Optimal Control Approach for Non-prehensile Ball Juggling Using a 7-DoF Manipulator

Joel Ramadani[1,*], Vasilije Rakčević[1], Riddhiman Laha[1], Arne Sachtler[1,3], Valentin Le Mesle[1], Achim J. Lilienthal[1], and Sami Haddadin[2]

***Abstract*— Non-prehensile object manipulation skills are important for real-world robot interactions, enabling highly dynamic tasks such as balancing a glass on a tray or the controlled sliding of items on a table. Among such tasks, those characterised by high-speed manipulation requirements and general sensitivity of the resulting hybrid dynamics are particularly hard to accomplish. Within these, juggling can be seen as a highly challenging maneuver to be solved. The key to robotic juggling is achieving dynamic stabilisation of an underactuated object. Since the object does not possess the ability of self-correction, its stability is entirely dependent on the forces applied to it. This creates a system that is sensitive to control inputs, where timing is critical to continuously counteract deviations and maintain the desired behavior. We develop a systematic method to control a 7-degree-of-freedom manipulator performing non-prehensile ball juggling with a tool. Our primary contribution is a model-based framework for generating juggling trajectories and stabilizing a periodic juggling motion for this hybrid system. The framework incorporates a two-stage optimal control approach to compute the underlying feasible motion patterns required for stable juggling. Offline-computed trajectories are then organised to enable real-time error correction without solving optimal control problems online. We demonstrate the effectiveness of the resulting controller by first evaluating its performance in a simulation environment and performing an experiment using a Franka Emika Panda robot.**

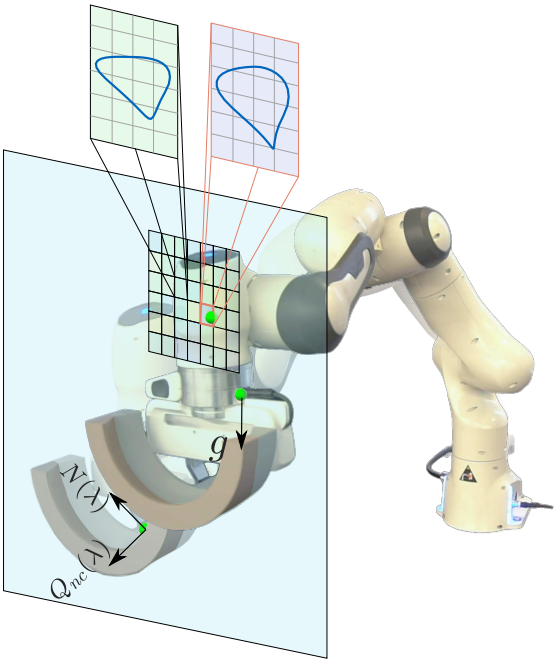

Fig. 1. Visualized juggling setup with a representation of the juggling plane and decision-making policy. Non-conservative force Q_{nc} and Normal force N are modelled in relation to the Lagrange multiplier λ. For the non-contact (flight) phase, ball motion is only influenced by gravity.

I. Introduction

Robot non-prehensile manipulation has been widely studied [1], spanning tasks from quasi-static interactions to highly dynamic behaviors that impose significant hardware and control challenges [2]. Robotic juggling represents one of the most demanding instances, with complexity strongly influenced by the manipulation strategy and physical setup.

One of the most commonly studied juggling tasks in literature is paddle juggling, where one or more balls are repeatedly hit into the air using a paddle (typically a racket or similar object). Here, the ball receives a brief impulse from the paddle during a short contact event that is modelled as an impulse exchange in [3], [4]. When modeled as a discrete nonlinear system, rhythmic paddle juggling can exhibit inherent self-stabilizing properties that allow open-loop execution [5], [6], [7]. However, longer contact duration in juggling based on non-prehensile object manipulation invalidates models based on purely impulsive events. Additionally, the self-stabilising property of these systems is not necessarily inherent for all juggling tasks.

In contrast to paddle juggling, toss juggling requires the ball to stay in contact for a longer duration of time. In robotics, funnel-shaped tools attached to the end-effector of a manipulator are often used to catch the ball. An analytical framework for modelling the task is introduced in [8], [9]. The difficulty of catching strongly depends on the mechanical design, as deep funnel-shaped tools can passively prevent bouncing and reduce the task to trajectory interception. Various trajectory optimization strategies have been proposed to improve catching performance, including energy-efficient motion planning [10], [11] and acceleration matching to reduce sensitivity to impact uncertainties [12], [13]. Learning-based approaches have further been explored to enhance robustness in complex catching scenarios [14]. These approaches primarily focus on accurate interception and impact handling, assuming that the ball is passively retained after contact (caught). In contrast, our task involves prolonged contact with complex dynamics, where the ball must be actively guided.

Beyond catch-toss juggling forms, relevant approaches also include scenarios of a ball that is continuously balanced or guided along surfaces without leaving contact. The first-order kinematic equations describing the contact between two three-dimensional objects are derived in [15]. An optimization framework for planning and controlling first-order rolling interactions between rigid bodies is described in [16].

*Corresponding author: joel.ramadani@tum.de

[1]Technical University of Munich, Munich, Germany.

[2]Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates.

[3]German Aerospace Center (DLR), Institute of Robotics and Mechatronics; Oberpfaffenhofen, Germany.

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. DOI: 10.1109/ICRA57385.2026.11696404

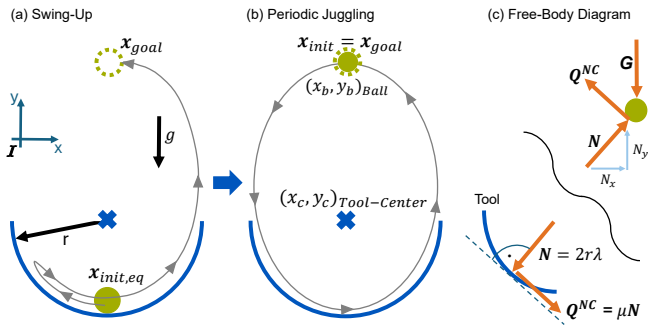

Fig. 2. Planar representation of the juggling task. (a) Swing Up phase; (b) Periodic juggling phase; (c) Free-Body Diagram of the Tool-Ball System

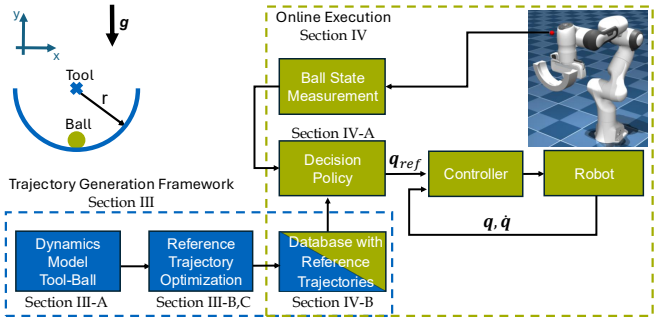

Fig. 3. Block Diagram of the Juggling Control Framework

Modelling interaction between the tool and object, alongside active control throughout the entire contact phase to guide the ball to a suitable launch state is at the core of the tackled problem. However, it's strongly coupled to the flight phase dynamics and thus can not be considered in separation.

