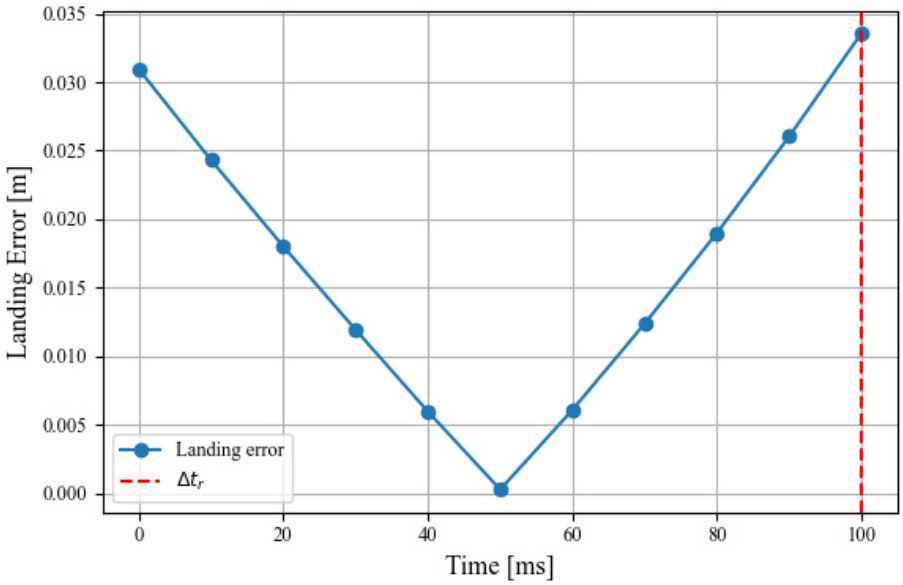


Figure 4.5: Random Configuration: Landing error of the thrown object during the release interval $\mathcal{I}_r$ for the fixed target at $[1.7, 0.0, 0.0]$ m. This corresponds to the MMERC obtained after $T_{\text{search}} = 120$ s of sampling.

Comparison To compare the proposed ARTS described in Section 4.3 with plain random sampling, the mean landing error over $\mathcal{I}_r$ and the corresponding standard deviation were evaluated. [7]

The choice of the mean error over $\mathcal{I}_r$ as an evaluation metric is motivated by practical considerations. Although the release dynamics of the gripper may follow a distribution closer to Gaussian than uniform, the effective release time can additionally vary depending on object properties, such as material stiffness. Therefore, evaluating performance over the entire interval $\mathcal{I}_r$ provides a meaningful measure of robustness under realistic operating conditions. The results are summarized in Table 4.1.

It can be observed that the configuration found with ARTS with the smallest mean landing error reduces the mean error by approximately a factor of two, while also exhibiting the lowest standard deviation compared to the MMERC. Furthermore, all ARTS configurations achieve a lower mean error than the MMERC, and only one configuration found with ARTS has a higher

[7] To calculate the mean error and the corresponding standard deviation, the landing error was sampled at a rate of 1 ms. With $\Delta t_r = 100$ ms, this results in 100 samples within each interval $\mathcal{I}_r$.

standard deviation than the MMERC. These results indicate improved robustness with respect to variations in the release timing.

Table 4.1: Mean and standard deviation of landing errors for configurations found with ARTS and simple random search, target at $[1.7, 0, 0]$ m.

	ARTS		Random Search	
Configuration	Mean [cm]	Std [cm]	Mean [cm]	Std [cm]
1	0.9	0.8	1.8	1.1
2	0.7	0.6	2.3	1.3
3	1.6	1.4	1.8	1.5
4	1.3	1.1	2.8	1.6
5	0.6	0.5	2.6	1.7

Random Target Evaluation

Random target locations are generated within a box-shaped workspace. One corner of the box is positioned 1 m from the wall on which the robot is mounted and 1 m laterally from the adjacent wall. The box is aligned parallel to the walls and has dimensions of $1.5 \times 1.5 \times 2.0$ m.

As in the fixed-target evaluation, ARTS is executed for 120 s for each randomly generated target and the performance of the proposed search heuristic Alg. 1 is compared against plain random sampling as above.

Table 4.2 summarizes the results. Configurations for which the estimated thrown object does not intersect the target plane at any release time within the 100 ms interval are discarded and therefore not included in the statistics.

The results indicate that, in most cases, the heuristic achieves substantially lower errors compared to simple random sampling. A more detailed breakdown of the evaluated configurations is provided in Appendix B.2.

Table 4.2: Comparison per randomly generated target (120 s per target). Values are reported in centimeters.

Target [x,y,z] (m)	ARTS		Random Sampling	
	Mean [cm]	Std [cm]	Mean [cm]	Std [cm]
[1.65, 1.02, 1.32]	–	–	–	–
[2.17, 0.66, 1.93]	–	–	–	–
[2.47, 0.13, 0.80]	6.0	4.8	15.8	9.6
[2.21, 0.27, 0.98]	7.5	6.2	28.6	16.6
[2.12, 0.34, 1.94]	–	–	–	–
[1.53, 0.89, 0.32]	1.2	0.9	1.4	1.0
[1.58, 0.87, 1.11]	–	–	14.5	8.5
[1.70, 0.67, 0.88]	3.5	2.9	5.2	4.8
[1.87, 0.09, 0.46]	1.9	1.4	6.6	6.2
[1.29, 1.05, 0.48]	1.3	1.0	4.2	4.0

4.4.2 Performance Evaluation in Simulation

A MuJoCo simulation environment is used to evaluate and visualize the throwing performance of the UR5 robot equipped with the simplified gripper. [8]

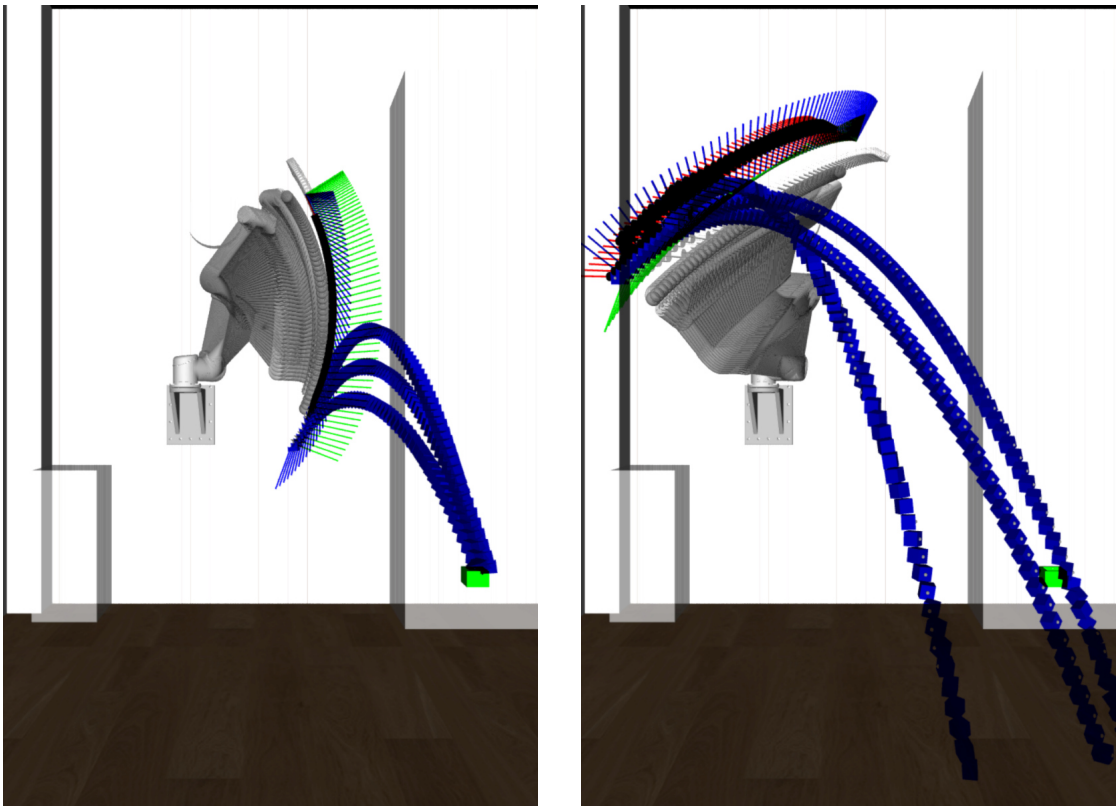
To compare configurations found with ARTS to simple random sampling, throws are executed toward two random target positions from the previous Section 4.4.1. For this purpose, several trials are performed with different offsets for triggering the gripper opening command within the release interval $\mathcal{I}_r$: 0 ms, 40 ms, and 80 ms. This allows the visualization and assessment of the influence of the release timing on the resulting flight trajectory.

