Bimanual dynamic grabbing and tossing of objects onto a moving target

Michael Bombile*, Aude Billard

Learning Algorithms and Systems Laboratory (LASA), Ecole Polytechnique Federale de Lausanne (EPFL), Lausanne, 1015, Vaud, Switzerland

ARTICLE INFO

Keywords:
dual-arm systems
grabbing and tossing
moving target
dynamical systems

ABSTRACT

Bimanual grabbing and tossing of packages onto trays or conveyor belts remains a human activity in the industry. For robots, such a dynamic task requires coordination between two arms and fast adaptation abilities when the tossing target is moving and subject to perturbations. Thus, this paper proposes a control framework that enables a bimanual robotic system to grab and toss objects onto a moving target. We develop a mixed learning-optimization method that computes the tossing parameters necessary to achieve accurate tossing tasks. Hence, we learn an inverse throwing map (a closed-form solution of the inverse non-linear throwing problem) that provides minimum release velocities of the object for given relative release positions. This map is embedded into a kinematics-based bi-level optimization that determines the associated feasible release states (positions and velocities) of the dual-arm robot. Additionally, we propose a closed-form modeling approach of the robot's tossable workspace (set of all positions reachable by an object if tossed by the robot) and use the model to predict intercept or landing locations that yield high probabilities of task success. Furthermore, we employ dynamical systems to generate the coordinated motion of the dual-arm system and design an adaptation strategy to ensure robustness of the interception in the face of target's perturbations in speed or location. Finally, we validate experimentally the framework on two 7-DoF robotic arms. We demonstrate the accuracy and robustness of the proposed approach. We also show its speed and energy advantages when compared to the traditional pick-and-place strategy.

1. Introduction

Recently, there has been an increasing need for robotic solutions in the logistics industry to address challenges related to the booming of e-commerce. The current workforce, although having better dexterity and flexibility than automated solutions, cannot keep up with the industry demands as the need for faster package handling solutions continues to increase [1, 2]. In classical robotic solutions such as pick-and-place, robots usually use a quasi-static approach (with near zero relative velocities) for picking up and releasing products, mainly to avoid impacts. In contrast, humans often use dynamic manipulation approaches and can safely interact with objects at non-zero relative contact velocities. Humans can quickly grab an object by snatching it and pass it along by throwing or tossing it. For instance, in depalletizing or sorting facilities, it is common to find humans picking up and tossing objects on trays or moving conveyor belts. Similarly, robotic pick-and-toss has been proposed as a faster and more energy efficient dynamic alternative to pick-and-place operations [3, 4]. Additionally, robotic throwing or tossing offers the benefit of extending the robot's reach beyond its physical boundaries [5]. For example, in waste sorting applications, the potential of pick-and-toss was demonstrated on a bimanual robot in [6] where the arms working independently and with no physical interaction between them use suction and blow mechanism to achieve the task or a Delta parallel robot in [3] and [7]. While in many applications, as in the aforementioned works, the picking up of objects can be realized with a single robotic arm equipped with a suction cup mechanism, there are however situations where bimanual grabbing of objects from the sides in more suitable. For instance, when picking up trays with open lid, lifting a case with too fragile cover to support the case' weight or when placing boxes on shelves with limited vertical space, just to name a few.

Thus, this paper focuses on bimanual pick-and-toss manipulation tasks and considers this problem in a depalletizing context. Unlike [6], the application envisioned in this paper extends the complexity of the pick-and-toss task as it requires the coordination of the two robotic arms from the grabbing of the object to its release. Although we previously addressed such a problem in [4] with a fixed target, this paper further extends the pick-and-toss task to the case where the target of the tossed object is moving[1] on a conveyor belt. This is illustrated in Figure 1. In the industry, such dynamic bimanual manipulation tasks are still largely done by humans for lack of similarly fast, precise and robust bimanual robotic systems. Thus, having bimanual robots with similar dynamic manipulation abilities, adaptivity and robustness would alleviate humans burden stemming from these repetitive and physically demanding tasks.

For throwing to be useful in industry, it must be accurate. The thrown object must land on the target within a given tolerance. This implies the control of the throwing parameters, particularly the release position and velocity, which must be appropriate for the thrown object to reach its target. Indeed, once an object is thrown, the thrower can no longer apply corrective action if the object does not follow the desired trajectory. Thus, tossing accurately an object onto a moving target using a robotic system is challenging. It requires solution of the following main subproblems:

michael.bombile@epfl.ch (M. Bombile); aude.billard@epfl.ch (A. Billard)
lasa.epfl.ch (A. Billard)
ORCID(s): 0000-0002-8031-9702 (M. Bombile); 0000-0002-7076-8010 (A. Billard)

[1]The moving target might represent here a free spot on a loaded and moving conveyor belt where the robot should put a depalletized object

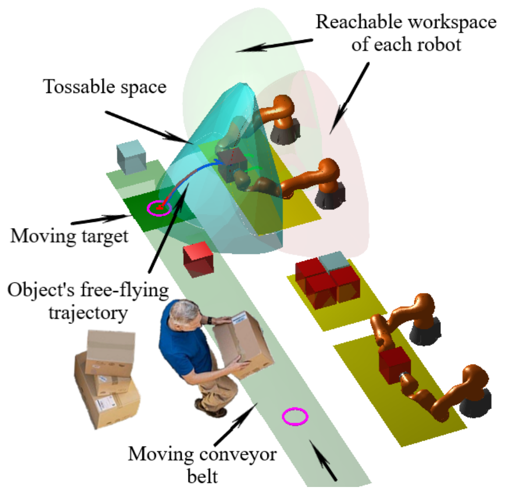

Reachable workspace of each robot
Tossable space
Moving target
Object's free-flying trajectory
Moving conveyor belt

Figure 1: Illustration of a hybrid human-robot depalletizing chain of dual-arm tossing tasks. In the upper depalletizing station, an object is tossed onto a moving target (in green) partly located in the convex hull of the tossable space (in cyan) of the dual-arm robot. In this scenario, the robot needs adaptation and fast re-planning abilities to face the perturbations induced by the human or conveyor belt on the target's speed and position.

1. Finding a feasible intercept position for the thrown object to meet the moving target.
2. Finding feasible throwing parameters (release position, release speed, and direction)
3. Generating the motion of the robot to grab the object and successfully reach the desired release state on time, such that the thrown object intercepts the target at the desired location.

Achieving this paper's objective requires solving an interception problem between the moving target and the thrown object. Finding a valid intercept point is straightforward; such a point lies along the path of the moving target within the throwable workspace of the robot (the robot's extended reachable workspace when throwing objects). As this point must be determined beforehand, on the target side, it requires estimation and prediction of the trajectory. On the robot side, it requires knowledge of the throwable workspace.

The problem of finding feasible throwing parameters is not trivial. It depends on the trajectory of the thrown object, which is subjected to gravity, and to nonlinear aerodynamic forces and phenomena that depend on its shape, speed, and environment (air density, pressure, etc.). The object's motion can be complex by combining translation and rotation around the object's center of mass, which may be geometrically eccentric, depending on the mass distribution.

Moreover, following the determination of the feasible throwing parameters corresponding to the desired intercept point, the motion of the robot must be generated and adapted such that the object is not thrown too early or too late to achieve the desired interception.

1.1. Related work

Precise throwing or tossing onto a moving target, whether performed by a single arm or a multi-arm system, is an interception problem, with challenges similar to robotic catching [8]-[14], batting, or hitting flying objects [15]-[21], or juggling [22]-[29]. Unlike these tasks, the interceptor is not the robot but rather the thrown object, whose final motion phase is governed by the projectile dynamics. Thus, in addition to robotic throwing literature, we briefly review a body of work on robotic interception tasks, with a focus on how the intercept position (configuration) was determined and how the robot motion was generated.

1.1.1. Robotic throwing

Robotic throwing is a dynamic manipulation task [30] that offers the possibility of positioning objects within or outside the physical workspace of a robot, and time and energy savings compared to non-dynamic manipulation methods [3, 4]. Since the dynamic manipulation work on robotic catching of a thrown object [31], several researchers have investigated the robotic catching and throwing problem. Apart from the dynamic aspects, research on robotic throwing is also motivated by potential applications in industry. For example, in [5], Frank et al. introduced the idea of using robotic throwing as a flexible alternative transportation method for certain types of products, and focused on a vision system to detect and track the thrown objects for catching purposes.

Robotic throwing has been demonstrated in the literature using different types of robotic systems that can be classified into three main categories: throwing with specialized devices such as 1-DoF or 2-DoF (degrees of freedom) launching mechanisms [30, 5, 32, 33, 34, 35, 36, 37], throwing by industrial robots [38, 39, 40, 41, 3] and throwing by humanoid robots [42, 43]. In the case of industrial robots, [38] used the KUKA KR-16 robot to demonstrate accurate throwing of a tennis ball to a target located approximately 2.5 m away. In [40] the TossingBot uses a UR5 robot to throw various objects with different shapes. More recently, an adaptive throwing solution was presented in [44] and validated on a Franka robot.

From the motion generation perspective, there are three main phases characterizing robotic throwing: an acceleration phase, a release phase, and a free-flight phase [45]. The first two phases are directly controlled by the robotic system, which must transport the object in a prehensile or non-prehensile manner to the desired release state (defined in terms of position and velocity) before releasing the object. This represents a trajectory generation problem with desired intermediate or transitory states, often addressed through motion planning [46, 39, 38], trajectory optimization [47, 41, 7] or methods based on optimal control [48], where the torques necessary to bring the robot to the desired state are directly computed. Unlike the planning-based methods which are less reactive and prone to spatial and temporal perturbations, this paper instead adopts an approach based on dynamical systems for their real-time adaptivity and robustness to perturbations.

The free-flying phase is governed by projectile dynamics. As no corrective action is possible after release of the object, the throwing task accuracy depends on how well the throwing parameters are determined, which depends on the modeling approach used for the free-flying object. In the literature, there are two main modeling approaches for throwing motion, with a third between them. The first approach relies on an analytical model of the object's free-flying dynamics, with or without nonlinear aerodynamic phenomena; the second approach is data-based. For instance, the authors of [38] used a simple ballistic motion to determine the throwing parameters, neglecting aerodynamic forces. Example of approaches that considered additionally the Newton drag forces can be found in [33]. In a batting application of free-flying objects, which can be seen as a simultaneous catching and throwing task, Jia et al., [21] considered the Magnus effect [49] and proposed a closed-form expression approximating the solution of the resulting nonlinear dynamics. Although physics-based analytical approaches can easily generalize to different conditions and objects, their prediction accuracy depends on knowledge of the object's physical properties and aerodynamic phenomena that occur. The latter are generally difficult to estimate, as discussed in [30].