Finally, planning methods used in contact juggling [17] are ill-suited for the highly dynamic nature of our task, which requires fast decision cycles and explosive motions to maintain stability. Consequently, real-time trajectory adaptation via online optimization becomes impractical, as solving the underlying Optimal Control Problem (OCP) exceeds the millisecond-scale time budget imposed by the task dynamics. Despite extensive work on impulsive paddle juggling, funnel-based catching, and continuous rolling interactions, control of prolonged-contact tool-based juggling remains largely unexplored.

This motivates the development of a new control architecture that can represent the task's complex hybrid dynamics adapted for real-time, high-speed motion generation.

II. Overview

Our main contribution is a model-based framework for feasible trajectory generation and sustained juggling for non-prehensile ball manipulation. Juggling involves a swing-up phase for launching the ball in the air, followed by controlled juggles as illustrated in Fig. 2. All aspects of the framework can be followed through Fig. 3. We begin by detailing the modelling and hybrid dynamics definition for the tool-ball interaction in Section III-A. Further, a two-stage OCP is introduced in sections III-B and III-C that results in feasible swing-up and juggling trajectories that can be executed on the robot. A scheme for an online error correction via a database of trajectories is introduced in IV. Simulation validation results and robot hardware experiment are presented in V.

A. Ball Juggling Task

Non-prehensile ball manipulation for juggling is performed via a tool (a half-circular tool with a V groove) mounted to the end-effector of a robotic manipulator. Initially, the ball is at rest in a stable equilibrium position of the tool $\boldsymbol{x}_{\text{init,eq}}$. In the first phase, referred to as the swing-up (Fig. 2 (a)), the robot aims to launch the ball into the air and bring it into a desired dynamic state, characterized by a target position and velocity. In particular, within this phase, the tool is actuated so that the ball gains enough energy to exit the tool at its edge. Subsequently, the ball loses contact and enters a flying trajectory, where its motion is only affected by gravity. Once the desired system state $\boldsymbol{x}_{\text{goal}}$ is reached, the objective is to maintain continuous juggling behaviour (Fig. 2 (b)): as the ball starts falling after reaching its apex, it is caught by the tool and guided back into the desired state by coordinated movements of the tool. The target state is defined by the ball's apex state vector $\boldsymbol{x}_{\text{goal}} = [x_b^*, y_b^*, \dot{x}_b^*, \dot{y}_b^*]^{\mathrm{T}}$, where the subscript b indicates the ball and the asterisk $(*)$ denotes the desired value for each state component. This state defines the desired apex of the flight parabola, corresponding to a specific juggling height y_b^* and a horizontal ball velocity $\dot{x}_b^*$. By definition of the apex, the vertical velocity $\dot{y}_b^*$ is zero.

B. Approach

The developed control framework for stable juggling, as illustrated in Fig. 3, relies on a library of pre-computed optimal trajectories for fast online trajectory adaptation. Each trajectory is generated offline in a two-stage process. First, an OCP founded on a two-dimensional model of the tool-ball interaction, computes kinematically ideal motions. Subsequently, a second OCP maps these task-space paths into dynamically feasible joint-space trajectories, ensuring that all of the robot's physical constraints are respected.

To ensure robustness against model inaccuracies and external disturbances, the two-stage optimization process is repeatedly solved offline for various boundary conditions and generates a trajectory database. The database populates the state space around the nominal target apex $\boldsymbol{x}_{\text{goal}}$, providing a library of recovery motions for a variety of perturbed ball states.

In the online phase, the ball's state is measured as it leaves the tool, and its subsequent flight path is estimated via flight dynamics. Based on this estimation, a decision policy selects the most suitable recovery trajectory from the database. The robot's joint-level velocity controller is used for the trajectory tracking. The overall result is a continuous and stable juggling pattern.

III. Trajectory Generation Framework

For the generation of the dynamically feasible juggling trajectories, the problem is decoupled into two main stages, each formulated as a distinct OCP:

1) *Stage 1. Task-Space OCP* – generates an ideal reference trajectory for the tool's motion based on a 2D reference model. The model captures the physics of the ball-tool interaction.
2) *Stage 2. Joint-Space OCP* – takes the 2D tool trajectory and maps it onto the full robot model. The objective here is to find a complete, dynamically feasible joint-space motion that accurately tracks the path from the first stage.

At the core of the OCP lies a computationally efficient formulation of the system dynamics model.

A. Dynamics Model

We begin the analysis by examining the fundamental dynamics of the system: the motion of a ball within a freely movable tool.

1) Assumptions: To ensure mathematical tractability while preserving key physical properties, we introduce the following assumptions:

- *Planar Point-Mass System:* Both tool and ball are constrained to move within the vertical xy-plane. The ball is modelled as a point mass, neglecting rotational dynamics.
- *Mass-less Kinematic Tool:* The tool's dynamics (inertia) are ignored; it serves as an ideal kinematic driver for the ball with a fixed orientation.
- *Unilateral Interaction:* The reaction force from the ball onto the tool is considered negligible due to the small mass ratio (Ball $\ll$ Robot).

These assumptions enable the formulation of a reduced model focusing on the ball's interaction dynamics in the first stage, and on the robot dynamics in the second. Consequently, the impact from the ball is treated as an external disturbance that is compensated by the feedback controller driving the robot.

2) Equations of Motion: Let I^X, I^Y be the inertial Cartesian coordinate system presented in Fig. 2 (a), and (x_c, y_c), (x_b, y_b) be the coordinates of the tool-center and the ball, respectively.