The release dynamics are implemented analogously to the delta robot described in Section 3.5.2. In particular, the normal force exerted by the gripper cylinder on the object is gradually reduced over time until detachment occurs. A initial normal force of about 10 N is commanded, resulting in release dynamics as illustrated in Fig. 3.8a, and a duration of approximately 25 ms from the gripper opening command to full detachment.

For both targets, throws with the first successfully converged configuration found using Alg. 1 and the configuration with the minimal mean error among randomly sampled candidates are simulated. For each target and release configuration, the throws were recorded with each release offset, and an overlay image was composed. The results are shown in Fig. 4.6 for the target at $[1.29,\ 1.05,\ 0.48]$ and in Fig. 4.7 for the target at $[1.53,\ 1.59,\ 0.32]$.

Both figures show a significantly more robust outcome for the heuristically found configuration. Although the trajectories initially diverge due to different release times, the flight trajectories converge back toward the target later on. In contrast, the trajectories generated with the randomly selected configuration exhibit a higher landing error for early and late releases.

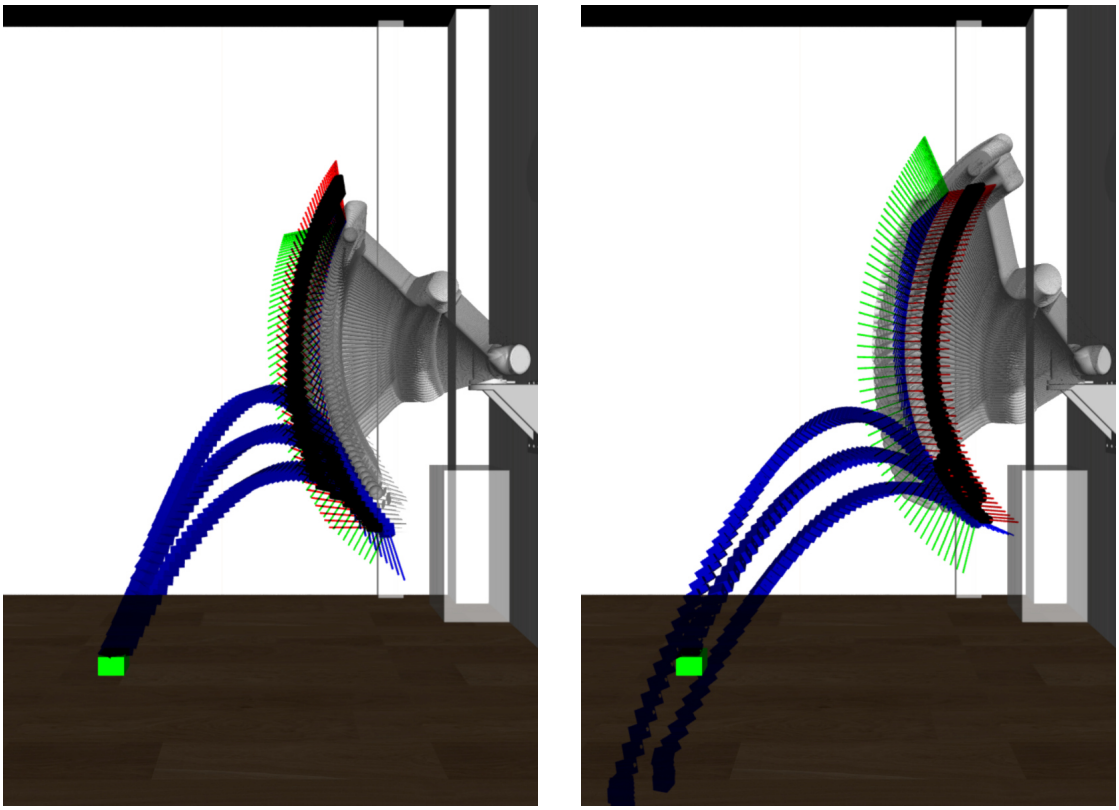
[8] The URDF from the lab setup [TAMS-Group, 2025] was converted as in the previous Chapter 3 and modified. The Python source code from Ian Chuang [2026] served as a basis.



(a) ARTS configuration

(b) Random configuration (minimal mean error)

Figure 4.6: Throwing results for the target position $[1.29,\ 1.05,\ 0.48]$. Comparison between the first succesfully converged ARTS configuration and the randomly sampled configuration with minimal mean error. For each configuration, gripper opening offsets of 0 ms, 40 ms, and 80 ms are evaluated.



(a) ARTS configuration

(b) Random configuration (minimal mean error)

Figure 4.7: Throwing results for the target position $[1.53,\ 1.59,\ 0.32]$. Comparison between the ARTS configuration and the the randomly sampled configuration with minimal mean error. Gripper opening offsets of 0 ms, 40 ms, and 80 ms are considered.

4.4.3 Performance Evaluation on Real Robot

To assess the performance on a real robot, a qualitative evaluation was conducted on the UR5 setup. Configurations were generated using ARTS for three targets located on the floor and one target placed on a table.

The control update frequency was set to 125 Hz, and the trajectory was generated at the same rate, resulting in position commands every 8 ms. [9]

For each target, multiple throwing trials were executed using both a golf ball and a tennis ball, while keeping the trajectory and gripper opening constraints identical.

The gripper opening was handled by a separate thread that continuously monitored the joint states. The release was triggered once the Euclidean norm between the current joint configuration and a predefined point along the trajectory fell below a threshold. This point was chosen to be ten trajectory samples (corresponding to 80 ms) before the desired release interval, compensating for the gripper opening delay.

The targets consisted of cups with a diameter of 5.5 cm for the golf ball (4.3 cm diameter) and 8 cm for the tennis ball (6.6 cm diameter). [10] In both cases, the cup opening was positioned 12 cm above the bottom of the cup. [11]



Figure 4.8: The four distinct objects used to assess throwing performance. The tennis ball and the golf ball are used for detailed assessment, with quantitative results reported in Table 4.4. The racquet ball and the wooden block are included primarily for visualization of the resulting trajectories as in Fig. 4.10.

For each target listed in Table 4.3, 20 throwing trials were performed. The quantitative results are summarized in Table 4.4[12]

Across all four targets and all objects, only one throw missed, resulting in an overall success rate of $\frac{159}{160} \approx 99.4\%$.

[9] The repository from TAMS-Group [2025] served as the basis for controlling the real robot.

[10] The targets can be seen in Fig. 4.10 for the golf ball and Fig. 4.10 for the tennis ball.

[11] The white cup is 12 cm in height. The smaller yellow cup was mounted on a heavy base to match the same height and prevent slippage if struck by the golf ball.

[12] Throws in which the ball landed in the cup but subsequently flipped out are counted as unsuccessful.

Table 4.3: Target positions in meters (x, y, z).

Target	Position [m] (x, y, z)
1	[1.7, 0.0, 0.12]
2	[2.0, 0.3, 0.12]
3	[1.7, 0.8, 0.12]
4	[1.6, 0.4, 0.9]

Table 4.4: Throwing success results (number of successful throws out of 20)

Ball Type	Target 1	Target 2	Target 3	Target 4
Golf Ball	20 / 20	20 / 20	20 / 20	20 / 20
Tennis Ball	19 / 20	20 / 20	20 / 20	20 / 20

Fig. 4.9 shows the noisy detachment behavior of the golf ball when throwing toward Target 4 on the table. Although the trajectories initially appear to diverge, they converge back toward the target afterwards.