The estimation of throwing parameters has also been addressed from a learning perspective. In an early work along this line, Aboaf et al. [50] directly exploited the object–target landing error in throwing tasks to learn the correct throwing parameters without modeling the underlying dynamics. Other example of learning-based approach can be found in [51, 52]. Although these approaches provide good accuracy, they do not generalize well to conditions other than those they were trained for. Approaches combining both physics and learning-based solutions have also been proposed to leverage their respective strengths. Such a hybrid approach was proposed for instance in [40]; to improve the success of throwing tasks, solution of a physics-based model was complemented with data-based components that learned the "residual physics" (unknown and unmodeled dynamics not captured by the physics-based model). The authors used deep neural networks to directly learn the residual physics in the control space, considering the synergy between grasping and tossing. This approach was successfully applied in tossing various small objects.

This paper also follows a hybrid modeling approach where an inverse throwing map is learned from a parameterized physics-based model of the free-flying dynamics. Unlike [40] and works that consider only the release speed as the main throwing parameter, our approach considers the full release state (position and velocity) and ensures their kinematic feasibility.

1.1.2. Robotic interception

Robotic interception consists of approaching a moving target to match its position and velocity in the shortest possible time [53]. This problem has been widely investigated from different perspectives in the literature. The cited works on catching, batting, and juggling addressed the interception problem. For slowly moving targets or targets with long-term predictable trajectories such as objects moving on a conveyor belt, the solution can be cast into the general prediction planning execution (PPE) framework [54, 55]. In this framework, the motion of the target is predicted. The robot's motion to an intercept or rendezvous location along the target trajectory is planned and executed. To address uncertainty and prediction errors, these steps can be repeated until interception, leading to an active PPE process (APPE). Interception of objects on conveyor belts was addressed with such a framework in [56] or [57] using a vision system. Allen et al. [58] demonstrated a more reactive vision-based hand–eye system with movement rates similar to human movement to track and grasp a moving model train. Over four decades, this topic has been addressed from different perspectives. Optimal control of tasks with robot dynamics and constraints was considered in [59], the optimal choice of interception point in [60], time-optimal considerations of tasks in [61], and the grasp reachability of the target in [62]. While these works considered the interception problem using a single robot, this paper focuses on the dual-arm system case, which requires enforcing the coordination of one robot with another when addressing the interception.

The robotic interception problem with dual-arm systems is not new. Previous work by our group [63] addressed the problem of robustly reaching moving objects with multi-arm systems. In that approach, generation of coordinated robot motion to perform reaching was based on dynamical systems and used a virtual object. A virtual object was also used in determination of intercept points by predicting object progress using a forward model. To ensure kinematic feasibility for the robots, the intercept points were determined along the intersection between the predicted object motion and the robot reachable space modeled with a Gaussian mixture model (GMM). Using this approach, catching a flying rod was demonstrated in [64] and reaching for various car parts was demonstrated in [65].

Although dual-arm coordinated interception tasks including grabbing and catching flying objects have been successfully performed, previous works including [66] and [65] were limited to fully controllable interceptors (the robots). This paper, however, addresses a problem where the interceptor is only partially controllable (the thrown object is only controllable up to the release).

1.1.3. Tossable workspace

Robotic throwing extends the robot workspace beyond its physical boundary. The tossable workspace is the set of all positions (within and outside the boundaries of the physical workspace) reachable by a given object if thrown by the robot. Unlike the normal robot reachable workspace, which depends only on robot joint configuration, the tossable workspace also depends on the kinematic and dynamic characteristics of the robot, and the properties (inertia, size, aerodynamic characteristics) and desired landing orientation of the object, making modeling difficult. Few studies have tackled the estimation problem of the tossable workspace of a robot. For instance, Gallant [41] proposed an approach for determining the maximum throwing reach of robots with kinematic and dynamic feasibility constraints. A trajectory optimization-based solution was proposed, with parameterized trajectories including cubic splines, polynomial functions, and Fourier series. The maximum throwing reach of 2-DoF, 3-DoF, and 5-DoF robots was determined. Although this approach can be a step toward estimating the tossable workspace, it was not explicitly determined. More recently, Asgari and Nikoobin [48] proposed an indirect solution-based optimal control approach to estimate the maximum set of points to which robotic manipulators can throw an object. They called such set the "maximum throw-able" workspace. They modeled the throwing trajectory using the simple ballistic motion and applied it as a moving boundary conditions to optimize the release speed and angle. While this approach could successfully estimate the feasible throw-able workspace of 2 DoF planar and spatial robots, as highlighted by the author themselves, it comes with high computation burden which makes it difficult for online usage.

This paper considers more complex robots (dual-arm of two 7 Dofs robots) and proposes to estimate the throw-able space based on the kinematic feasibility of release states associated with landing points spanning the robot workspace and beyond. Unlike [48], this paper goes beyond the mere estimation of the throwable points by proposing a probabilistic model of their distribution and uses it to predict intercept (landing) positions that yield a high probability of task success.

1.2. Contribution

The above review of related work shows that an integrated robotic framework that allows the interception of a moving target by a thrown object using a dual-arm robotic system is missing. Thus, this paper extends our previous work [4] and offers, to our knowledge, the first example of coordinated control of a bimanual arm platform to toss objects so that they land on a moving target. Furthermore, we show that the approach is fast to compute and hence can re-plan a feasible trajectory on the fly in the face of disturbances, such as a change in the trajectory of the target, or displacement of one of the arms.

To achieve this, we propose to leverage on closed-form representation of part of the planning problem to simplify computation at run time and hence ensure online replanning. We also propose appropriate coordination across the two arms for them to toss in synchrony with one another and with the moving target, so as to preserve this synchronization in the face of delay in motion of either the arm or target.

The paper's contributions can be summarized as follows:

1. we propose a method to compute kinematically feasible throwing parameters (release position and velocity) for a dual-arm robotic system using a learned inverse throwing map combined with a kinematics-based optimization framework.

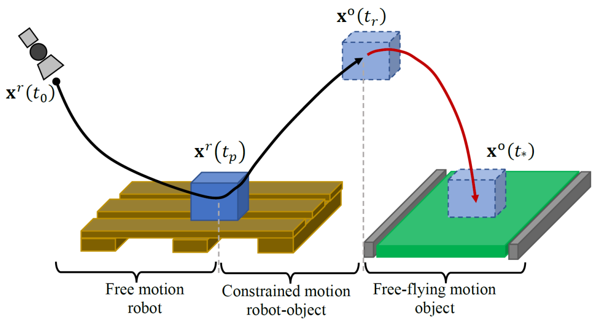


Figure 2: Illustration of dual-arm robotic depalletizing task with fast grabbing and tossing of an object. The overall motion can be split into three main phases; the system motion is determined by the free robot motion, the constrained robot–object motion, and the object free-flying motion before interception of the target at landing.

2. we propose a method to derive a closed-form model (probabilistic) of the tossable workspace of a robot (set of all positions reachable by an object if tossed by the robot). This model enables the prediction of intercept locations with a higher probability of success.
3. we offer a systematic assessment of this approach in a realistic scenario with a bimanual robot platform tossing boxes on a conveyer belt. We demonstrate its effectiveness in reducing task completion time and energy expenditure when compared to a traditional pick-and-place approach.

2. Modeling

Consider a dual-arm robot tasked to quickly grab an object from a pallet and toss it onto a target moving on a conveyor belt, as shown in Figure 2. The system motion throughout the task is characterized by three main phases: a free-motion phase where only the robot moves (from initial time (t_0) to pick up time (t_p)), a constrained robot–object motion phase (from (t_p) to the release time (t_r)), and a free-flying motion phase for the object (from (t_r) to the landing time (t_l)).

2.1. Robot and object dynamic model

The dynamics of the dual-arm robot interacting with its environment can be written as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{J}_e^\top(\mathbf{q})F_e = \boldsymbol{\tau} \tag{1}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_D \times n_D}$ and $\mathbf{b}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n_D}$ are block-diagonal matrix inertia and block-diagonal matrix of centrifugal, Coriolis, and gravity forces of the dual-arm robot, respectively. The vectors $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^{n_D}$ are the vector of joint position, velocity and acceleration of the dual-arm robot, respectively. n_D denotes the overall number of DoF, such that $n_D = n_L + n_R$, where n_L and n_R are the DoF of the left and right robotic arm, respectively. $\mathbf{J}_e(\mathbf{q}) \in \mathbb{R}^{12\times n_D}$ and $F_e \in \mathbb{R}^{12}$ are block-diagonal Jacobian matrix of the interacting end-effector(s) and the vector of associated wrenches, respectively. $\boldsymbol{\tau} \in \mathbb{R}^{n_D}$ is the vector of joint torques.

The dynamics of the object assumed to be rigid, with mass m_{o} and inertia $\mathcal{I}_{\mathrm{o}}$ can be written as

$$\begin{bmatrix} m_{\mathrm{o}}\mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathcal{I}_{\mathrm{o}} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{x}}^{\mathrm{o}} \\ \dot{\boldsymbol{\omega}}^{\mathrm{o}} \end{bmatrix} + \begin{bmatrix} -m_{\mathrm{o}}\mathbf{g} \\ \boldsymbol{\omega}^{\mathrm{o}} \times \mathcal{I}_{\mathrm{o}}\boldsymbol{\omega}^{\mathrm{o}} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_o \\ \boldsymbol{\tau}_o \end{bmatrix} \tag{2}$$

where $\ddot{\mathbf{x}}^{\mathrm{o}} \in \mathbb{R}^3$ and $\dot{\boldsymbol{\omega}}^{\mathrm{o}} \in \mathbb{R}^3$ are the object's linear and angular acceleration, respectively. $\mathbf{g} \in \mathbb{R}^3$ represents the gravity vector; $\mathbf{f}_{\mathrm{o}} \in \mathbb{R}^3$ and $\boldsymbol{\tau}_o \in \mathbb{R}^3$ denote the effective force and torque, respectively, acting at the object's frame.