To describe the motion of the ball while in contact with the tool, the Lagrangian formalism [18] is used, extended by a Lagrange multiplier λ to enforce the physical constraint:

$$L = T - U + \lambda C \tag{1}$$

where T and U are the kinetic and potential energy of the ball, respectively, and the term λC enforces the geometric constraint that the ball remains on the tool's surface:

$$r^2 = (x_b - x_c)^2 + (y_b - y_c)^2 \tag{2}$$

With the ball's mass m and the tool's radius r, the specific Lagrangian for the system is given by:

$$L = \underbrace{\frac{1}{2} m \left(\dot{x}_b^2 + \dot{y}_b^2\right)}_{T} - \underbrace{m g y_b}_{U} + \lambda \underbrace{\left((x_b - x_c)^2 + (y_b - y_c)^2 - r^2\right)}_{C} \tag{3}$$

By defining the vector of generalized coordinates as $\boldsymbol{\eta} = [x_b, y_b]^T$, the Euler-Lagrange equations take the general form:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{\eta}}} - \frac{\partial L}{\partial \boldsymbol{\eta}} = \boldsymbol{Q}_{\text{nc}} \tag{4}$$

where the vector of non-conservative forces, given by $\boldsymbol{Q}_{\text{nc}} = [Q_x^{\text{nc}}, Q_y^{\text{nc}}]^T$, is used to model Coulomb friction in the system. Evaluating the partial derivatives of the Lagrangian from Equation (4) yields the following equations of motion:

$$m\ddot{x}_b - \underbrace{2\lambda(x_b - x_c)}_{N_x} = Q_x^{\text{nc}} \tag{5a}$$

$$m\ddot{y}_b + mg - \underbrace{2\lambda(y_b - y_c)}_{N_y} = Q_y^{\text{nc}} \tag{5b}$$

where the non-conservative forces Q_x^{nc} and Q_y^{nc} model the Coulomb friction. They are defined as:

$$Q_x^{\text{nc}} = -\mu \cdot 2\lambda r \cdot \hat{T}_x \cdot \text{sgn}\left(\langle \boldsymbol{v}_{\text{rel}}, \hat{\boldsymbol{T}} \rangle\right) \tag{6a}$$

$$Q_y^{\text{nc}} = -\mu \cdot 2\lambda r \cdot \hat{T}_y \cdot \text{sgn}\left(\langle \boldsymbol{v}_{\text{rel}}, \hat{\boldsymbol{T}} \rangle\right) \tag{6b}$$

where μ is the friction coefficient, $\hat{\boldsymbol{T}}$ is the unit tangent vector at the contact point, and $\boldsymbol{v}_{\text{rel}}$ is the relative velocity between the ball and the tool:

$$\hat{\boldsymbol{T}} = \frac{1}{r}\begin{pmatrix} y_b - y_c \\ -(x_b - x_c) \end{pmatrix}, \qquad \boldsymbol{v}_{\text{rel}} = \begin{pmatrix} \dot{x}_b - \dot{x}_c \\ \dot{y}_b - \dot{y}_c \end{pmatrix} \tag{7}$$

The signum function on the dot product, $\text{sgn}(\langle \boldsymbol{v}_{\text{rel}}, \hat{\boldsymbol{T}} \rangle)$, ensures that the friction force always opposes the tangential component of the relative motion. Note that (5a) and (5b) govern the dynamics only as long as the ball is in contact with the tool. For completeness, the dynamics of the ball during free fall can be described by the following equations:

$$\ddot{x}_b = 0, \qquad \ddot{y}_b = -g. \tag{8}$$

Examining (5a) and (5b) and comparing them to the free fall dynamics in (8), the choice of the Cartesian over the polar coordinate system becomes clear. Substituting $\lambda = 0$ in (5a) and (5b) yields equations that coincide with the free fall dynamics in (8). The presented equations result in a uniform description of the system's hybrid dynamics.

3) The Lagrange Multiplier: The components $N_x = 2\lambda(x_b - x_c)$ and $N_y = 2\lambda(y_b - y_c)$ in (5a) and (5b) define a constraint force vector $\boldsymbol{N} = [N_x, N_y]^T$ (see Fig. 2 (c)). The magnitude of the normal force N exerted by the tool is given by $N = \sqrt{N_x^2 + N_y^2}$. This expression simplifies to $N = 2\lambda r$. Consequently, the Lagrange multiplier λ can be physically interpreted as the normal force scaled by the tool's diameter, i.e., $\lambda = N/2r$.

The value of λ can be determined using the holonomic constraint for keeping the ball movement on the tool's surface, defined in (2). By differentiating this constraint equation twice with respect to time and substituting the equations of motion for $\ddot{x}_b$ and $\ddot{y}_b$ (from (5a) and (5b) without the contribution of the friction forces), one can solve for λ:

$$\lambda = \frac{m}{2r^2}\left[-\|\boldsymbol{v}_{\text{rel}}\|^2 + g(y_b - y_c) + \ddot{x}_c(x_b - x_c) + \ddot{y}_c(y_b - y_c)\right] \tag{9}$$

An interesting property of derived dynamics (5) is that a closed-form solution for components of the normal force N, and subsequently λ from (9), is not required to be included in the dynamics. λ is rather defined as an optimization parameter that abstracts the interaction between the tool and a ball. Unidirectional physical property is encoded via constraint $\lambda(t) \geq 0$.

B. Task-Space Optimal Control for Tool Trajectory Generation

Based on the derived system dynamics, we formulate a *Task-Space OCP* to generate tool motions for both the swing-up and periodic juggling of the ball. The control input $\boldsymbol{u}(t)$ is the tool's Cartesian acceleration, $\boldsymbol{u}(t) = [\ddot{x}_c(t), \ddot{y}_c(t)]^T$, while the state $\boldsymbol{x}(t)$ comprises the positions and velocities of the ball and tool.

A key challenge is the system's hybrid nature, alternating between contact and free fall. We address this by formulating the OCP as a Mathematical Program with Complementarity

Constraints (MPCC), such that the transition is managed by a complementarity condition subject to frictionless contact constraints [19], [20]. The Lagrange multiplier λ is introduced as a decision variable for the complementarity condition in the optimization. This condition is parametrized by a smooth signed distance function $\phi(\boldsymbol{x},\alpha)$, representing log-sum-exp with a smoothing parameter α. It reaches zero when the ball is in contact with the tool's surface.

$$\phi(\boldsymbol{x},\alpha) = \frac{1}{\alpha}\log\left[e^{\alpha(r^2-(x_b-x_c)^2-(y_b-y_c)^2)} + e^{\alpha(y_b-y_c)}\right] \quad (10)$$

With this, the complete OCP, which finds the optimal trajectories and final time t_f, is formulated as:

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\lambda(\cdot),t_f} \quad J = \int_0^{t_f} \|\boldsymbol{u}(t)\|^2 \,\mathrm{d}t \quad (11a)$$

$$\text{such that} \quad \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t),\boldsymbol{u}(t),\lambda(t)), \quad t\in[0,t_f] \quad (11b)$$

$$b(\boldsymbol{x}(0),\boldsymbol{x}(t_f)) = 0 \quad (11c)$$

$$0 \le \lambda(t) \perp \phi(\boldsymbol{x}(t),\alpha) \ge 0,\ t\in[0,t_f] \quad (11d)$$

The objective (11a) minimizes the control effort. While higher-order terms like jerk penalties were evaluated, this formulation yielded improved convergence behavior and smoother acceleration profiles better suited for tracking on the current hardware. The Problem is subject to the system dynamics (11b) and boundary conditions (11c). The core of the hybrid model lies in the complementarity condition (11d), which enforces the contact logic.