To make this observation clearer, Fig. 4.10 presents the trajectories of all four objects shown in Fig. 4.8. Each object is shown in a trial where it was successfully thrown once.

The figure indicates that the stiff and smooth golf ball is released first, while the racquetball, which is more compressible and sticky, is released last.



Figure 4.9: Target 4 for the golf ball. Overlay of 20 successful throws recorded at 30 fps. The noise in the release can be seen even with the same object, as the flight trajectories diverge before they converge back to the target.

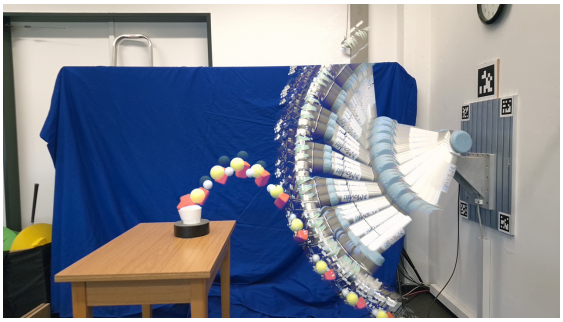


Figure 4.10: Target 4, overlay of four successful throws recorded at 30 fps using four distinct objects, as shown in Fig. 4.8. The flight trajectories of the golf ball and racquetball differ clearly, likely due to the later detachment of the racquetball caused by its softer material and higher friction.

4.5 Summary

Throwing with a closed grasp using an articulated robotic arm poses several challenges, including accurate trajectory tracking as well as uncertain detachment. Nevertheless, it was shown that joint configurations exist which, even at constant joint velocity, can successfully hit a target, despite the object detachment isn't ideally instantaneous. To find such configurations, a heuristic search method called ARTS was proposed.

The performance of ARTS was evaluated and compared to a baseline approach based on simple random sampling of joint configurations. The results indicate that ARTS can find robust throwing configurations more efficiently and reliably than random sampling.

However, the robustness of the configurations found by ARTS appears to come at the cost of a significantly reduced set of feasible release configurations. Furthermore, for the wall-mounted UR5 setup and the constraints considered in this work, only bottom-up throwing motions were identified. In particular, the number and variety of configurations is much smaller compared to approaches based on random sampling or the velocity hedgehog method [Liu et al., 2022]. These methods, together with tube acceleration, also provide robustness against release uncertainty [Liu and Billard, 2024]; however, this comes at the cost that not only the desired position and velocity must be achieved prior to release, but also the desired acceleration.

Despite these limitations, the results remain promising for industrial applications in which targets are fixed and pickup locations can be selected accordingly, such that the reduced set of feasible trajectories does not become a limiting factor.

4.6 Future Work

This work demonstrated throwing accuracy and robustness on a position-controlled industrial robot using a non-specialized gripper. These results suggest the potential for replacing PaP applications in industrial settings with PaT approaches. Accordingly, a dedicated evaluation of such use cases would be of interest, particularly with respect to cycle time and power consumption.

In this work, only objects grasped at their center of mass were considered. It would be interesting to evaluate performance when this is not the case. Due to the use of approximately constant joint velocity, the forces toward the object may be primarily influenced by gravity, potentially enabling robust throws even for off-center grasps in certain configurations.

Another interesting direction for future work would be to initialize ARTS not with random samples, but with configurations obtained from previous ARTS runs with nearby targets, different CoM locations relative to the end-effector, or different flight dynamics, and to evaluate the extent to which such configurations can be reused under these perturbations.

Another possible extension is to assess variability and robustness by systematically varying pitch, velocity, and acceleration at configurations found with ARTS, as explored with configurations found by velocity hedghehog in Liu et al. [2025].

Chapter 5

Summary and Future Work

In this work, prehensile throwing has been evaluated on two gripper-equipped robotic systems: a delta robot and an articulated robotic arm. Due to their distinct structures, two different approaches were developed to achieve robust throws despite uncertain release within a release interval. It was shown that both systems are capable of accurate throws even when the release is noisy or uncertain.

For the delta robot, the duration of the release interval can be adjusted over a wide range to accommodate different grippers and objects. Furthermore, it was shown that the torque during the designed release remains relatively small, which likely reduces vibrations, power consumption, and mechanical wear.

For the articulated robotic arm, it was demonstrated that certain joint positions, when traversed at approximately constant joint velocity, allow for robust throws during a release interval. While the error does not reduce to zero over the entire interval, as it can for the delta robot, high accuracy was still achieved, as evidenced by throwing balls into a cup.

Regarding future work, the delta robot shows particularly promising features for replacing PaP operations with PaT in industrial setups.

Throwing with articulated robotic arms, however, remains a challenging problem, with many open questions due to the system's high complexity.

Bibliography

ABB Robotics. IRB 360 FlexPicker product page, 2026. URL https://new.abb.com/products/robotics/industrial-robots/irb-360. Accessed: 2026-02-21.

E.W. Aboaf, C.G. Atkeson, and D.J. Reinkensmeyer. Task-level robot learning. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1309–1310 vol.2, 1988. doi: 10.1109/ROBOT.1988.12245.

Lars Berscheid and Torsten Kroeger. Jerk-limited real-time trajectory generation with arbitrary target states. In *Robotics: Science and Systems XVII. Ed:: Dylan A. Shell*. Robotics: Science and Systems Foundation, 2021. ISBN 978-0-9923747-7-8. doi: 10.15607/RSS.2021.XVII.015.

John C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 3 edition, 2016. doi: 10.1002/9781119121534.

Justin Carpentier, Joseph Mirabel, Nicolas Mansard, and the Pinocchio Development Team. Pinocchio: fast forward and inverse dynamics for poly-articulated systems, 2015–2018. URL https://stack-of-tasks.github.io/pinocchio.

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*, 40:3232–3251, 2024. doi: 10.1109/TRO.2024.3416009.

Michael Görner, Robert Haschke, Helge Ritter, and Jianwei Zhang. Moveit! task constructor for task-level motion planning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 190–196, 2019. doi: 10.1109/ICRA.2019.8793898.

Hugo Hadfield, Lai Wei, and Joan Lasenby. The forward and inverse kinematics of a delta robot. In Nadia Magnenat-Thalmann, Constantine Stephanidis, Enhua Wu, Daniel Thalmann, Bin Sheng, Jinman Kim, George Papagiannakis, and Marina Gavrilova, editors, *Advances in Computer Graphics*, pages 447–458, Cham, 2020. Springer International Publishing. ISBN 978-3-030-61864-3.

Ghina Hassan, Marc Gouttefarde, Ahmed Chemori, Pierre-Elie Hervé, Maher El Rafei, Clovis Francis, and Damien Sallé. Time-optimal pick-and-throw s-curve trajectories for fast parallel robots. *IEEE/ASME Transactions on Mechatronics*, 27(6):4707–4717, 2022. doi: 10.1109/TMECH.2022.3164247.

Ian Chuang. Manipulator-Mujoco: Template Repository for Manipulator Control in MuJoCo, 2026. URL https://github.com/ian-chuang/Manipulator-Mujoco. Accessed: 2026-03-06.

K-Scale Labs. urdf2mjcf: URDF to MJCF Converter, 2026. URL https://github.com/kscalelabs/urdf2mjcf. Version 0.2.39.

Seungsu Kim, Alexandre Coninx, and Stephane Doncieux. From exploration to control: Learning object manipulation skills through novelty search and local adaptation. *Robotics and Autonomous Systems*, 136:103710, 2021. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2020.103710.

Takahiro Kizaki and Akio Namiki. Two ball juggling with high-speed hand-arm and high-speed vision system. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1372–1377, 05 2012. doi: 10.1109/ICRA.2012.6225090.

Yang Liu and Aude Billard. Tube acceleration: Robust dexterous throwing against release uncertainty. *IEEE Transactions on Robotics*, 40:2831–2849, 2024. doi: 10.1109/TRO.2024.3386391.

Yang Liu and Aude Billard. On transient release dynamics in robot throwing: A sliding pivot model. *IEEE Transactions on Robotics*, 42:132–151, 2026. doi: 10.1109/TRO.2025.3634386.