When the object is firmly grasped, the coupling between the robot and the object can be written as

$$\begin{bmatrix} \mathbf{f}_o \\ \boldsymbol{\tau}_o \end{bmatrix} = \mathbf{G}_{\mathrm{o}} F_{\mathrm{e}} \tag{3}$$

where $\mathbf{G}_{\mathrm{o}} \in \mathbb{R}^{6\times 12}$ denotes the bi-manual grasp matrix [67].

The rigid grasp imposes kinematic constraints such that

$$\underbrace{(\mathbf{G}_{\mathrm{o}}^{+})^{\top}\mathbf{J}_{\mathrm{e}}(\mathbf{q})}_{\mathbf{J}_{\mathrm{o}}(\mathbf{q})}\dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{x}}^{\mathrm{o}} \\ \boldsymbol{\omega}^{\mathrm{o}} \end{bmatrix} \text{ and } \underbrace{(\mathbf{I} - \mathbf{G}_{\mathrm{o}}^{+}\mathbf{G}_{\mathrm{o}})^{\top}\mathbf{J}_{\mathrm{e}}(\mathbf{q})}_{\mathbf{J}_{\mathrm{c}}(\mathbf{q})} = \mathbf{0} \tag{4}$$

where $\mathbf{J}_{\mathrm{o}}(\mathbf{q}) \in \mathbb{R}^{6\times n^D}$ and $\mathbf{J}_{\mathrm{c}}(\mathbf{q}) \in \mathbb{R}^{12\times n^D}$ denote the object's motion and contact constraint Jacobian, respectively. $\mathbf{G}_{\mathrm{o}}^{+}$ denotes the generalized inverse of $\mathbf{G}_{\mathrm{o}}$.

Therefore, during the constrained motion, all states of the dual-arm are restricted within the feasible set, $\mathcal{F}$, defined by

$$\mathcal{F} = \left\{ s = \begin{bmatrix} \mathbf{x}(\mathbf{q}) \\ \dot{\mathbf{x}}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix} \middle| \begin{array}{c} \mathbf{q}_{\min} \le \mathbf{q} \le \mathbf{q}_{\max} \\ \mathbf{J}_{\mathrm{o}}(\mathbf{q})\dot{\mathbf{q}} = [\dot{\mathbf{x}}^{\mathrm{o}\top}\ \boldsymbol{\omega}^{\mathrm{o}\top}]^{\top} \\ \mathbf{J}_{\mathrm{c}}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0} \\ |\dot{\mathbf{q}}| \le \dot{\mathbf{q}}_{\max} \end{array} \right\} \tag{5}$$

where $s = (\mathbf{x}(\mathbf{q}), \dot{\mathbf{x}}(\mathbf{q}))$ represents the task-space state of the dual-arm robot; $\mathbf{x}(\mathbf{q})$ and $\dot{\mathbf{x}}(\mathbf{q}, \dot{\mathbf{q}})$ are the pose and twist velocity vectors, respectively, obtained through forward kinematics. $\mathbf{q}_{\min}$, $\mathbf{q}_{\max}$ and $\dot{\mathbf{q}}_{\max}$ are the joint position and velocity limits, respectively.

2.2. Object free-flying motion

Assuming that the center of mass (CoM) of the object corresponds to its geometric center, and when thrown, the object does not rotate and is subjected only to gravity and aerodynamic drag forces, its dynamics reduces to the first raw of (2). Thus, its motion will be governed by

$$\ddot{\mathbf{x}}^{\mathrm{o}} = \mathbf{g} + \frac{\mathbf{f}_D}{m_{\mathrm{o}}} \tag{6}$$

where $\mathbf{f}_D = \mathbf{f}_o$ and represents the aerodynamic drag force (the lift force is neglected) expressed as

$$\mathbf{f}_D = -\frac{\rho_{air} c_D A_o}{2}\dot{\mathbf{x}}\,\|\dot{\mathbf{x}}\| \tag{7}$$

where ρ_{air} is the air density, c_D denotes the drag coefficient and A_o represents the cross-sectional area of the object in the motion direction. Eq. (6) can be simplified and written as a function of the object's states ($\mathbf{x}^{\mathrm{o}}$ and $\dot{\mathbf{x}}^{\mathrm{o}}$) as

$$\frac{d}{dt}\begin{pmatrix} \mathbf{x}^{\mathrm{o}} \\ \dot{\mathbf{x}}^{\mathrm{o}} \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{x}}^{\mathrm{o}} \\ -\eta\dot{\mathbf{x}}^{\mathrm{o}}\,\|\dot{\mathbf{x}}^{\mathrm{o}}\| + \mathbf{g} \end{pmatrix} \tag{8}$$

where $\eta \triangleq \frac{\rho_{air} c_D A_o}{2m_o}$. With these assumptions, Eq. (8) allows prediction of the object trajectory from a given initial state that denoted as $s_0^{\mathrm{o}} \triangleq (\mathbf{x}_0^{\mathrm{o}}, \dot{\mathbf{x}}_0^{\mathrm{o}})$ to its landing state denoted as $s_l^{\mathrm{o}} \triangleq (\mathbf{x}_l^{\mathrm{o}}, \dot{\mathbf{x}}_l^{\mathrm{o}})$.

3. Problem Statement and Proposed Approach

3.1. Problem statement

Assuming that a low-level robot controller of the form $\boldsymbol{\tau} = \boldsymbol{\tau}(\dot{\mathbf{x}}^{\mathrm{d}}, F_{\mathrm{e}}^{\mathrm{d}})$ is available and it generates torque commands based on desired manipulation task expressed in terms of desired end-effector motion[2] $\dot{\mathbf{x}}^{\mathrm{d}}$ and forces[3] $F_{\mathrm{e}}^{\mathrm{d}}$. Consider now our task consisting of grabbing and tossing an object onto a moving target with a dual-arm robot modeled by (1), and with the dynamics of the object expressed in Eq. (2) and (8). There are three problems to solve:

P1: how to determine a reachable intercept location (denoted by $\mathbf{x}_I \in \mathbb{R}^3$) between the tossed object and the moving target;

P2: how to determine the object's release position and velocity $s_r^{o*} \triangleq (\mathbf{x}_r^{\mathrm{o}*}, \dot{\mathbf{x}}_r^{\mathrm{o}*})$ and the associated release state of the dual-arm system $s_r^* \triangleq (\mathbf{x}_r^*, \dot{\mathbf{x}}_r^*)$ that will result in the tossed object landing at the intercept location $\mathbf{x}_I$;

P3: how to generate the desired robot motion $\dot{\mathbf{x}}_{\mathrm{d}}$ and force $F_{\mathrm{e}}^{\mathrm{d}}$ to grab the object and successfully reach the desired release state s_r^* on time such that the thrown object intercepts the target at $\mathbf{x}_I$.

Addressing (**P1**) requires a model of the dual-arm robot tossable workspace to predict the likelihood of reaching any potential intercept location.

Solving (**P2**) requires finding a robot state that satisfies both kinematic feasibility constraints and grasping constraints on the object. Thus, according to (5), a valid release state for the robot should satisfy $s_r^* \in \mathcal{F}$.

Solving (**P3**) is nothing but developing a robust coordinated motion and force control strategy for the dual-arm system to reach, grab, and toss the object to ensure successful interception of the moving target by the object in the presence of spatial and temporal perturbations. Such perturbations could consist of a live modification of the target motion (speeding up or slowing down) or a displacement of the target location on the conveyor belt.

[2]The motion task can also be expressed in terms of $\ddot{\mathbf{x}}_{\mathrm{d}}$

[3]On the object side, $F_{\mathrm{e}}^{\mathrm{d}}$ is distributed between effective ($\mathbf{f}_o$, τ_o) and contact wrench

In addressing Problems **P1-P3**, we make the following assumptions regarding the system:

- the dual-arm robot establishes rigid contact with the grasped object until release
- the object does not rotate or tumble during its free-flying motion phase
- the post-landing impact of the object is negligible

3.2. Overview of proposed approach

Our proposed approach to achieve dual-arm grabbing and precise tossing of an object onto a moving target consists of three parts:

S1: devising a kinematics-based optimization algorithm that computes constraint-satisfying throwing states while minimizing the throwing speed (see Section 4);

S2: learning a model of the tossable workspace S_T from a distribution of reachable points determined using the algorithm in **S1** (see Section 5);

S3: generating the desired motion $\dot{\mathbf{x}}_\mathrm{d}$ and force F_e^d using our previously developed dynamical system based on a dual-arm controller for fast grabbing and tossing of objects [4] with an adaptation factor $\beta(\mathbf{x})$ to compensate for changes in the target motion, and prediction and tracking inaccuracies in the robot motion (see Section 6)

The proposed approach is summarized in the control architecture illustrated in Figure 3. Target position measurements are used to estimate target motion and predict target trajectory (determined by the conveyor belt). The predicted trajectory of the target and the learned tossable workspace of the dual-arm robot and its predicted motion are used to determine an interception point updated over time. The computed interception position is the desired landing position of the thrown object, and is used in conjunction with the learned projectile throwing map to determine feasible release state of the object. The obtained feasible release state is sent as a reference for the DS-based controller. The motion generated by the DS-based controller is adapted to compensate for state prediction and control error during execution of the desired motion.

4. Estimation of Feasible Throwing States

In this section, we propose an approach for computing kinematically feasible throwing release states (position and velocity) for a dual-arm robotic system from desired landing positions. Our approach focuses on solutions (essentially local optimum) that primarily satisfy the feasibility constraints and then minimize the throwing speed rather than finding the best solutions. Thus, with our approach encoding optimal partial release states (velocities) in a learning model, obtaining a corresponding feasible full release state will be considered a success. Our approach's emphasis on feasibility allows rapid adaption by seeking a new feasible solution in case of perturbations that requires fast replanning.