1) Solving the Task-Space OCP*:* The OCP is formulated and solved using CasADi [21] within MATLAB. It is transcribed into a nonlinear program (NLP) using a direct collocation method with Gauss-Legendre integration points. The NLP is solved using the interior point solver IPOPT [22] in combination with the linear solver MUMPS. Additionally, the linear solver MA97 from the HSL [23] library was evaluated, and both combinations produced comparable results. However, MA97 showed improved computational speed.

We solve the Task-Space OCP (11) separately for the swing-up motion and the periodic juggling motion, since the transition point is predefined by the desired ball apex state. The main difference in formulations lies in the boundary conditions (11c): For the swing-up, the ball starts in the equilibrium position $\boldsymbol{x}_{\text{init,eq}}$ and reaches $\boldsymbol{x}_{\text{goal}}$ as terminal boundary condition (cf. Fig. 2a), whereas for the periodic juggling task, $\boldsymbol{x}_{\text{goal}}$ is imposed as boundary condition at both the beginning and the end of the trajectory (cf. Fig. 2b).

No path constraints are imposed on the tool's velocity or acceleration. Instead, only the Lagrange multiplier $\lambda(t)$ is bounded, which implicitly constrains the tool accelerations through the ball-tool interaction (cf. (9)). This choice showed an improved numerical convergence behaviour and also allows enforcing smoother catches as an upper bound on λ effectively limits the initial impact force when the ball is caught by the tool.

C. Joint-Space OCP *for Feasible Reference Trajectory Tracking*

Since the task requires the robot arm for tool manipulation, its limits have to be taken into account. For transferring generated 2D task-space trajectories to a Franka Emika Panda [24] robot, a standard Jacobian-based Cartesian controller is not sufficient, because the trajectories for both the swing-up and the juggling are highly dynamic. Additionally, kinematic constraints require the tool to stay on a 2D task space, which can introduce additional difficulties for Cartesian speed tracking. Thus, to translate the planar reference trajectory into physically feasible joint trajectories for the 7-DoF robot, a second OCP (12) was defined to incorporate the full system dynamics and actuation constraints.

The process begins by preprocessing the trajectories from the initial 2D juggling model. The time intervals where the ball is in contact with the tool are identified by using the Lagrange multipliers associated with the contact constraints of the task-space OCP. This is done because precise trajectory tracking is only required while the ball is in contact with the tool. The adopted 2D tool path is then embedded in the robot's 3D workspace as $\boldsymbol{p}_{des}$. To find a suitable workspace region characterised by high agility, a manipulability analysis was performed on the Franka Panda. The forward kinematics of this configuration define the initial end-effector pose, to which the 2D trajectory is rigidly attached.

The *Joint-Space OCP* is formulated to find a feasible joint-space trajectory that follows the desired tool trajectories, while simultaneously maintaining the tool in a desired orientation.

$$\text{find} \quad \boldsymbol{x_r}(\cdot),\ \boldsymbol{u_r}(\cdot)$$

$$\text{subject to} \quad \dot{\boldsymbol{x}}_r(t) = f(\boldsymbol{x_r}(t),\boldsymbol{u_r}(t)), \quad t\in[t_{\text{init}},t_{\text{goal}}] \quad (12a)$$

$$\boldsymbol{q}_{\min} \le \boldsymbol{q}(t) \le \boldsymbol{q}_{\max} \quad (12b)$$

$$|\boldsymbol{u}_r(t)| \le 0.9\ \boldsymbol{u}_{r,\max} \quad (12c)$$

$$|\dot{\boldsymbol{u}}r(t)| \le 0.9\ \dot{\boldsymbol{u}}_{r,\max} \quad (12d)$$

$$\boldsymbol{p}_{\text{ee}}(\boldsymbol{q}(t)) = \boldsymbol{p}_{\text{des}}(t), \quad \forall t\in T_c \quad (12e)$$

$$\boldsymbol{R}_{\text{ee}}(\boldsymbol{q}(t)) = \boldsymbol{R}_{\text{des}}, \quad \forall t\in T_c \quad (12f)$$

$$\boldsymbol{x_r}(t_{\text{init}}) = \boldsymbol{x_r}(t_{\text{goal}}) \quad (12g)$$

Here, $\boldsymbol{x_r} = (\boldsymbol{q},\dot{\boldsymbol{q}})$ represents the robot's joint states. The path constraints for joint torques $\boldsymbol{u}_r$ (12c) and their rate of change $\dot{\boldsymbol{u}}_r$ (12d) were chosen, as these are the most versatile limits specified by the Franka Control Interface (FCI). The torque limits in the OCP were set to 90% of the robot's maximum capabilities to ensure a feedback controller can react to tracking errors and external disturbances.

The dynamics constraint (12a) encodes the standard robot equations of motion:

$$\dot{\boldsymbol{x}}_r = \begin{bmatrix} \dot{\boldsymbol{q}} \\ M(\boldsymbol{q})^{-1}\left(\boldsymbol{u}_r - C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - g(\boldsymbol{q})\right) \end{bmatrix}, \quad (13)$$

where $M(\boldsymbol{q})$ is the joint-space mass matrix, $C(\boldsymbol{q},\dot{\boldsymbol{q}})$ contains the Coriolis and centrifugal terms, and $g(\boldsymbol{q})$ represents gravity. We constrain the forward kinematics of the robot to track the desired tool trajectory by enforcing the position and orientation constraints (12e)–(12f) only during the contact phases T_c. This gap leaves the initial and final segments of the trajectory unconstrained and provides the optimizer with more flexibility to plan smooth transitions between juggling cycles.

The *Joint-Space OCP* (12) is again formulated as a NLP using direct collocation and the CasADi [21] framework. To improve solving speed, the Pinocchio [25] library was adopted, as it provides faster differentiation of the complete robot dynamics and its integration with CasADi ensures highly efficient symbolic processing.