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, 2022. doi: 10.1109/IROS47612.2022.9981231.

Yang Liu, Bruno Da Costa, and Aude Billard. Learning to throw-flip. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3209–3216, 2025. doi: 10.1109/IROS60139.2025.11246138.

Arthur Louette, Gaspard Lambrechts, Damien Ernst, Eric Pirard, and Godefroid Dislaire. Reinforcement learning to improve delta robot throws for sorting scrap metal, 2024. URL https://arxiv.org/abs/2406.13453.

Yuntao Ma, Yang Liu, Kaixian Qu, and Marco Hutter. Learning accurate whole-body throwing with high-frequency residual policy and pullback tube acceleration. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1771–1778, 2025. doi: 10.1109/IROS60139.2025.11246279.

OpenAI. ChatGPT [Large language model], 2026. URL https://chat.openai.com/. Version used until March 21, 2026.

Kai Ploeger and Jan Peters. Controlling the cascade: Kinematic planning for n-ball toss juggling. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1139–1144, 2022. doi: 10.1109/IROS47612.2022.9981678.

Kai Ploeger, Michael Lutter, and Jan Peters. High acceleration reinforcement learning for real-world juggling with binary rewards. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 642–653. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/ploeger21a.html.

Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Springer, 2 edition, 2007. doi: 10.1007/b98885.

Fredy Raptopoulos, Maria Koskinopoulou, and Michail Maniadakis. Robotic pick-and-toss facilitates urban waste sorting. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1149–1154, 2020. doi: 10.1109/CASE48305.2020.9216746.

L. Rey and R. Clavel. The delta parallel robot. In C. R. Boër, L. Molinari-Tosatti, and K. S. Smith, editors, *Parallel Kinematic Machines*, pages 401–417, London, 1999. Springer London. ISBN 978-1-4471-0885-6. doi: 10.1007/978-1-4471-0885-6_29.

ROS-Industrial. joint_limits.yaml for UR5 from universal_robot repo, 2026. URL https://github.com/ros-industrial/universal_robot/blob/noetic/ur_description/config/ur5/joint_limits.yaml. Accessed: 2026-03-12.

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, 2008. doi: 10.1109/IROS.2008.4651142.

Avishai Sintov and Amir Shapiro. A stochastic dynamic motion planning algorithm for object-throwing. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2475–2480, 2015. doi: 10.1109/ICRA.2015.7139530.

TAMS-Group. tams_ur5_setup: UR5 robot setup repository, 2025. URL https://github.com/TAMS-Group/tams_ur5_setup. Accessed: 2026-03-06.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Claudio Urrea and John Kern. Recent advances and challenges in industrial robotics: A systematic review of technological trends and emerging applications. *Processes*, 13(3), 2025. ISSN 2227-9717. doi: 10.3390/pr13030832.

Lennart Werner, Fang Nan, Pol Eyschen, Filippo A. Spinelli, Hongyi Yang, and Marco Hutter. Dynamic throwing with robotic material handling machines. In *2024 IEEE/RSJ International*

Conference on Intelligent Robots and Systems (IROS), pages 98–104, 2024. doi: 10.1109/IROS58592.2024.10802743.

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. doi: 10.1109/TRO.2020.2988642.

Appendix A

Glossary

A.1 Prehensile throwing with Delta Robots

Time Variables

t Continuous time variable

t_0 Start time of the throwing motion

t_a End time of acceleration phase (Phase I)

t_r Nominal release time

t_e End time of the complete throwing motion

t_f Ballistic flight time

Δt Discrete integration time step

Δt_r Gripper release duration

Δt_d Transition delay between phases

T_j Duration of a single jerk phase

T_a Total acceleration duration ($2T_j$)

T_{comp} Phase II / Compensation duration

T_{throw} Total throw duration ($t_e - t_0$)

Kinematic Variables

$\mathbf{q}_i$ Joint position of joint i

$\dot{\mathbf{q}}_i$ Joint velocity of joint i

$\boldsymbol{\theta}$ Vector of Delta motor angles $[\theta_1, \theta_2, \theta_3]^T$

θ_i Motor angle of chain i

$\mathbf{p}$ End-effector position

$\mathbf{p}_0$ Initial end-effector position / Grasp position

$\mathbf{p}_r$ Nominal release position

$\mathbf{p}_t$ Target position

$\mathbf{v}$ End-effector velocity

$\mathbf{v}_r$ Nominal release velocity

$\mathbf{a}(t)$ Cartesian acceleration

$a_{\text{mag}}(t)$ Acceleration magnitude profile

$j_{\max}$ Maximum jerk

$\mathbf{J}$ Jacobian matrix

Geometric Parameters

L_u Length of upper arm

L_l Length of lower arm

r_b Base radius

r_p Platform radius

$\mathbf{C}_1, \mathbf{C}_2$ Circle centers in inverse kinematics

d Distance between circle centers

r_1 Radius of upper-arm circle ($= L_u$)

r_2 Radius of projected lower-arm circle

(x_i, z_i) Intersection point of the two circles in the XZ-plane

$\mathbf{e}_i$ Position vector of the i-th elbow joint in the base frame

$\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ Orthonormal basis vectors used in trilateration

i Projection of $(\mathbf{e}_3 - \mathbf{e}_1)$ onto $\mathbf{e}_x$

j Projection of $(\mathbf{e}_3 - \mathbf{e}_1)$ onto $\mathbf{e}_y$

ξ Local x-coordinate of the end-effector in trilateration frame

η Local y-coordinate of the end-effector in trilateration frame

ζ Local z-coordinate of the end-effector in trilateration frame

Directional Vectors

$\mathbf{\Delta}$ Vector from initial to target position

$\hat{\mathbf{d}}_h$ Horizontal unit direction vector

$\hat{\mathbf{d}}(\theta)$ 3D unit throwing direction

Dynamic Variables

$\mathbf{g}$ Gravitational acceleration vector

g_0 Gravitational acceleration constant ($9.81\,\mathrm{m/s^2}$)

$\tau_i(t)$ Joint torque of joint i

$\tau_{\max}^{\mathrm{acc}}$ Peak torque during acceleration phase

$\tau_{\max}^{\mathrm{comp}}$ Peak torque during compensation phase

F_{grasp} Commanded grasp force

Error and Evaluation Metrics

E_{pos} Horizontal landing position error

$E_{\mathrm{pos,max}}$ Maximum positional error

Discretization Variables

N_a Number of discrete steps in Phase I

N_c Number of discrete steps in Phase II

k Discrete time index

A.2 Prehensile throwing with Robotic Arms

Time Variables

Δt_f Duration of forward propagation interval

Δt_r Duration of release window

Δt_d Duration of settling window

T_{search} Total computation / search time allocated for finding candidate throws

Joint-Space Variables

$\mathbf{q}$ Vector of joint positions

$\dot{\mathbf{q}}$ Vector of joint velocities

$\mathbf{q}_0$ Initial grasp configuration

$\mathbf{q}_r$ Joint configuration at start of release window

$\mathbf{q}_f$ Forward-propagated joint configuration

$\dot{\mathbf{q}}_f$ Forward-propagated joint velocity

End-Effector / Cartesian Variables

$\mathbf{x}$ End-effector pose in Cartesian space

$\dot{\mathbf{x}}$ End-effector velocity, $\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$

$\mathbf{p}_t$ Target position

$\mathbf{v}(\theta)$ End-effector velocity required to reach target at pitch angle θ

$\hat{\mathbf{d}}$ Horizontal unit vector pointing from end-effector to target

Kinematic Matrices

$\mathbf{J}(\mathbf{q})$ Manipulator Jacobian mapping $\dot{\mathbf{q}}$ to $\dot{\mathbf{x}}$

$\mathbf{J}_v(\mathbf{q})$ Translational part of the Jacobian used for velocity inversion

$\mathbf{J}_v^\top(\mathbf{J}_v\mathbf{J}_v^\top + \lambda^2\mathbf{I})^{-1}$ Damped least-squares pseudoinverse

Ballistic / Throwing Variables

g Gravitational acceleration constant

d Horizontal distance from end-effector to target

Δz Vertical displacement from release to target

θ Pitch angle

v_0 Required launch speed for ballistic trajectory

$\epsilon(\mathbf{q}, \dot{\mathbf{q}})$ Landing error metric: deviation of thrown object from target

ARTS Algorithm Variables

N_r Maximum number of iterations per sample

$\bar{\mathbf{q}}$ Midpoint joint configuration between current and forward-propagated state

$\bar{\dot{\mathbf{q}}}$ Midpoint joint velocity between current and forward-propagated state

$\mathbf{q}_{\text{next}}$ Joint configuration for next iteration

e Robustness error metric for candidate release state

$e_{\min}$ Minimum robustness error observed for a sample

C Set of candidate sets

$c_{\min}$ Configuration with minimal e

Appendix B

Appendix

B.1 Visualization of Convergence and Throw Robustness

B.1.1 Visualization of Convergence

The following figures illustrate the convergence behavior of the remaining configurations identified by the ARTS algorithm over a 120 s optimization horizon. The proposed algorithm ARTS is performed for a throw toward the fixed target at $[1.7,\ 0.0,\ 0.0]$ m, as introduced in Section 4.3.2.