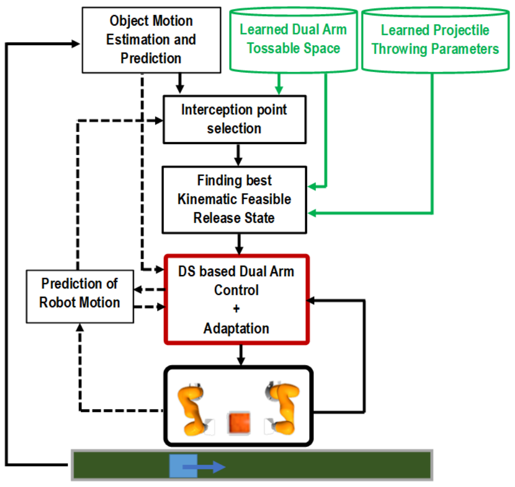

Object Motion Estimation and Prediction
Learned Dual Arm Tossable Space
Learned Projectile Throwing Parameters
Interception point selection
Finding best Kinematic Feasible Release State
Prediction of Robot Motion
DS based Dual Arm Control + Adaptation

Figure 3: Block diagram showing processes and information flow in proposed approach for dual-arm tossing of an object onto a moving conveyor belt. The blocks "Learned Dual-Arm Tossable Space" and "Learned Projectile Throwing Parameters" are trained offline. The black continuous and dashed lines represent information flow updated at 200 Hz and 10 Hz, respectively.

4.1. Learning an inverse throwing map

Unlike the forward dynamics in Eq. (8), we are now concerned with the inverse throwing problem, which consists of finding an initial throwing state $(\mathbf{x}_0^\mathrm{o}, \dot{\mathbf{x}}_0^\mathrm{o})$ that yields a desired landing position[4] $\mathbf{x}_l^\mathrm{o}$. As stated before, this problem is not trivial and admits multiple solutions.

4.1.1. *Proposed concept*

To address this challenge, we propose a two-step approach.

Step 1: We determine a function $\dot{\mathbf{x}}^\mathrm{o} = \mathbf{v}_r(\bar{\mathbf{x}}^\mathrm{o}) \in \mathbb{R}^3$ that computes the release velocity from the relative position $\bar{\mathbf{x}}^\mathrm{o} \in \mathbb{R}^3$ between the desired landing position and the release position of the object ($\bar{\mathbf{x}}^\mathrm{o} = \mathbf{x}_l^\mathrm{o} - \mathbf{x}_0^\mathrm{o}$). We resolve the inherent redundancy problem by adopting a throwing strategy that seeks minimal throwing velocity. Such a strategy is also beneficial for the robot as it requires less kinetic energy for throwing. Other redundancy resolution strategies such as minimum landing velocity (vertical or horizontal components) can also be adopted.

However, except for linear projectile motion, $\dot{\mathbf{x}}^\mathrm{o} = \mathbf{v}_r(\bar{\mathbf{x}})$ does not have a closed-form solution mainly due to the nonlinear aerodynamic forces. Thus, we propose a closed-form expression of $\mathbf{v}_r(\bar{\mathbf{x}}^\mathrm{o})$ learned from data. To that end, we use Gaussian mixture regression (GMR) [68] for its ability to handle multi-dimensional input and output data. We define throwing situations for a given object as $\mathcal{C}_\mathrm{v} = \{\bar{\mathbf{x}}^\mathrm{o}, \dot{\mathbf{x}}^\mathrm{o}\}$ and model a dataset of N such throwing situations $(\{\mathcal{C}_\mathrm{v}^i\}_{i=1\ldots N})$ using a GMM. The model is assumed to have K_v Gaussian functions, can be represented by its parameters as $\Omega_{\mathrm{v}_\mathrm{r}} \equiv$

[4]We leave the landing velocity $\dot{\mathbf{x}}_l^\mathrm{o}$ as a free variable as we are more concerned with the object's landing accuracy on the target than the landing speed.

$\{\pi^k, \mu^k, \Sigma^k\}_{k=1\dots K_v}$, where π^k, $\mu^k$and Σ^k are the prior, the mean and the covariance of the k^{th} Gaussian distribution, respectively. At any query time, given a desired relative release position $\bar{\mathbf{x}}^{o*}$, the desired release velocity $\dot{\mathbf{x}}^{o*}$ is obtained by computing the expectation over the conditional distribution $p(\dot{\mathbf{x}}^{o*}|\bar{\mathbf{x}}^{o*}, \Omega_{v_r})$. The resulting function can be written as

$$\mathbf{v}_r(\bar{\mathbf{x}}^o) \approx \sum_{k=1}^{K} h^k(\bar{\mathbf{x}}^o)\tilde{\mu}^k_{\dot{\mathbf{x}}^o|\bar{\mathbf{x}}^o}(\bar{\mathbf{x}}^o) \tag{9}$$

with $\tilde{\mu}^k_{\dot{\mathbf{x}}^o|\bar{\mathbf{x}}^o} = \mu^k_{\dot{\mathbf{x}}^o} + \Sigma^k_{\dot{\mathbf{x}}^o\bar{\mathbf{x}}^o}\left(\Sigma^k_{\bar{\mathbf{x}}^o\bar{\mathbf{x}}^o}\right)^{-1}(\bar{\mathbf{x}}^o - \mu^k_{\bar{\mathbf{x}}^o})$, where $\mu^k_{\bar{\mathbf{x}}^o}$ and $\mu^k_{\dot{\mathbf{x}}^o}$ are element vectors of the mean μ^k of the k^{th} Gaussian function associated with input data $\bar{\mathbf{x}}^o$ and output data $\dot{\mathbf{x}}^o$, respectively. Similarly, $\Sigma^k_{\bar{\mathbf{x}}^o\bar{\mathbf{x}}^o}$ and $\Sigma^k_{\dot{\mathbf{x}}^o\bar{\mathbf{x}}^o}$ are element matrices extracted from the covariance matrix Σ^k. In Eq. (9), $h^k(\bar{\mathbf{x}}^o)$ weights the relative importance of the k^{th} Gaussian function in the regression of $\mathbf{v}_r(\bar{\mathbf{x}}^o)$.

Step 2: Using the obtained expression of $\mathbf{v}_r(\bar{\mathbf{x}}^o)$, we seek through an optimization process a value of the release position $\mathbf{x}^o_0$ that is reachable by the robot and yields a minimum feasible $\dot{\mathbf{x}}^o$. To that end, we must derive the Jacobian between variations of $\mathbf{v}_r$ and $\bar{\mathbf{x}}^o$. With the smoothness of Gaussian functions, this Jacobian can be derived in closed form. As $\bar{\mathbf{x}}^o$ is a function of the robot's forward kinematics, such that

$$\bar{\mathbf{x}}^o = \mathbf{x}^o_l - \mathbf{x}^o(\mathbf{q}) \tag{10}$$

the Jacobian between variations of $\mathbf{v}_r$ and $\mathbf{q}$ can obtained as

$$d\mathbf{v}_r = \mathbf{J}_{\mathbf{v}_r}(\mathbf{q})d\mathbf{q} = \underbrace{\frac{\partial \mathbf{v}_r}{\partial \bar{\mathbf{x}}^o}}_{\mathbf{J}_{\mathbf{v}_r}(\bar{\mathbf{x}}^o)}\underbrace{\frac{\partial \bar{\mathbf{x}}^o}{\partial \mathbf{x}^o}}_{-\mathbf{I}_{3\times3}}\underbrace{\frac{\partial \mathbf{x}^o}{\partial \mathbf{q}}}_{\mathbf{J}_{ov}(\mathbf{q})} d\mathbf{q} \tag{11}$$

where the expression of $\mathbf{J}_{\mathbf{v}_r}(\bar{\mathbf{x}}^o)$ is presented in Appendix (9.1). The $\mathbf{J}_{ov}(\mathbf{q}) \in \mathbb{R}^{3\times n^D}$ term is simply the translational block component of the direction of the object's motion Jacobian $\mathbf{J}_o(\mathbf{q})$ in (4).

4.1.2. Data generation and model training

The inverse throwing map in Eq. (9) encodes solutions to the estimation of throwing release velocity when the release position is given and a minimum release speed strategy is adopted. These are parameterized by η (defined in Eq.(8)).

To train the model of the inverse throwing map, we need examples of throwing situations (set of release positions and release velocities). To that end, we proceed as follows.

First, we artificially generate 10^5 3D relative release positions within a spherical sector defined by a throwing reach ranging from $[0, 2.5]m$[5], and throwing directions within a cone angle of $\left[\pm\frac{5\pi}{12}\right]$ *rad* around the x axis as shown in Figure 6-(left). The data are generated for 20 values of the aerodynamic parameter η to capture the characteristics of various objects (eg. change of mass and/or shape).

Second, to generate the release velocity $\dot{\bar{\mathbf{x}}}$ corresponding to the generated set of $\bar{\mathbf{x}}$, we solve, for each point a two-point boundary value problem (TPBVP) [69] defined by the object's free-flying dynamics (8) with the origin $(0, 0, 0)$ as initial position and $\bar{\mathbf{x}}$ the final position. We solve this problem using a shooting algorithm.

We assume that the projectile motion lies in a plane to reduce the dimensionality of the problem from 3D to 2D. We use a Cartesian-to-cylindrical coordinate transformation ((x, y, z) to (r, φ, z))to extract the equivalent planar coordinates (r, z) lying in the vertical plane containing the origin and the landing position. r and z represent the radial distance and the height in the coordinates system, respectively. This transformation yields a planar release velocity parameterized by the release angle θ_0 and release speed v_0. The geometric representation of the 3D to 2D coordinates transformation is illustrated in Figure 4.

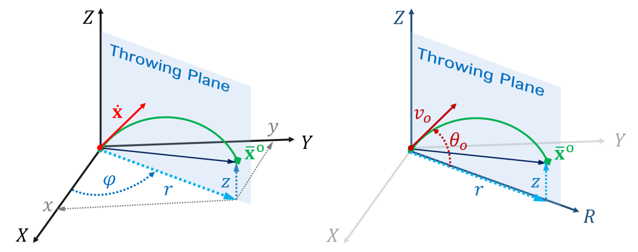


Figure 4: Geometric representation throwing variables. -(left): throwing problem reduction from 3D to 2D using Cartesian to Cylindrical coordinates. -(right): resulting planar throwing parameters

To determine the release angle θ^*_0 and speed v^*_0, we initialize our shooting algorithm with the analytical solution of the linear ballistic motion with minimal release speed.