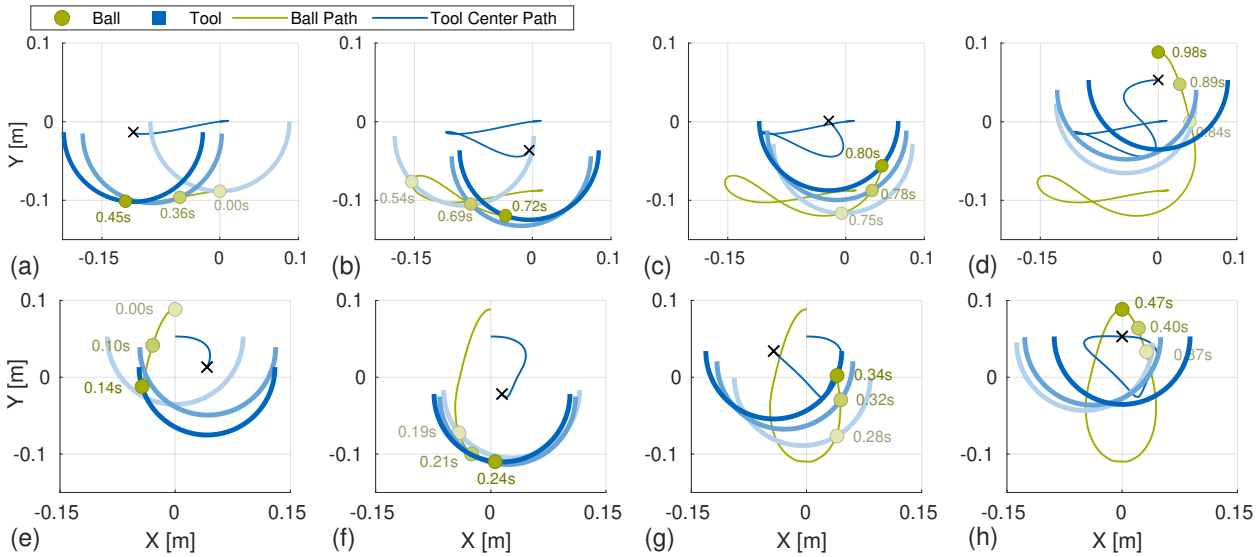

Fig. 4. Snapshots of a reference trajectory obtained from the *Task-Space OCP*. (a-d) Swing-up motion of the tool and ball; (e-h) Periodic juggling cycle.

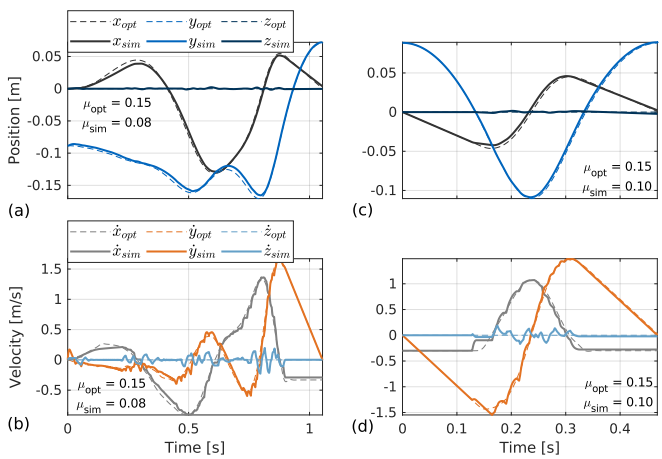

Fig. 5. Comparison of the optimal and simulated ball trajectories for the swing-up (a-b) and juggling (c-d) maneuvers.

IV. Stabilization of the Juggling Motion

A. Decision Policy

A Poincaré analysis [26] confirms that the optimized trajectories do not exhibit self-stabilizing properties under open-loop execution, reflecting the inherent instability of the system. Thus, they are not sufficient for sustained juggling: the ball tends to fall out of the tool after a few cycles when executing the same tool trajectory periodically. To address this, we introduce a feedback mechanism that adapts the tool's motion once per cycle based on the observed ball state.

For deciding on a trajectory for the next cycle, a ball state at t_{release} is measured. The ball's position and velocity $\tilde{\boldsymbol{x}}_b$ are used to predict its future state at the apex $\hat{\boldsymbol{x}}_b$ of its parabolic flight path (see Fig. 6). This predicted apex state is then matched to the closest entry in the pre-computed trajectory database, using the minimum Euclidean distance d_{min}, and the corresponding tool trajectory is selected for the upcoming cycle. By estimating the apex state immediately at release, the system can utilize the flight phase to execute a smooth, continuous transition to the newly selected trajectory before the ball needs to be caught again.

B. Database generation

To generate the database, the *Task-Space OCP* is solved repeatedly in a loop. Since the goal is to correct ball trajectory deviation by bringing the ball back to the correct apex (see Fig. 6), the database contains trajectories associated with all perturbed apex states of the ball (a square grid of apex positions spanning 5 cm $\times$ 5 cm around the desired apex, with a resolution of 3.125 mm) as the initial boundary condition. The terminal boundary is fixed at the desired apex. For each grid point, several candidate horizontal velocities are assigned. The expected velocity v_{exp} at the apex is estimated as a linear function of the horizontal offset, such that balls reaching their apex left of the desired point are associated with higher horizontal velocities than those reaching it on the right (see Fig. 6). This ensures that the database captures the velocity error trend for counterclockwise ball motion. Indeed, for the fixed release position, the balls that end up further left have higher horizontal velocity at the apex.

Each consecutive solution (ball and tool trajectories) takes the previous iteration as a warm start for the current problem. This approach aims to prevent the solutions from drifting apart over iterations. By drifting, we mean that, due to the non-convexity of the problem, multiple distinct solutions exist that successfully return the ball to the desired state. For instance, it was observed that some solutions resulted in the ball being caught with prolonged contact times, differing significantly from the solution presented in Fig. 4b.