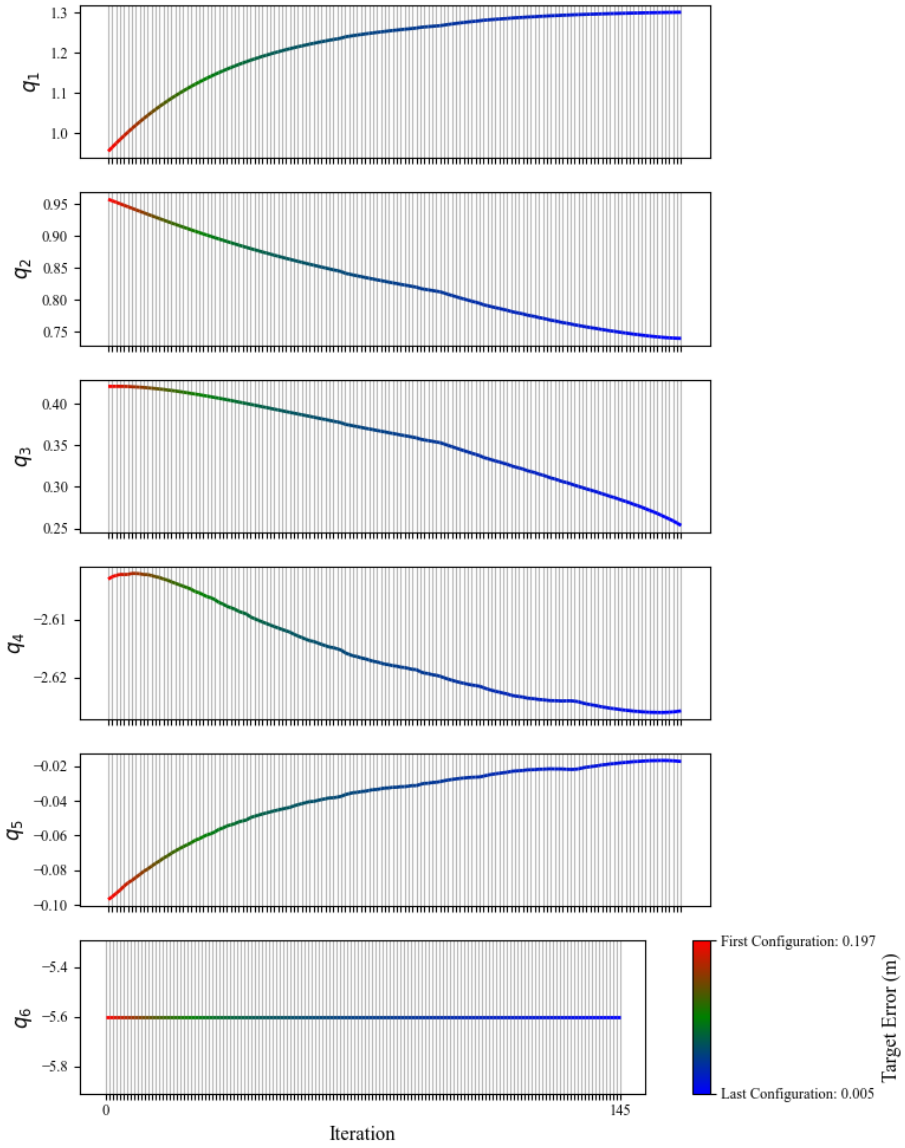


Figure B.1: Second successful evolution of the configuration parameters from iteration 1 to iteration 300 for the fixed target at [1.7, 0.0, 0.0] m. The progression illustrates the iterative adjustment of the joint configuration during ARTS.

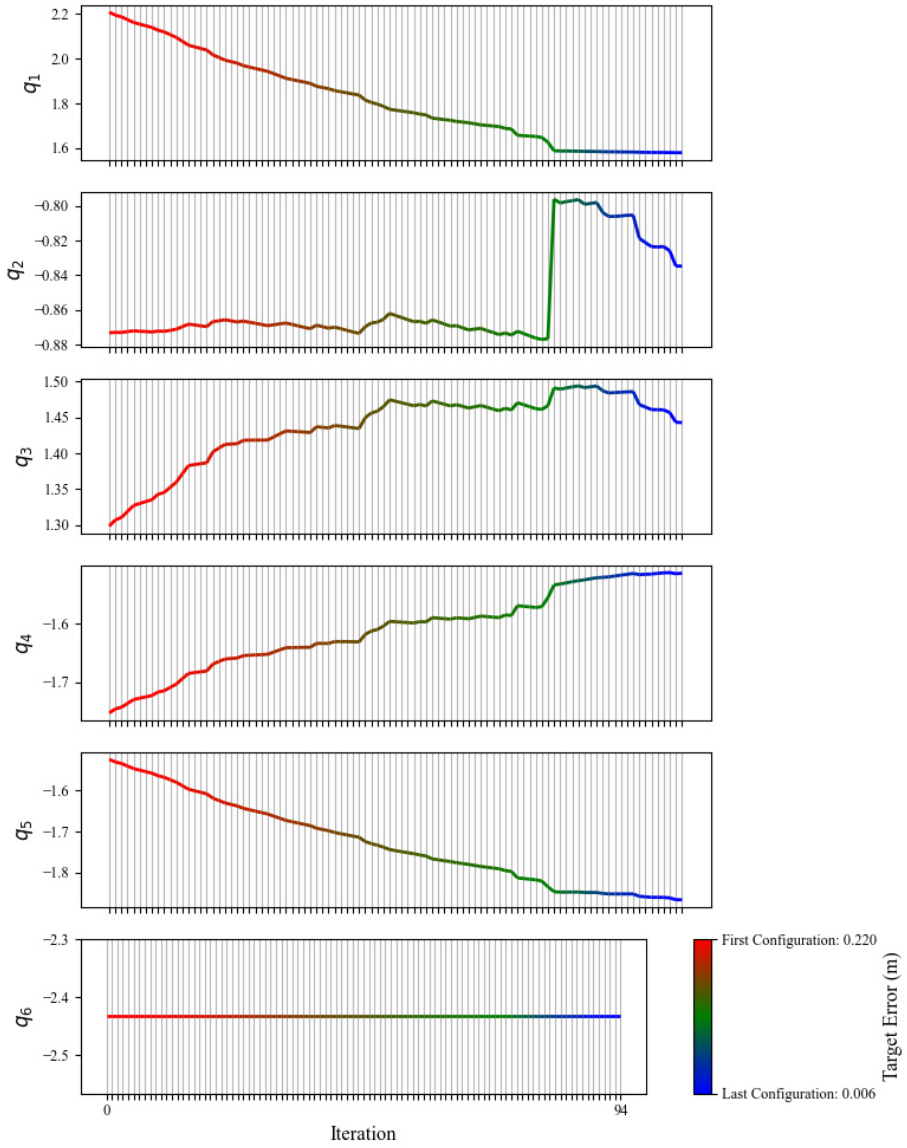


Figure B.2: Third successful evolution of the configuration parameters from iteration 1 to iteration 300 for the fixed target at [1.7, 0.0, 0.0] m. The progression illustrates the iterative adjustment of the joint configuration during ARTS.