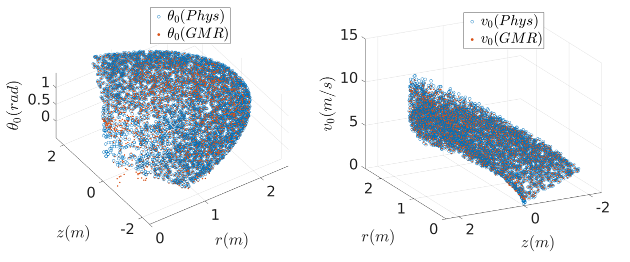
$$\begin{aligned} \theta_0 &= arctan\left(\frac{z}{r} + \sqrt{\left(\frac{z}{r}\right)^2 + 1}\right) \\ v_0 &= \sqrt{\frac{g.r^2.(1 + tan^2(\theta_0))}{2.(r.tan(\theta_0) - z)}} \end{aligned} \tag{12}$$

Our shooting algorithm has two stages. In the first stage, using v_0 and starting at θ_0, we search for the angle θ^*_0 that yields the maximal reach with the velocity v_0. Once θ^*_0 is obtained, the second stage searches for the release speed that leads to the landing position (r_l, z_l), with the speed iteratively updated as

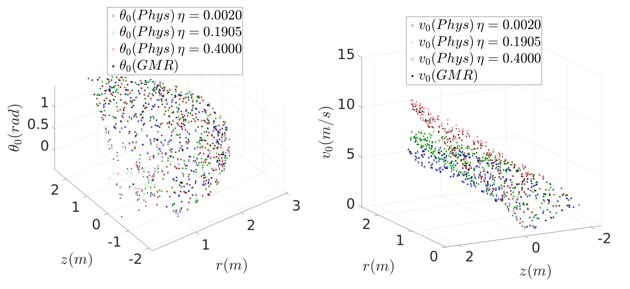
$$v_0(i+1) = v_0(i) + \kappa(r^d_l - r_l) \tag{13}$$

where r^d_l is the desired landing position, while κ denotes the update rate of the algorithm. The algorithm stops when the predicted release position reach the desired landing position within a small tolerance distance. Figure 5 illustrates the obtained planar throwing parameters (Figure 5(a)), with samples of three different values of η (Figure 5-(b)).

[5]This restriction on the spatial span of data considers the robot limits; for instance, a maximum reach of 2.5 m requires a minimum release speed of 5 m/s, which is beyond the robot capability.



(a)



(b)

Figure 5: Illustration of optimal planar throwing parameters: (a) 5000 samples of throwing parameters; (b) throwing parameters corresponding to three different values of non-linear drag force η

Once the release speed v_0^* and angle θ_0^* were obtained, we reconstructed the 3D equivalent $\dot{\mathbf{x}}^o$, which together with $\bar{\mathbf{x}}^o$ constitute the dataset required for the training. The resulting dataset with position and velocity data points is shown in Figure 6.

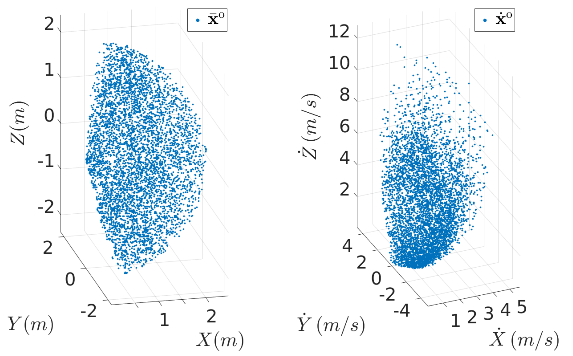


Figure 6: Example of dataset used to learn the inverse throwing map. The left figure shows the desired landing position relative to the frame origin; the right figure shows the corresponding 3D release velocities obtained when using a minimal throwing release speed strategy.

With the obtained dataset, we trained our GMM model with two-thirds of the data using the expectation-maximization (EM) algorithm initialized with k-means. Our model uses the Bayesian information criterion (BIC) to choose the number of Gaussian functions, 25 in this case. With the obtained GMM, we can predict with GMR the throwing release velocity based on the desired relative position $\bar{\mathbf{x}}^o$.

We validated our GMR model using the remaining one-third of the data. We obtained an RMSE of 0.13 m/s for the throwing velocities. Figure 7 shows error histograms between velocities generated by the learned GMR model and ground truth velocities (physics) in the x, y, and z dimensions.

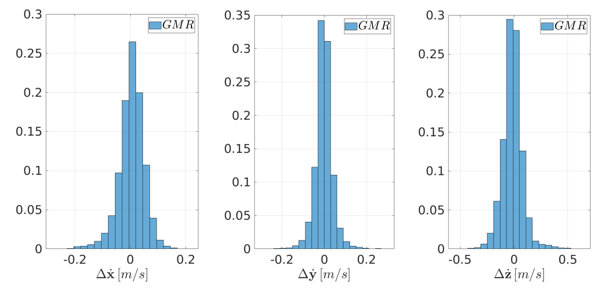


Figure 7: Validation errors between velocities generated by the learned GMR model and ground truth velocities (physics): velocity errors in x (left), in y (middle), and in z (right)

The landing position errors corresponding to the testing data set are illustrated in Figure 8 along with the ground truth landing errors (physics-based model). It is observed that the physics-based parameters are accurately predicted by the GMR; 97% of the predicted landing positions fall within 0.1m (4% of maximum reach of 2.5m) of the target and 80% fall within 0.05m. Figure 9 provides examples of throwing trajectories for the learned model and the ground truth.

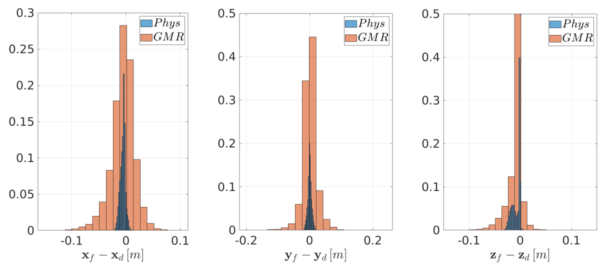


Figure 8: Validation errors for desired landing positions obtained using learned throwing parameters (GMR model) and ground truth parameters: errors in x (left), in y (middle), and in z (right)

4.2. Optimal feasible release state: Concept

To compute the optimal feasible release state, our main idea is to define a throwing task-related cost function denoted as $l(\mathbf{q}, \dot{\mathbf{q}})$, and then compute its optimizer in terms of joint position and velocity $(\mathbf{q}^*, \dot{\mathbf{q}}^*)$ that satisfy the task constraints. The task-space release state is obtained through forward kinematics as $\mathbf{s}_r^* = (\mathbf{x}_r(\mathbf{q}^*),\ \dot{\mathbf{x}}_r(\mathbf{q}^*))$ with

$$\mathbf{q}^*, \dot{\mathbf{q}}^* = \underset{\mathbf{q}, \dot{\mathbf{q}}}{\text{argmin}}\ l(\mathbf{q}, \dot{\mathbf{q}}) \tag{14}$$

$$s.t. \quad \mathbf{J}_c(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0} \tag{15}$$

$$\mathbf{q}_{\min} \leq \mathbf{q} \leq \mathbf{q}_{\max} \tag{16}$$

$$|\dot{\mathbf{q}}| \leq \dot{\mathbf{q}}_{\max} \tag{17}$$

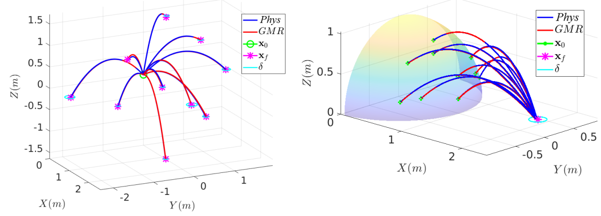


Figure 9: Example of ten trajectories and landing positions for throwing tasks using learned throwing parameters and those obtained using physics: left: different desired landing positions from the same release position at the origin; right: different release positions for the same desired landing position

where Eq. (15), (16) and (17) enforce the feasibility constraints, for the grasp, joint position and joint velocity, respectively.

To define the overall objective function $l(\mathbf{q}, \dot{\mathbf{q}})$, we use the previously learned throwing map $\mathbf{v}_r(\mathbf{x}_l^d, \mathbf{x}_r(\mathbf{q})) \in \mathbb{R}^3$ that computes the throwing velocity from the relative position between the release position $\mathbf{x}_r(\mathbf{q})$ and the desired landing position $\mathbf{x}_l^d$. We aim at achieving throwing tasks with minimal release speed as possible.

Based on the definition of the feasible set (5), we define the cost function as

$$l(\mathbf{q}, \dot{\mathbf{q}}) = \left\|\mathbf{v}_r(\mathbf{q})\right\|^2_{w_p} + \left\|\mathbf{J}_o(\mathbf{q})\dot{\mathbf{q}} - \mathbf{v}_r(\mathbf{q})\right\|^2_{\mathrm{w}_v} + \left\|\mu\theta(\mathbf{R}_O(\mathbf{q}_k), \mathbf{R}_O^d)\right\|^2_{w_O} \tag{18}$$

Thus, minimizing the first sub-objective in Eq. (18) amounts to seeking joint configurations that yield minimal release velocity; minimizing the second sub-objective produces (for configuration $\mathbf{q}$) joint velocities $\dot{\mathbf{q}}$, whose associated task-space velocity approaches the desired throwing velocity ($\mathbf{J}_{\mathrm{vo}}(\mathbf{q})\dot{\mathbf{q}} = \dot{\mathbf{x}}_r(\mathbf{q}) \to \mathbf{v}_r(\mathbf{q})$). The third sub-objective helps resolve the redundancy by specifying desired end-effector orientation during the throwing task.

4.3. Optimal feasible release state: Solution

Optimization of Eq. (14) with the cost function in Eq. (18) is nonlinear in terms of joint configuration and does not have a closed-form solution. Thus, we propose to solve it iteratively using sequential quadratic programming [70] with the joint acceleration as the decision variable. Such an approach allows simultaneous updating of both the position and velocity subjected to their respective constraints as follows

$$\begin{aligned} \mathbf{q}^* &\leftarrow \mathbf{q}_{k+1} = \mathbf{q}_k + \delta t\dot{\mathbf{q}}_k + (\delta t)^2\ddot{\mathbf{q}}^* \\ \dot{\mathbf{q}}^* &\leftarrow \dot{\mathbf{q}}_{k+1} = \dot{\mathbf{q}}_k + \delta t\ddot{\mathbf{q}}^* \\ &\quad \text{until convergence} \end{aligned} \tag{19}$$

where δt is the time step and represents the step length of the algorithm; $\ddot{\mathbf{q}}^*$ is the optimal acceleration at the k^{th} iteration and provides the stepping direction.