To map each of these reference trajectories to the full robot model and ensure their dynamic feasibility, *Joint-Space OCP* is utilized. First, a single trajectory with only a periodic boundary condition, $\boldsymbol{x}_{r,\text{init}} = \boldsymbol{x}_{r,\text{goal}}$ is solved. Its solution establishes a feasible common reference state $\boldsymbol{x}_{r,\text{apex}}$ at t_{apex} for the robot. For all subsequent trajectories in the database *Joint-Space OCP*s are then solved with modified

TABLE I
Model Parameters for Trajectory Database

Parameter	Symbol	Value
Gravitational acceleration	g	$9.81\,\text{m}\,\text{s}^{-2}$
Ball mass	m	0.1 kg
Tool radius	r	0.088 45 m
Tangential friction coeff.	μ	0.17

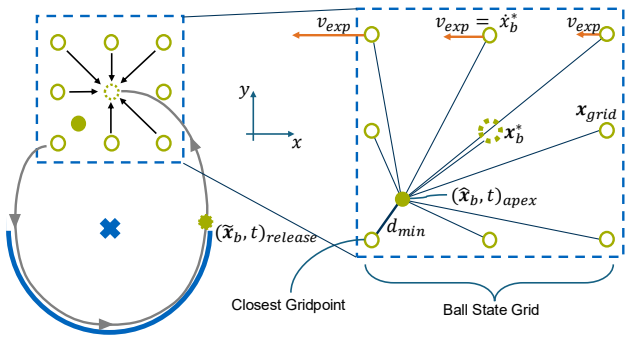

Fig. 6. Representation of the trajectory database combined with the decision policy to select the next trajectory to perform. The ball state is measured at release, and its apex state is estimated. Based on this estimation, a suitable trajectory is retrieved from the database.

boundary constraints to match the initially found reference state: $\boldsymbol{x}_{r,\text{init}} = \boldsymbol{x}_{r,\text{apex}}$ and $\boldsymbol{x}_{r,\text{goal}} = \boldsymbol{x}_{r,\text{apex}}$.To ensure a feasible transition between different juggling trajectories from the database, the robot's trajectories are designed to start and finish in the same joint state for all trajectories in the database. This setup is a prerequisite for implementing the periodic controller in Section IV-C.

C. Trajectory Transitioning via Time Reparameterization

As each control cycle starts and ends in the ball apex position, which is also associated with the robot's dynamic state that ensures a smooth transition between cycles, timing deviations require an additional correction strategy. Namely, the robot must reach its target state exactly when the ball reaches its apex. However, disturbances and model inaccuracies can cause the ball's actual trajectory to deviate from the planned trajectory, often resulting in a discrepancy between predicted and actual time point for reaching apex t_{goal}. To prevent the resulting asynchrony, an adaptive mechanism is implemented to rescale the tool's trajectory timing, effectively ensuring that the tool can start the next trajectory as soon as the ball reaches its apex t_{goal} . This online adaptation of the motion's temporal execution is enabled by a path-velocity decomposition method [27]. Given the optimal control solution as a discrete-time joint trajectory $\{\vec{q}_{\text{orig}}(t_i), \dot{\vec{q}}_{\text{orig}}(t_i)\}$, the solution is decomposed into a time-independent geometric path, $\vec{q}_{\text{path}}(s)$. This path is parameterized by its arc length $s \in [0, s_{\text{total}}]$ and is represented by a cubic spline interpolated through the original joint configuration waypoints. During online execution, when the ball is launched at t_{release} and the apex time $t_{\text{to_apex}}$ is estimated, the trajectory's duration is reparameterized to a new end time t_{apex}. This is achieved by computing a new, smooth timing law, $s_{\text{new}}(t)$, as the solution to a boundary value problem for the interval $[t_{\text{release}},\ t_{\text{apex}}]$. This ensures C^1-continuity at the transition point t_{release}. The four boundary conditions are defined as:

$$s_{\text{new}}(t_{\text{release}}) = s_{\text{orig}}(t_{\text{release}}), \qquad s_{\text{new}}(t_{\text{apex}}) = s_{\text{total}}$$
$$\dot{s}_{\text{new}}(t_{\text{release}}) = \|\dot{\vec{q}}_{\text{orig}}(t_{\text{release}})\|, \qquad \dot{s}_{\text{new}}(t_{\text{apex}}) = \|\dot{\vec{q}}_{\text{start}}\|$$

where $s_{\text{orig}}(t)$ is the timing law of the original trajectory and $\dot{\vec{q}}_{\text{start}}$ is the joint velocity component of the robot's apex state $\boldsymbol{x}_{r,\text{apex}}$. This boundary value problem is solved using a piecewise cubic Hermite [28] polynomial, which yields a unique and continuously differentiable solution for $s_{\text{new}}(t)$.

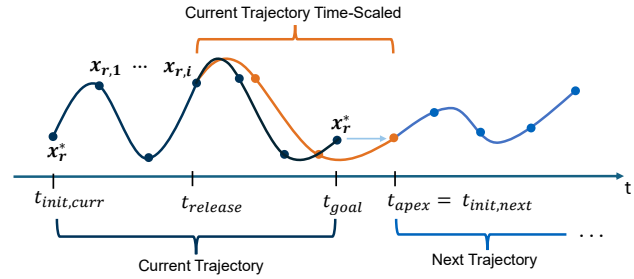

Fig. 7. Path-Preserving Trajectory Re-Timing. At t_{release}, the tool trajectory is re-scheduled on a new time scale, in case the calculated apex state of the tool (based on the ball state at detachment point) $\boldsymbol{x}_{r,\text{apex}}$ is different from expected one associated to the current tool trajectory. This also means that the current estimated time for reaching the apex t_{apex} would be shifted from the originally planned one t_{goal}. Note that the new trajectory execution begins at t_{apex}.

The new desired state for the robot is then synthesized from this new timing law. The desired joint position vector $\vec{q}_{\text{des}}(t)$ is obtained by evaluating the geometric path $\vec{q}_{\text{path}}$ at the newly timed arc length:

$$\vec{q}_{\text{des}}(t) = \vec{q}_{\text{path}}(s_{\text{new}}(t)). \tag{14}$$

The corresponding desired joint velocity vector $\dot{\vec{q}}_{\text{des}}(t)$ is computed via the chain rule:

$$\dot{\vec{q}}_{\text{des}}(t) = \frac{d\vec{q}_{\text{path}}}{ds}(s_{\text{new}}(t)) \cdot \dot{s}_{\text{new}}(t). \tag{15}$$

The resulting vectors $\vec{q}_{\text{des}}(t)$ and $\dot{\vec{q}}_{\text{des}}(t)$ serve as the reference signals for the low-level controller. This ensures that the original geometric path is followed while achieving a C^1-continuous transition between trajectory cycles.

V. Experiments

A. Model Validation

An evaluation of the ball's dynamic response to the OCP-generated tool trajectories is performed in MuJoCo [29] physics engine by simulating the direct interaction between the tool and the ball. Manipulator dynamics is omitted to isolate errors related to interaction behaviour only (e.g. ball deformation, rolling, etc.). Since our model operates in two dimensions, a tool was designed with a V-shaped groove to better align the simulated 3D system with the modelling assumptions by restricting the ball's movement.