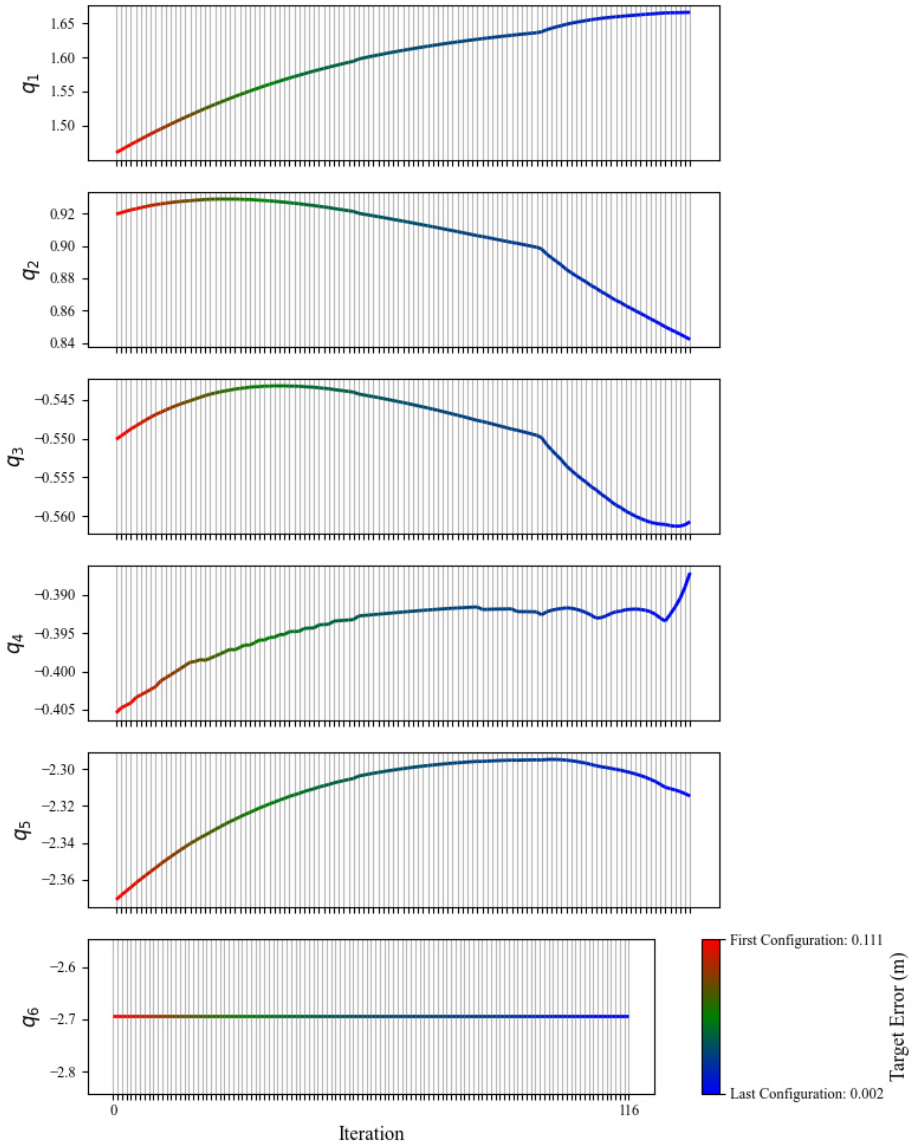


Figure B.3: Fourth successful evolution of the configuration parameters from iteration 1 to iteration 300 for the fixed target at [1.7, 0.0, 0.0] m. The progression illustrates the iterative adjustment of the joint configuration during ARTS.

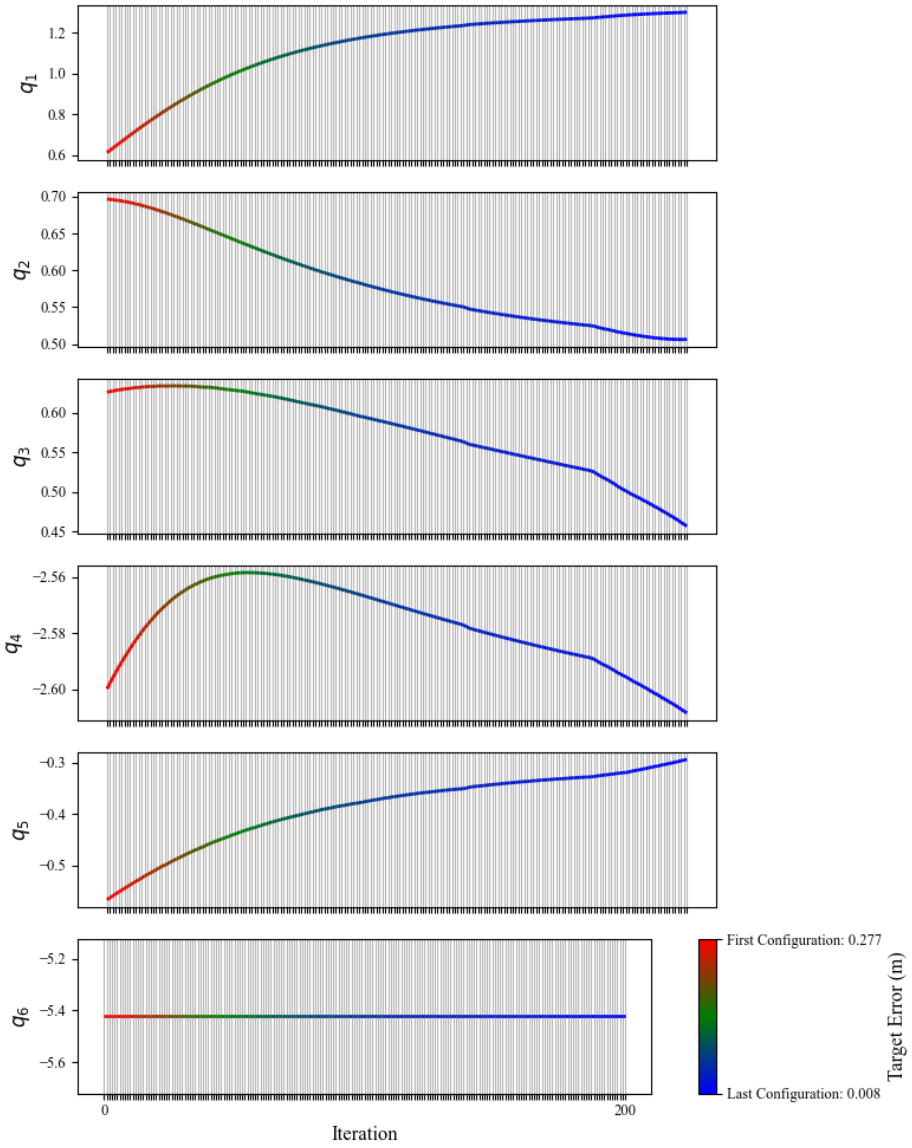


Figure B.4: Fifth successful evolution of the configuration parameters from iteration 1 to iteration 300 for the fixed target at [1.7, 0.0, 0.0] m. The progression illustrates the iterative adjustment of the joint configuration during ARTS.