To compute the optimal acceleration $\ddot{\mathbf{q}}^*$, we reformulate the problem as a bilevel optimization problem [71]; at the top level, we compute the feasible release velocity $\dot{\mathbf{x}}_r^f$ for the configuration $\mathbf{q}_k$ as

$$\dot{\mathbf{x}}_r^f = \mathbf{J}_o(\mathbf{q}_k)\dot{\mathbf{q}}_k^f \quad \text{with} \tag{20}$$

$$\dot{\mathbf{q}}_k^f = \underset{\dot{\mathbf{q}}}{\operatorname{argmin}} \ \left\|\mathbf{J}_o(\mathbf{q}_k)\dot{\mathbf{q}} - \mathbf{v}_r(\mathbf{q}_k)\right\|^2 \quad s.t. \quad |\dot{\mathbf{q}}| \leq \dot{\mathbf{q}}_{\max} \tag{21}$$

The velocity $\dot{\mathbf{x}}_r^f$ represents the closest throwing velocity to $\mathbf{v}_r(\mathbf{q}_k)$ that the robot can achieve from configuration $\mathbf{q}_k$, with $\dot{\mathbf{q}}_k^f$ as its corresponding joint-space value.

At the bottom level of the optimization, we solve for the acceleration $\ddot{\mathbf{q}}_k^*$ that minimizes: the difference between $\dot{\mathbf{x}}_r^f$ and $\mathbf{v}_r(\mathbf{q}_k)$, the module of $\mathbf{v}_r(\mathbf{q}_k)$ and the difference between the current and desired end-effector orientation ($\mathbf{R}_O(\mathbf{q})$ and $\mathbf{R}_O^d$). Thus, following Eq. (14), the bottom level problem is reformulated as

$$\ddot{\mathbf{q}}_k^* = \underset{\ddot{\mathbf{q}}}{\operatorname{argmin}} \ \sum_i l_i'(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k) \quad \text{with} \quad i = \{\mathrm{p}, O\} \tag{22}$$

$$s.t. \quad \mathbf{J}_c(\mathbf{q}_k)\ddot{\mathbf{q}}_k + \dot{\mathbf{J}}_c(\mathbf{q}_k)\dot{\mathbf{q}}_k = \mathbf{0} \tag{23}$$

$$\mathbf{q}_{\min} \leq \mathbf{q}_k + \delta t\dot{\mathbf{q}}_k + (\delta t)^2\ddot{\mathbf{q}}_k \leq \mathbf{q}_{\max} \tag{24}$$

$$\dot{\mathbf{q}}_{\min} \leq \dot{\mathbf{q}}_k + \delta t\ddot{\mathbf{q}}_k \leq \dot{\mathbf{q}}_{\max} \tag{25}$$

$$\mathbf{q}_{\min} \leq \mathbf{q}_k + \delta t\dot{\mathbf{q}}_k + (\delta t)^2\ddot{\mathbf{q}}_k \pm \Delta\mathbf{q} \leq \mathbf{q}_{\max} \tag{26}$$

where Eqs. (23)-(25) enforce the feasibility constraints. The constraint (26) ensures that the joint has motion range $\Delta\mathbf{q}$ to accelerate from zero to $\dot{\mathbf{q}}_k^f$. $\Delta\mathbf{q}$ for each joint is expressed as

$$\Delta q_j = \frac{(\dot{q}_j^f)^2}{2\ddot{q}_{j,\max}} \tag{27}$$

The term $l_i'(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k)$ represents the redefined sub-objectives at the acceleration level. Exploiting the kinematics of the robots, we propose to formulate $l_i'(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k)$ as in an acceleration-based inverse kinematics such that

$$l_i'(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_k) \triangleq ||\mathbf{J}_i(\mathbf{q}_k)\ddot{\mathbf{q}}_k - (\ddot{\mathbf{x}}_i^d - \dot{\mathbf{J}}_i(\mathbf{q}_k)\dot{\mathbf{q}}_k)||^2_{\mathrm{w}_i} \tag{28}$$

with $i = \{\mathrm{p}, O\}$ where p and O relate to position and orientation, respectively. $\mathbf{J}_{\mathrm{p}}(\mathbf{q})$ and $\mathbf{J}_O(\mathbf{q}) = \mathbf{J}_\omega(\mathbf{q})$ are the Jacobian matrices of the linear and angular velocity of the end-effectors, respectively. $\ddot{\mathbf{x}}_i^d$ represents the desired task accelerations, designed by defining error functions associated with each sub-objective of Eq. (18) forcing the dynamics of these error functions to exponentially decrease toward zero to achieve minimization. For instance, to design $\ddot{\mathbf{x}}_O^d$, we defined an orientation error function between the current and desired end-effector ($\mathbf{R}_O(\mathbf{q}_k)$ and $\mathbf{R}_O^d$) using axis-angle representation of the relative orientation. We designed the dynamics of this error such that it exponentially converges toward zero using a stable proportional-derivative (PD) control law. We designed $\ddot{\mathbf{x}}_{\mathrm{p}}^{d}$ as a convex combination of two task components associated with the throwing velocity as

$$\ddot{\mathbf{x}}_{\mathrm{p}}^{d} = \gamma \ddot{\mathbf{x}}_{\mathrm{fvr}}^{d} + (1-\gamma)\ddot{\mathbf{x}}_{\mathrm{mvr}}^{d} \tag{29}$$

where $\ddot{\mathbf{x}}_{\mathrm{fvr}}^{d} \in \mathbb{R}^3$ and $\ddot{\mathbf{x}}_{\mathrm{mvr}}^{d} \in \mathbb{R}^3$ represent task accelerations that provide directions toward feasibility of release velocity and minimum throwing velocity, respectively; γ is a scalar weighting their relative importance. We propose to define $\ddot{\mathbf{x}}_{\mathrm{fvr}}^{d}$ and $\ddot{\mathbf{x}}_{\mathrm{mvr}}^{d}$ as

$$\begin{aligned}\ddot{\mathbf{x}}_{\mathrm{fvr}}^{d} &= -K_p(\dot{\mathbf{x}}_r^{f} - \mathbf{v}_r(\mathbf{q})) - K_d \mathbf{J}_{\mathrm{p}}(\mathbf{q})\dot{\mathbf{q}} \\ \ddot{\mathbf{x}}_{\mathrm{mvr}}^{d} &= -K_d(\mathbf{J}_{\mathrm{p}}(\mathbf{q})\dot{\mathbf{q}} + K_p \mathbf{J}_r^{-1}(\mathbf{x})\mathbf{v}_r(\mathbf{q}))\end{aligned}$$

where $K_p \in \mathbb{R}^{3\times3}$ and $K_d \in \mathbb{R}^{3\times3}$ are positive definite gain matrices, and $\mathbf{J}_r(\mathbf{x}) = \frac{\partial \mathbf{v}_r(\mathbf{x})}{\partial \mathbf{x}} \in \mathbb{R}^{3\times3}$ is the Jacobian of the throwing velocity with respect to the release position.

We mainly consider three stopping conditions for the proposed iterative algorithm. The algorithm stops as soon as a feasible trajectory is found, with a feasible state whose predicted landing position is within a predefined tolerance of the desired landing position. The algorithm stops when the objective function reaches a plateau for number of iterations. The algorithm also stops when the maximum number of iterations is reached. The task–space release state $s_r^* = (\mathbf{x}_r(\mathbf{q}^*),\ \dot{\mathbf{x}}_r(\mathbf{q}^*))$ is obtained through forward kinematics of the joint-space state. Algorithm 1 summarizes the proposed approach.

Algorithm 1: computing optimal feasible throwing state

```
Input: $\mathbf{x}_f^d,\ \mathbf{R}_O^d, \mathbf{G}_o, \mathbf{q}_{min}, \mathbf{q}_{max}, \dot{\mathbf{q}}_{max}, \mathbf{v}_r(\mathbf{x}_f^d, \mathbf{x}_r(\mathbf{q}))$
Output : desired release state $\mathbf{q}^*, \dot{\mathbf{q}}^*$ and
    $s_r^* = (\mathbf{x}_r(\mathbf{q}^*),\ \dot{\mathbf{x}}_r(\mathbf{q}^*, \dot{\mathbf{q}}^f))$
Initialization: $\mathbf{q}_k \leftarrow \mathbf{q}_0, \dot{\mathbf{q}}_k \leftarrow \mathbf{0}, \ddot{\mathbf{q}}_k \leftarrow \mathbf{0}, iter = 0;$

if ($\mathbf{x}_f^d$ is reachable)
    while (true) do
        1: update model: $\mathbf{x}_r(\mathbf{q}_k), \mathbf{J}_i(\mathbf{q}_k), \dot{\mathbf{J}}_i(\mathbf{q}_k)$ with $i = \{\mathrm{p}, O\}$
        2: compute $\bar{\mathbf{x}} = \mathbf{x}_f^d - \mathbf{x}_r(\mathbf{q}_k), \mathbf{v}_r(\bar{\mathbf{x}}), \mathbf{J}_r(\bar{\mathbf{x}})$
        3: compute feasible velocity: $\dot{\mathbf{q}}_k^f$ and $\dot{\mathbf{x}}_r^f(\mathbf{q}_k)$ following (20)
        4: compute task acceleration: $\ddot{\mathbf{x}}_i^d$ with $i = \{\mathrm{p}, O\}$ (29)
        5: compute $\ddot{\mathbf{q}}_k^*$: solve optimization (22)
        6: update state:$\mathbf{q}_k, \dot{\mathbf{q}}_k \leftarrow$ (19)
        7: compute cost function: $\sum l_i'(\mathbf{q}_k, \dot{\mathbf{q}}_k)$

        if (($|\Delta\ell_i'(\mathbf{q}_k, \dot{\mathbf{q}}_k)| \approx 0,\ \forall k = \{k \ldots k + n_{\mathrm{plateau}}\}$)
        or ($iter \geq iter_{max}$))
            $\mathbf{q}^* \leftarrow \mathbf{q}_k, \dot{\mathbf{q}}^* \leftarrow \dot{\mathbf{q}}_k^f$
            $s_r^* = (\mathbf{x}_r(\mathbf{q}^*),\ \dot{\mathbf{x}}_r(\mathbf{q}^*, \dot{\mathbf{q}}^f))$
            break;
        end if
        $iter = iter + 1$
end if
```

4.4. Generation of best feasible release states

Once the inverse throwing map is validated and its corresponding Jacobian is obtained following Eq. (11), we can solve the optimization in Eq. (23) for the best feasible release configurations of the dual-arm robot for tossing the object. Figure 10 shows ten release configurations and the corresponding free-flying trajectories of the object to the desired landing position in task-space.