As shown in Fig. 5, the best match between the optimal and simulated trajectories is achieved when the simulation's friction coefficient is set lower than the model's. This suggests that the friction coefficient in our model acts as a lumped parameter, compensating for multiple energy dissipation phenomena present in the more realistic simulation. These phenomena include, for example, the increased effective friction from the V-shaped tool's double contact points and energy losses due to ball rolling.

B. Controller Performance Analysis

The controller's sensitivity to the contact model was analyzed via a parameter sweep across a range of tangential friction coefficients and solref damping ratios (which determines the elasticity of the impact), with the results presented in Fig. 8a. Each simulation run began with a standardized swing-up maneuver, at which point the closed-loop juggling controller took over. The stability for each parameter configuration was then measured by the total

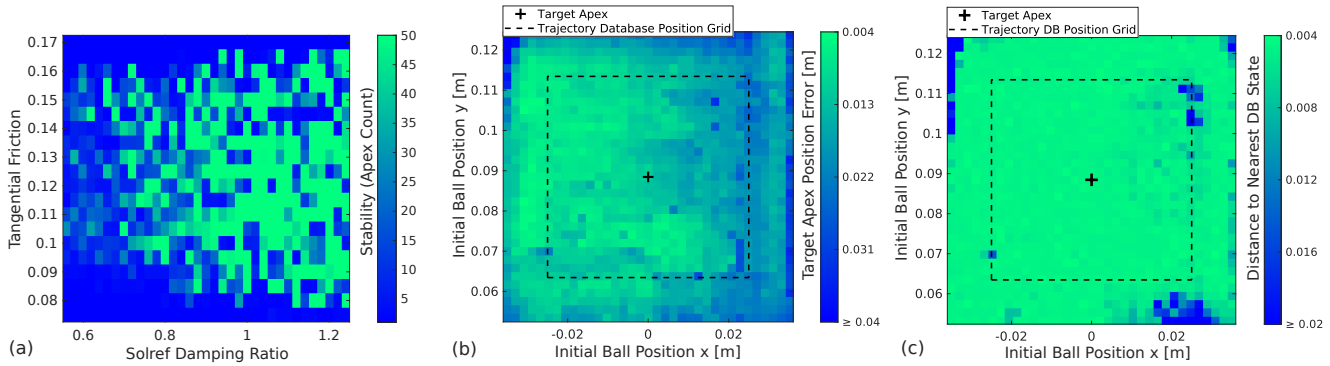

Fig. 8. Mapping the performance landscape of the juggling controller by analyzing the ball's resulting motion under variations in physical parameters and initial conditions. (a) Stability map of the control approach across different contact parameters. All parameters except for the tangential friction and damping ratio were kept at their MuJoCo default values. The color of each cell indicates the stability, measured by the maximum number of successful juggles (apex count) initiated from that parameter set; (b) Accuracy map of the worst-case positional error (Euclidean distance in x and y) between the achieved apex and the single nominal target apex. Each cell's value represents the maximum error found over 50 random trials initiated from that cell; (c) Robustness map of the worst-case state error (Euclidean distance in position and velocity) between the achieved apex and the nearest corresponding state in the pre-computed trajectory database.

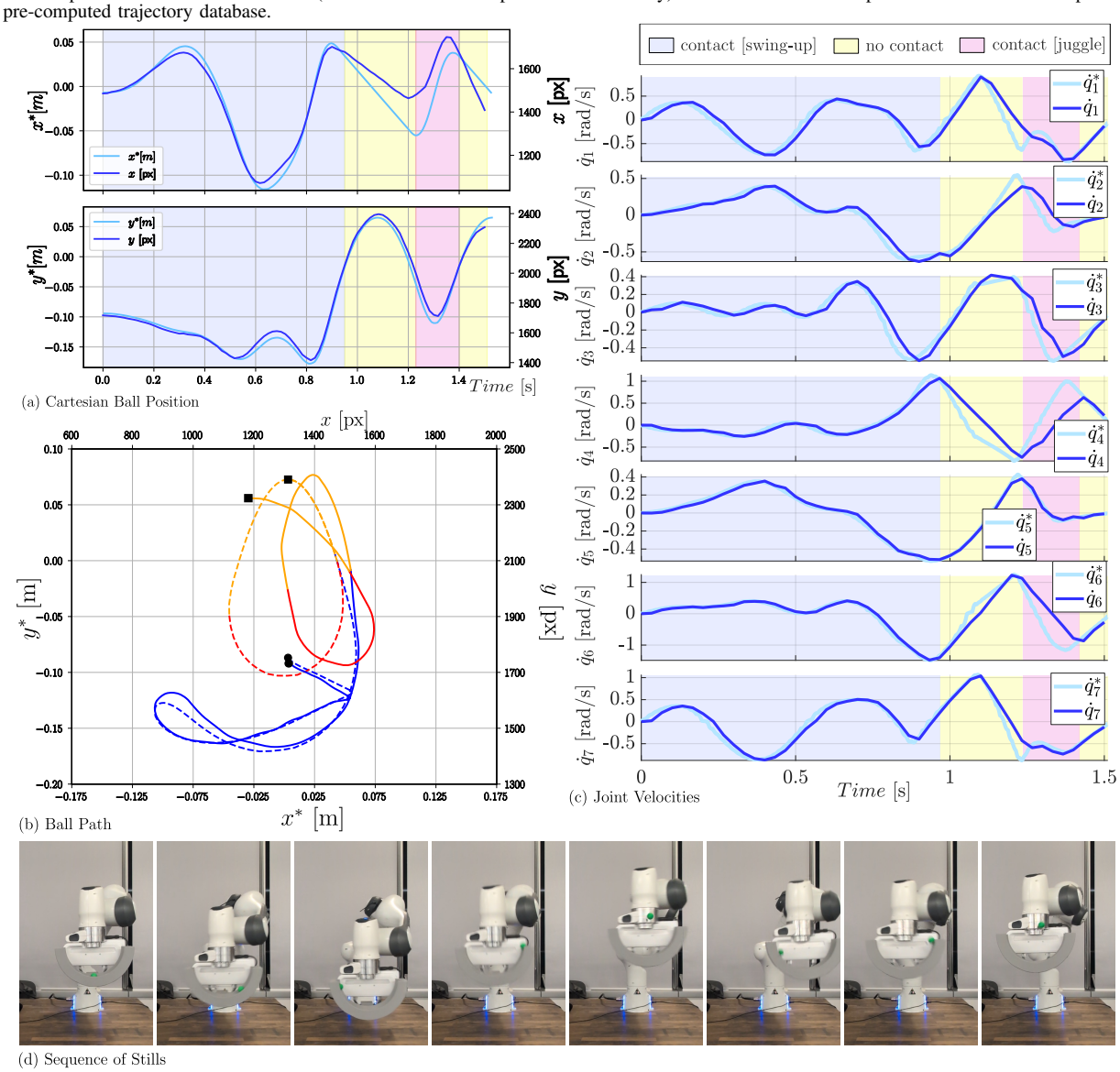

Fig. 9. Experiment performed on Panda Franka Emika Robot for swing up and one cycle of juggling using a velocity controller for trajectory tracking. The experimentally evaluated friction coefficient of $\mu = 0.105$ is used for trajectory generation. a) and b) show the ball position in Cartesian space. The actual ball trajectory is extracted from a video recording. c) shows the tracking performance of the joint-level velocity controllers; and d) shows a sequence of stills of swing up and once cycle of juggling.

number of consecutive juggles (apex count) the system could sustain before the ball was dropped.