B.1.2 Visualization of Landing Error at Configurations found with ARTS

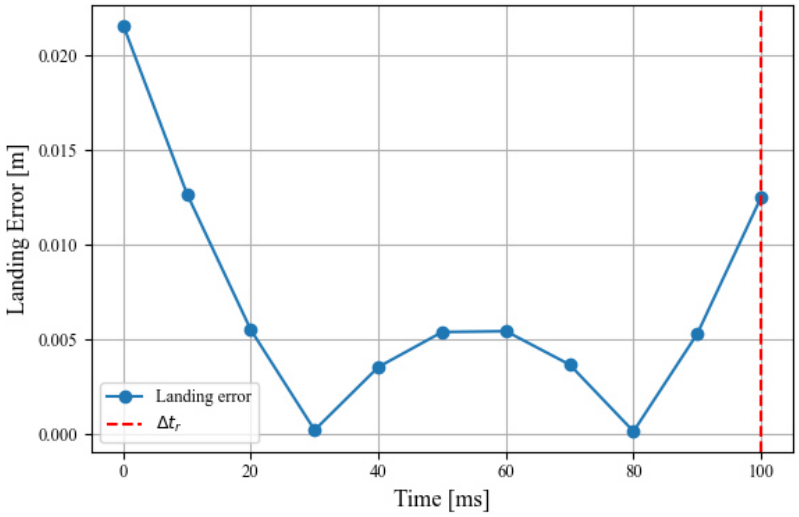


Figure B.5: Second ARTS configuration landing error of the thrown object, evaluated over the release interval Δt_r, for the fixed target at $[1.7,\ 0.0,\ 0.0]$ m.

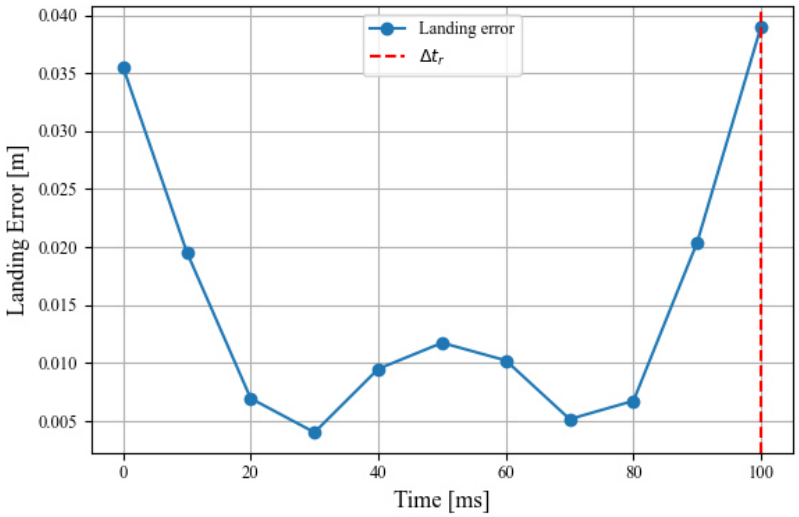


Figure B.6: Third ARTS configuration landing error of the thrown object, evaluated over the release interval Δt_r, for the fixed target at $[1.7,\ 0.0,\ 0.0]$ m.

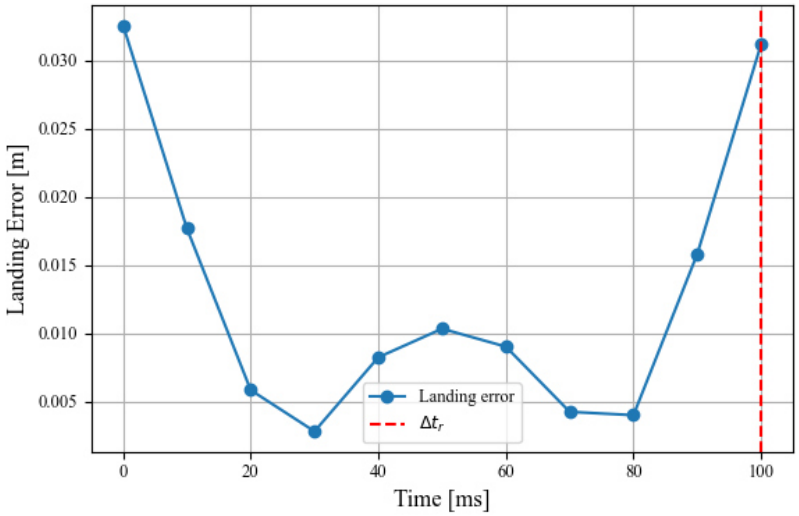


Figure B.7: Fourth ARTS configuration landing error of the thrown object, evaluated over the release interval Δt_r, for the fixed target at $[1.7,\ 0.0,\ 0.0]$ m.

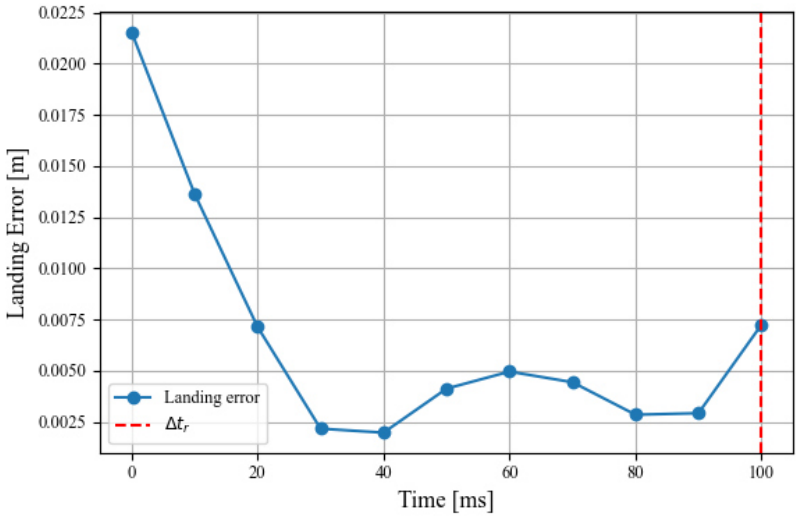


Figure B.8: Fifth ARTS configuration landing error of the thrown object, evaluated over the release interval Δt_r, for the fixed target at $[1.7,\ 0.0,\ 0.0]$ m.

B.1.3 Visualization of Landing Error at Random Configurations

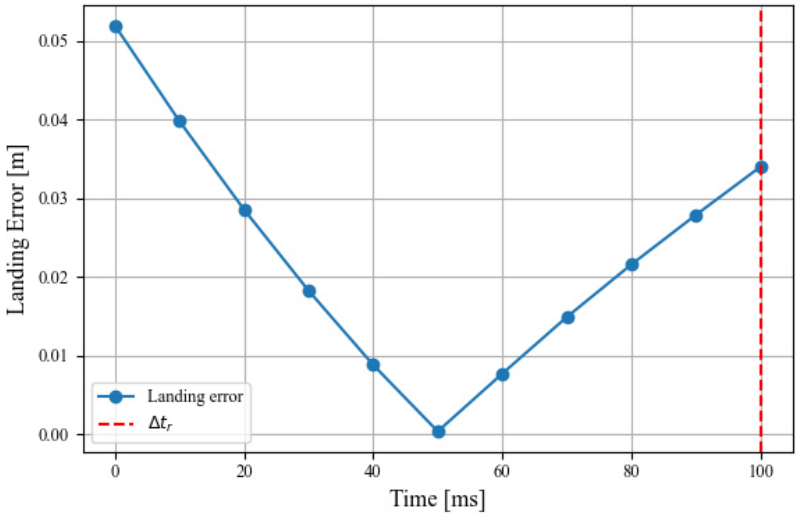


Figure B.9: Second random configuration landing error of the thrown object, evaluated over the release interval Δt_r, for the fixed target at $[1.7,\ 0.0,\ 0.0]$ m.

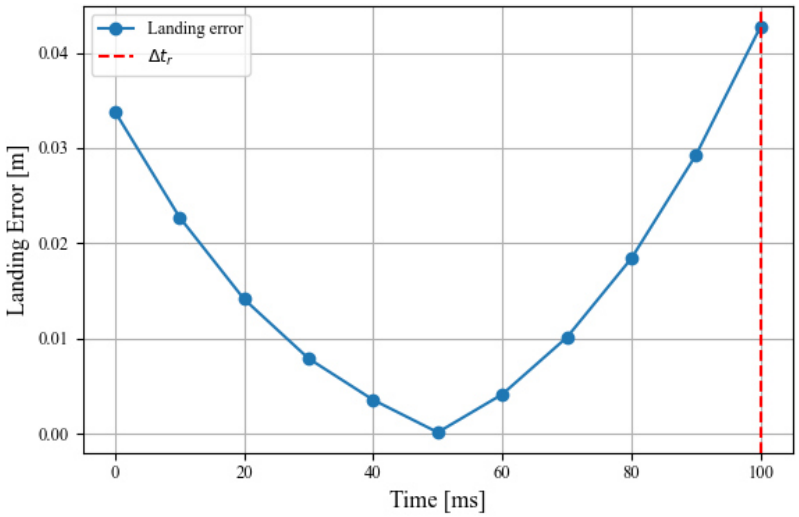


Figure B.10: Third random configuration landing error of the thrown object, evaluated over the release interval Δt_r, for the fixed target at $[1.7,\ 0.0,\ 0.0]$ m.