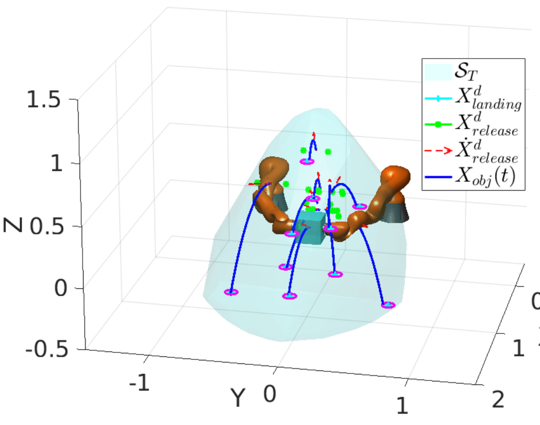


Figure 10: Illustration of ten computed feasible release configurations in 3D (green dot) and corresponding object trajectories (black) to the desired landing position (cyan points). The magenta circle represents a tolerance radius of $0.05\,m$ around the landing position. The red arrows indicate the directions of the 3D release velocities.

The corresponding joint–space release configurations for position and velocity are shown in Figure 11. It is observed that the computed configurations are within the robot joint limits, and thus are kinematically feasible, with some joints reaching their maximum velocities.

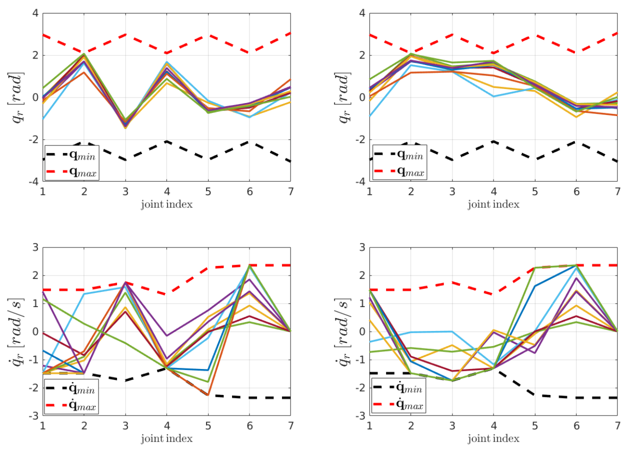


Figure 11: Illustration of joint–space configurations corresponding to ten computed feasible release states: top: joint positions for left and right robotic arms; bottom: joint velocities for left and right arms. The joint limits for position and velocity are shown in red and black dashed lines, respectively.

5. Learning Tossable Space

In this section, we address modeling of the tossable workspace of the dual-arm robotic system. The tossable workspace is the set of all positions reachable by an object if thrown by the robot. We propose to approximate it by modeling the distribution of the reachable positions. Such modeling provides a probability map of the reachability of possible landing positions. It is useful for selecting the best interception positions, chosen as points that represent a high probability of success.

We sampled the forward half space[6] of the robot by randomly generating 10^5 uniformly distributed 3D positions within a radius of $0.2 - 1.75\,m$ and within a cone of $(\pm\frac{\pi}{3} rad)$ around the x axis, as shown in Figure 12. We determined the throwing-reachability of the generated points using the optimization in Eq. (23), which checks for each point whether or not the robot admits a feasible release state. We modeled the probability distribution of the obtained feasible points using a GMM. As with the previously learned projectile dynamics, the number of Gaussian functions, here 13 was determined using the BIC, and the model was trained with the expectation–maximization algorithm initialized with k-means. The likelihood contours are shown in Figure 13.

Thus, a given target position $\mathbf{x}^t$ is considered in the robot tossable space if its likelihood (or probability density function) expressed as

$$\mathcal{P}(\mathbf{x}^t|\mathcal{M}_{toss}) = \sum_{k=1}^{K_t} \pi_k \mathcal{N}(\mathbf{x}^t|\mu_k, \Sigma_k) \tag{30}$$

exceeds a threshold δ_{toss}. We chose this likelihood threshold such that it yields 99% prediction accuracy on the training set of feasible configurations (we obtained a $\delta_{toss} = 0.15$).

Once the tossable space is learned, it is used to determine the interception point. Valid interception points are restricted to the region defined by the intersection of the robot tossable space and the target path, mainly determined by the conveyor belt. We described the target through a set of points and predicted their future positions along the conveyor belt. We chose the best interception location as the position along the target trajectory with the set of points yielding the highest tossable likelihood.

6. Dual-arm Throwing Task Control

Once the interception point is selected and a feasible release configuration is found, the next step is to generate the motion of the robot to execute the throwing task. We controlled the coordinated motion of the dual-arm robot using our previously proposed modulated DS-based controller [4], which allows us to leverage the kinetic energy of the robot through quick grabbing and tossing of the object. In this way, we ensure the dynamic feasibility of the task, which requires that the robot, from its current state with its inertia and that of the grasped object, can accelerate quickly enough to reach the desired release state while remaining within its hardware limits. Thus, we avoid conditions in which the robot holds the object and waits before tossing to intercept the target.

[6]without loss of generality, we only consider the space beyond the forward half plane of the robot's workspace

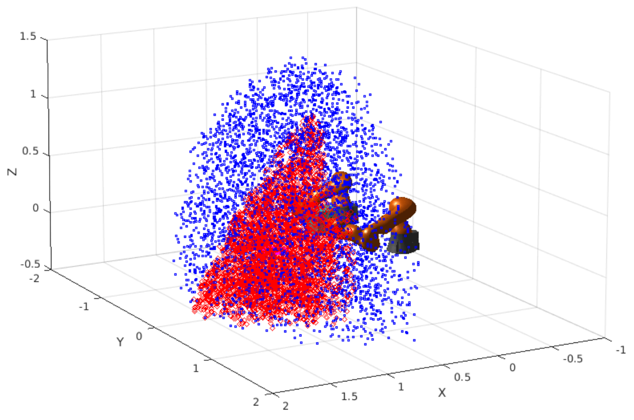


Figure 12: Example of 4000 samples from the generated dataset to learn the tossable space of the dual-arm robot. The red points indicate feasible solutions; the blue points are not feasible according to our algorithm.

6.1. Dual-arm coordinated control

The desired motion of the dual-arm robot in the control framework [4] can be written as

$$\dot{\mathbf{x}} = M(\mathbf{x}) f_n(\mathbf{x}) + f_g(\mathbf{x}) \tag{31}$$

where $\mathbf{x} = \begin{bmatrix} \mathbf{x}^L \\ \mathbf{x}^R \end{bmatrix} \in \mathbb{R}^6$ is the state vector of the DS, with $\mathbf{x}^L$ and $\mathbf{x}^R$ representing the position of the left and right robot arms, respectively, in the dual-arm system. $f_n(\mathbf{x}) \in \mathbb{R}^6$ is the nominal DS, and $M(\mathbf{x}) \in \mathbb{R}^{6\times 6}$ is the state-dependent modulation matrix that locally shapes the motion generated by $f_n(\mathbf{x})$. The $f_g(\mathbf{x})$ term represents the equivalent grasping force projected in the motion space.

In that framework, we proposed a tossing task that consisted of releasing a grabbed object at a desired position with a desired velocity by generating appropriate absolute and relative velocities ($\dot{\mathbf{x}}_d^{abs}$ and $\dot{\mathbf{x}}_d^{rel}$) for the dual-arm system (see [4] for more details).

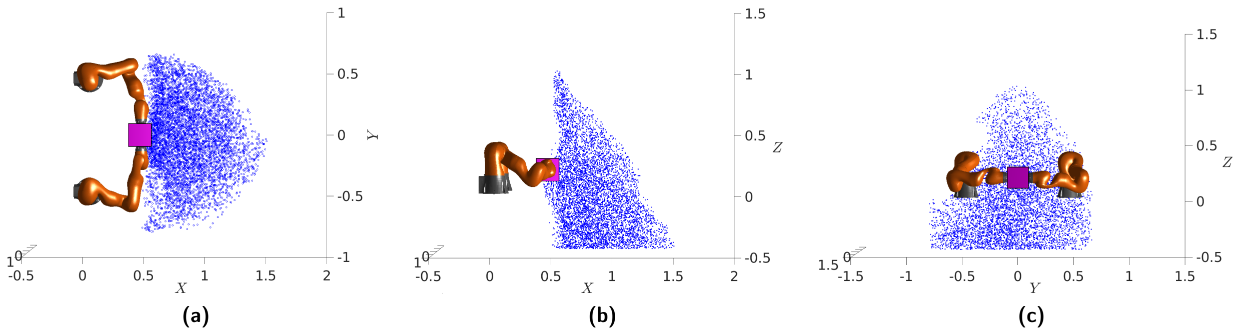
However, regardless of the control strategy, the problem of when to release the target for successful interception must be addressed. To that end, we propose to determine a state of the target that should trigger the robot's movement (reach, pick and toss). We will refer to such a state as the target's "*state-to-go*".