Using a stable set of contact parameters identified in the previous analysis, we then performed a grid-based Monte Carlo simulation to quantitatively evaluate the controller's robustness. A 7x7 cm grid of initial apex positions, centered around the nominal target state, was discretized into 2x2 mm cells. Within each cell, 50 initial ball states were randomly sampled from a uniform distribution, and horizontal velocities were sampled from a dynamic range based on the cell's x-position. For each sampled initial state, the decision policy selected the optimal recovery trajectory from the database, and a single juggle cycle was simulated in MuJoCo. From the recorded final apex states, the worst-case performance of each cell was evaluated and visualized in terms of target deviation (Fig. 8b) and from the point of view of robustness by the ability to choose a reasonably close database point with a recovery trajectory in Fig. 8c.

C. Hardware Experiments

For the hardware experiments, the circular tool was 3D-printed and attached to the robot's gripper. To calibrate our contact model, an initial system identification step was performed. We executed a series of swing-up trajectories and used external camera recordings to measure the resulting flight path of the ball, allowing us to empirically determine the effective friction coefficient of the physical tool-ball pair. With this calibrated friction parameter, a trajectory consisting of a swing-up and one subsequent juggling maneuver was generated. This trajectory was then mapped to the joint space and executed on the robot. A stop-motion of the experiment, data on controller performance, alongside the resulting ball trajectory, can be seen in Fig. 9. Experiments are limited to a swing-up and single juggle cycle for evaluating the presented model-based framework and robot performance abilities.

VI. Conclusion

Highly dynamic and timing-sensitive systems such as non-prehensile robot juggling require an online strategy to counteract ball deviations and maintain the desired behavior. For that, a model-based two-stage optimal control framework has been presented. In the first stage, the *Task-Space OCP* generates juggling motions from an analytical 2D model. In the second stage, *Joint-Space OCP* takes the resulting 2D trajectories and maps them to dynamically feasible joint-space trajectories. This process is repeatedly solved offline to generate a library of pre-computed correction trajectories for the predicted perturbed apex state range. A decision policy is then deployed online. It takes the ball state and determines the correction trajectory to be executed next. To ensure a smooth transition between different DB trajectories, a trajectory re-parametrisation approach has been adopted. Performance was evaluated in both simulation and hardware experiments using the Franka Panda robot with a circular tool and a ball to perform swing-up and juggling tasks.

Acknowledgement

The authors would like to thank Bingkun Huang for fruitful discussions and help with the hardware experiments. This work has partially been supported by the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG), grant no. 16ME0997K.

References

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

[2] R. I. C. Muchacho, R. Laha, L. F. Figueredo, and S. Haddadin, "A solution to slosh-free robot trajectory optimization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[3] M. Buehler and D. E. Koditschek, "Robotics in an intermittent dynamical environment: A prelude to juggling," 1987.

[4] E. W. Aboaf, C. G. Atkeson, and D. J. Reinkensmeyer, "Task-level robot learning," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*. IEEE, 1988, pp. 1309–1310.

[5] S. Schaal, C. G. Atkeson, and D. Sternad, "One-handed juggling: A dynamical approach to a rhythmic movement task," *Journal of Motor Behavior*, vol. 28, no. 2, pp. 165–183, 1996.

[6] S. Schaal and C. G. Atkeson, "Open loop stable control strategies for robot juggling," in *Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 913–918.

[7] P. Reist and R. D'Andrea, "Design and analysis of a blind juggling robot," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1228–1243, 2012.

[8] P. J. Beek and A. Lewbel, "The science of juggling," *Scientific American*, vol. 273, no. 5, pp. 92–97, 1995.

[9] B. Polster, *The mathematics of juggling*, 2003.

[10] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3719–3726.

[11] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2592–2599.

[12] B. Hove and J.-J. E. Slotine, "Experiments in robotic catching," in *1991 American Control Conference*. IEEE, 1991, pp. 380–386.

[13] A. Namiki and N. Itoi, "Ball catching in kendama game by estimating grasp conditions based on a high-speed vision system and tactile sensors," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 634–639.

[14] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.

[15] D. J. Montana, "The kinematics of contact and grasp," *The International Journal of Robotics Research*, vol. 7, no. 3, pp. 17–32, 1988.

[16] J. Z. Woodruff, S. Ren, and K. M. Lynch, "Motion planning and feedback control of rolling bodies," *IEEE Access*, vol. 8, 2020.

[17] J. Z. Woodruff and K. M. Lynch, "Robotic contact juggling," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1964–1981, 2023.

[18] J. L. Lagrange, *Mécanique analytique*. Mallet-Bachelier, 1853, vol. 1.

[19] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[20] H. Yao, R. Laha, A. Sinha, J. Hall, L. F. Figueredo, N. Chakraborty, and S. Haddadin, "On the synthesis of reactive collision-free whole-body robot motions: A complementarity-based approach," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*.

[21] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, 2019.

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

[23] A. HSL, "Collection of fortran codes for large scale scientific computation," *See http://www. hsl. rl. ac. uk*, 2007.

[24] S. Haddadin, S. Parusel, L. Johannsmeier, S. Golz, S. Gabl, F. Walch, M. Sabaghian, C. Jähne, L. Hausperger, and S. Haddadin, "The franka emika robot: A reference platform for robotics research and education," *IEEE Robotics & Automation Magazine*, vol. 29, no. 2, pp. 46–64, 2022.

[25] J. Carpentier, G. Saurel, G. Buondonno *et al.*, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[26] J. K. Ramadani, "Robust optimal control approach for stable ball juggling using a 7-dof manipulator," Master's thesis, Technical University of Munich, 2025. [Online]. Available: https://mediatum.ub.tum.de/1843356

[27] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[28] W. Gautschi, *Numerical analysis*. Springer Science & Business Media, 2011.

[29] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.