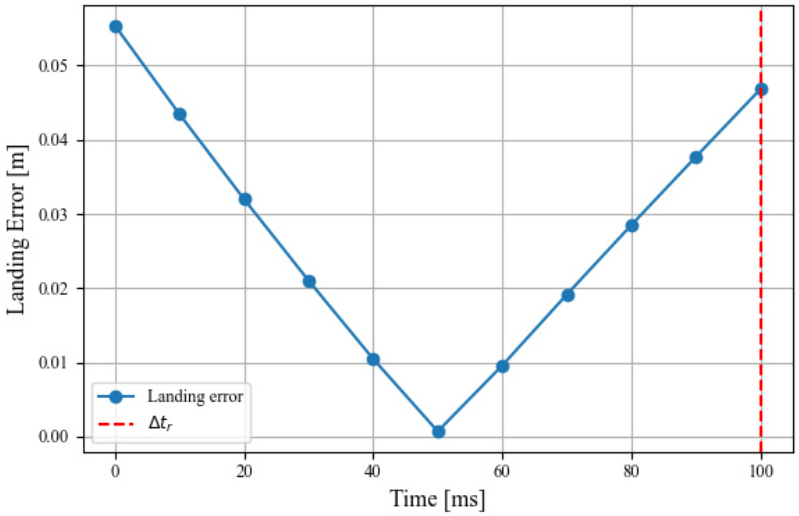


Figure B.11: Fourth random configuration landing error of the thrown object, evaluated over the release interval Δt_r, for the fixed target at $[1.7,\ 0.0,\ 0.0]$ m.

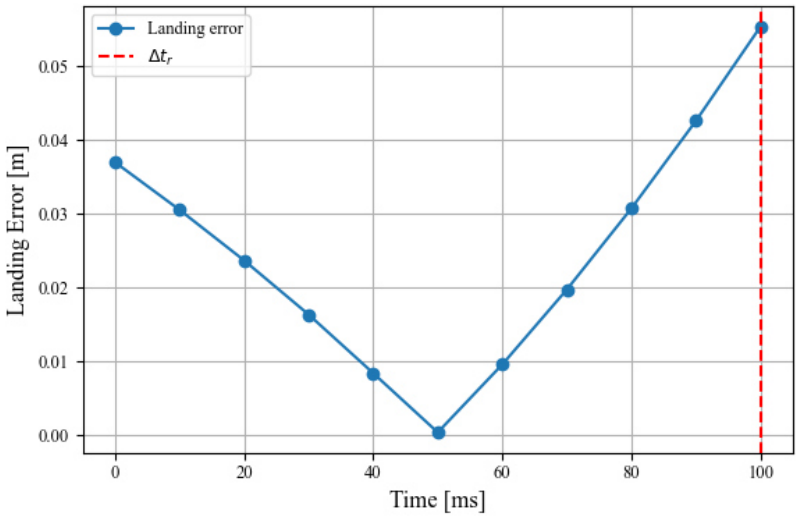


Figure B.12: Fifth random configuration landing error of the thrown object, evaluated over the release interval Δt_r, for the fixed target at $[1.7,\ 0.0,\ 0.0]$ m.

B.2 Performance Evaluation of ARTS

Tables B.1 and B.2 present the results of the throw configuration generation for randomly generated targets, as described in Section 4.4.1.

For ARTS, all configurations for which the target pane is reachable throughout the 100 ms time window are reported. Furthermore for the random search, only the configuration achieving the smallest mean error is shown.

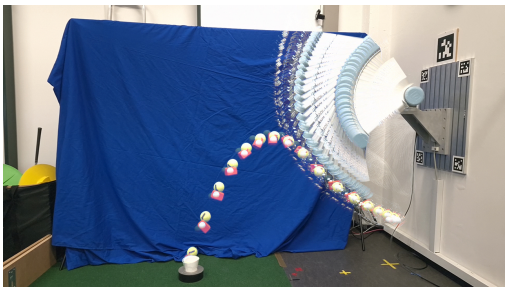
Table B.1: Performance of ARTS for randomly generated targets (120 s per target).

Target [x,y,z] (m)	Status	Configuration	Mean [cm]	Std [cm]
[1.65, 1.02, 1.32]	Not fully reachable	–	–	–
[2.17, 0.66, 1.93]	Empty	–	–	–
[2.47, 0.13, 0.80]	Valid	1	6.0	4.8
[2.21, 0.27, 0.98]	Valid	1	7.5	6.2
[2.12, 0.34, 1.94]	Empty	–	–	–
[1.53, 0.89, 0.32]	Valid	1	0.7	0.5
		2	1.2	0.8
		3	1.2	0.9
[1.58, 0.87, 1.11]	Not fully reachable	–	–	–
[1.70, 0.67, 0.88]	Valid	1	3.0	2.6
		2	3.5	2.9
		3	2.5	2.0
		4	2.9	2.4
[1.87, 0.09, 0.46]	Valid	1	1.9	1.4
		2	1.6	1.2
		3	1.8	1.3
		4	1.5	1.1
[1.29, 1.05, 0.48]	Valid	1	1.3	1.1
		2	1.3	1.0
		3	1.3	1.0

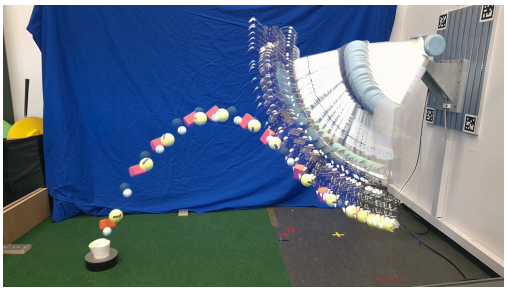
Table B.2: Performance of random search for randomly generated targets (120 s per target).

Target [x,y,z] (m)	Status	Configuration	Mean [cm]	Std [cm]
[1.65, 1.02, 1.32]	Not fully reachable	–	–	–
[2.17, 0.66, 1.93]	Empty	–	–	–
[2.47, 0.13, 0.80]	Valid	Min Mean	15.8	9.6
[2.21, 0.27, 0.98]	Valid	Min Mean	28.6	16.6
[2.12, 0.34, 1.94]	Empty	Min Mean	–	–
[1.53, 0.89, 0.32]	Valid	Min Mean	1.4	1.0
[1.58, 0.87, 1.11]	Valid	Min Mean	14.5	8.5
[1.70, 0.67, 0.88]	Valid	Min Mean	5.2	4.8
[1.87, 0.09, 0.46]	Valid	Min Mean	6.6	6.2
[1.29, 1.05, 0.48]	Valid	Min Mean	4.2	4.0

B.3 Supplementary Figures from Real-World Experiments



(a) Target 1, overlay of four successful throws recorded at 30 fps using four distinct objects.



(b) Target 2, overlay of four successful throws recorded at 30 fps using four distinct objects.



(c) Target 3, overlay of four successful throws recorded at 20 fps using four distinct objects.

Figure B.13: Overlays of throws for Targets 1–3 using various objects. Each subfigure shows the results for a single target.

Appendix C

Use of Generative AI

During the preparation of this thesis, the freely accessible version of GPT-5 mini [OpenAI, 2026] was used to support linguistic formulation, stylistic revision, and to comment and document source code. The content, argumentation structure, and scientific analysis were developed independently.

Tool: ChatGPT (OpenAI)
URL: https://chat.openai.com
Usage period: until March 21, 2026

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Wirtschaftsinformatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe. Sofern im Zuge der Erstellung der vorliegenden Abschlussarbeit generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten Wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

Hamburg, den 21.03.2026

Lukas Sommerhalder