6.2. Estimation of the target's state-to-go

We assume that the target and thrown object intercept at a position $\mathbf{x}_I^*$ and time t_*. Thus,

$$\mathbf{x}^t(t_*) = \mathbf{x}^o(t_*) = \mathbf{x}_I^* \tag{32}$$

where $\mathbf{x}^t$ denotes the target's position, and $\mathbf{x}^o$ denotes the object position. From the robot-object perspective t_* is expressed as the sum of: 1) the duration of the robot motion



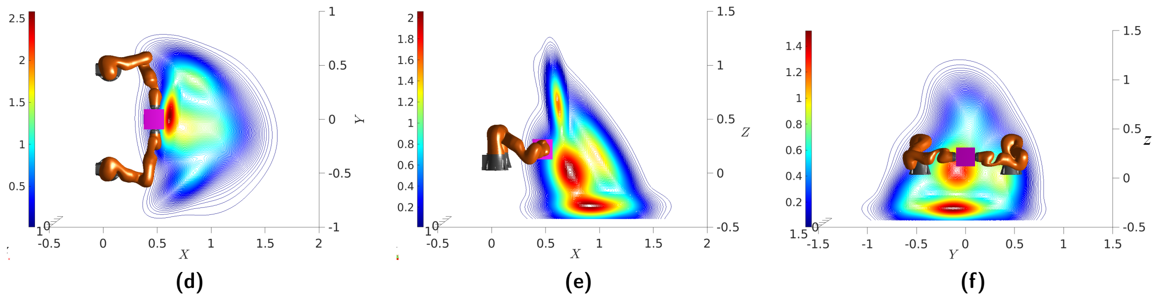


Figure 13: Representation of 2D projections of 3D tossable positions (top) and corresponding likelihood contours obtained by modeling the distribution of tossable points using a GMM with 13 Gaussian functions: (a)–(d): projection on XY plane; (b)–(e): projection on YZ plane; (c)–(f): projection on YZ plane. The likelihood of finding tossable states is lowest in blue regions and highest in red regions. The spread of these regions is not uniform across the workspace, and not symmetric with respect to the two robots (in our configuration, the second and sixth joints are not symmetric between the robots), as they are dependent on the highly nonlinear joint configurations for position and velocity.

from its starting time up to the release time (t_r) of the object, and 2) the free-flying time of the object. Hence

$$t_* = t_r + \Delta t_{ff} \tag{33}$$

where Δt_{ff} is the object's free flying duration from the release time t_r to the landing. We can also express t_* as function of the robot's path to the release position and the robot's average speed along that path as

$$t_* = \frac{L(\mathbf{x}^r(t_r), \mathbf{x}^r(t_0))}{\bar{v}^r} + \Delta t_{ff} \tag{34}$$

where $\mathbf{x}^r$ is the robot position[7], and $L(\mathbf{x}^r(t_r), \mathbf{x}^r(t_0))$ denotes the robot path length from its position at time t_0 ($\mathbf{x}^r(t_0)$) to its position at the release time t_r ($\mathbf{x}^r(t_r)$), expressed as

$$L(\mathbf{x}^r(t_r), \mathbf{x}^r(t_0)) = \int_{t_0}^{t_r} \|\dot{\mathbf{x}}^r(t)\| \, dt \tag{35}$$

where $\|\dot{\mathbf{x}}^r(t)\|$) denotes the L^2-norm of the robot's velocity. In Eq. (34), $\bar{v}^r$ represents the average speed of the robot along its path ($\bar{v}^r = \frac{1}{t_r - t_0}\int_{t_0}^{t_r} \|\dot{\mathbf{x}}^r(t)\| \, dt$).

Given that the target moves along a rectilinear path (the conveyor belt), for the interception to ideally happen at $\mathbf{x}_I^*$ if the robot motion starts at the time t_0, the target's *state-to-go* or the state corresponding to t_0 can be determined as

$$\mathbf{x}^t(t_0) = \mathbf{x}_I^* - \frac{\dot{\mathbf{x}}^t}{\|\dot{\mathbf{x}}^t\|} L(\mathbf{x}_I^*, \mathbf{x}^t(t_0)) \tag{36}$$

where $\frac{\dot{\mathbf{x}}^t}{\|\dot{\mathbf{x}}^t\|}$ accounts for the target's direction of motion. $L(\mathbf{x}_I^*, \mathbf{x}^t(t_0))$ denotes the path length (distance) covered by the target during the interval from t_0 to t_*, it is expressed as a function of the robot's motion as

$$L(\mathbf{x}_I^*, \mathbf{x}^t(t_0)) = \bar{v}^t \left[\frac{L(\mathbf{x}^r(t_r), \mathbf{x}^r(t_0))}{\bar{v}^r} + \Delta t_{ff} \right] \tag{37}$$

where $\bar{v}^t = \frac{1}{t_r - t_0}\int_{t_0}^{t_r} \|\dot{\mathbf{x}}^t(t)\| \, dt$ represents the average speed of the target along the path from $\mathbf{x}^t(t_0)$ to $\mathbf{x}^t(t_*) = \mathbf{x}_I^*$.

Eqs. (36) and (37) indicate that given the intercept point $\mathbf{x}_I^*$ and the feasible release position $\mathbf{x}_r^r = \mathbf{x}^r(t_r)$, the target's state-to-go can be easily estimated with few assumptions. Indeed, what is left to be estimated is: 1) the average speed of the target $\bar{v}^t$, which can be approximated from velocity measurements over a time window; 2) the average speed of the robot $\bar{v}^r$, which can be approximated from the forward integration of the velocity norm of the stable DS that drives the robot from its current position $\mathbf{x}^r(t)$ until the release position $\mathbf{x}_r^r$ is reached; 3) the robot's path length $L(\mathbf{x}^r(t_r), \mathbf{x}^r(t_0))$, which can also be approximated, as in 2), from the DS; 4) the object's free-flying duration Δt_{ff}, which can be estimated from forward integration of the projectile dynamics given the desired intercept location and the computed release state.

[7] for the dual-arm robot, this position refers to the absolute position of the two end-effectors.

6.3. Motion Adaptation

The target's state-to-go estimated in the previous section was based on the average speed of the target and the DS motion. It does not account for possible perturbation of the target motion (slowing down or speeding up). Moreover, errors in motion tracking and prediction (through forward integration) will create divergence between the computed and actual states of the robot. Consequently, the interception will not happen as planned.

To ensure successful completion of the task, we propose a twofold adaptation strategy with, first, a velocity modulation strategy and second, an attractor adaptation strategy.

6.3.1. Velocity modulation

The DS can be accelerated or decelerated at will by multiplying the function by a positive scalar. This does not affect the stability properties at the attractor. Thus, we can adapt the robot's DS-based velocity at run time to adapt to changes in the velocity of the moving target as follows:

$$\dot{\mathbf{x}}_d = \beta(\mathbf{x})M(\mathbf{x})f_n(\mathbf{x}) + f_g(\mathbf{x}) \tag{38}$$

where $\beta(\mathbf{x})$ is the adaptation factor, which is simply a state-depend scaling factor computed as

$$\beta(\mathbf{x}) = \frac{\bar{v}^t}{\bar{v}^r}\left|\frac{L(\mathbf{x}^r(t_r), \mathbf{x}^r(t))}{L(\mathbf{x}_I^*, \mathbf{x}^t(t)) - \bar{v}^t.\Delta t_{ff}}\right| \tag{39}$$

with $\beta(\mathbf{x})\left\|\dot{\mathbf{x}}_d\right\| \leq \|\dot{\mathbf{x}}\|_{max}$, such that the adapted robot velocity does not exceed the maximum allowable velocity. $\beta(\mathbf{x}) \geq 0$ to preserve the stability of the DS.

6.3.2. Adaptation of attractor

Although the velocity modulation strategy can slow down or speed up the velocity of the robot, it cannot, however, reverse the robot's motion direction. Such reversal may be useful, for instance to force the robot to retract to an initial position. The robots needs to quickly accelerate to toss at the desired throwing speed. It may not be able to do so, if the path is too short, as joint limits may be reached. we propose additionally an adaptation strategy that adapts the attractor of the nominal DS $f_n(\mathbf{x})$ when the target's velocity changes its direction. Hence, we define the attractor as

$$\mathbf{x}_* = \alpha(\mathbf{x}^t, \dot{\mathbf{x}}^t)\mathbf{x}_d + (1 - \alpha(\mathbf{x}^t, \dot{\mathbf{x}}^t))\mathbf{x}_{stb} \tag{40}$$

where $\mathbf{x}_d$ is the desired attractor of the nominal DS, and $\mathbf{x}_{stb}$ denotes a standby attractor to which the robot should retract to. $\alpha(\mathbf{x}^t, \dot{\mathbf{x}}^t) \in [0, 1]$ is a target's state-dependent scalar function that goes to 1 or 0 depending on whether the target moves in the direction of the interception or not. We defined $\alpha(\mathbf{x}^t, \dot{\mathbf{x}}^t)$ as

$$\alpha(\mathbf{x}^t, \dot{\mathbf{x}}^t) = \frac{1}{1 + e^{-a((\mathbf{x}_I^* - \mathbf{x}^t)^\top \dot{\mathbf{x}}^t)}} \tag{41}$$

where $a > 0$ represents the steepness factor of the function defined by $\alpha(\mathbf{x}^t, \dot{\mathbf{x}}^t)$.

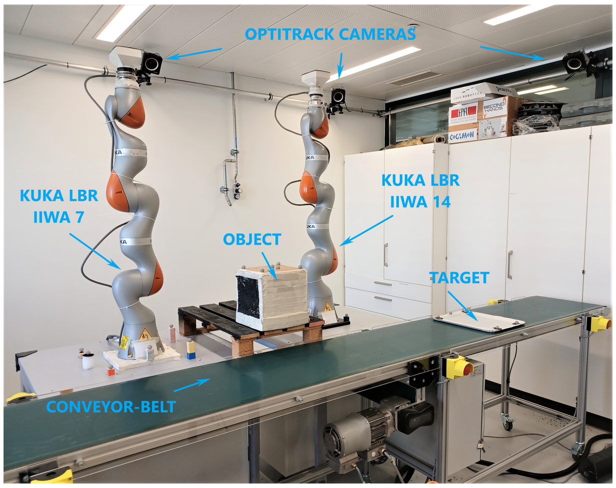


Figure 14: Experimental setup used for validating the proposed control scheme. The image shows the two KUKA robots, the conveyor belt, some Optitrack cameras, the target, and the object on a pallet.

7. Experimental Validation

To validate the proposed approach, we used the same robotic setup as in our previous work [4], a dual-arm system consisting of a pair of KUKA LBR IIWA7 and IIWA14 robots. To move the target, we used a conveyor belt with velocity ranging from 0.05 m/s to 1.5 m/s. The object and target position were measured using an *Optitrack* motion capture system; velocity information was estimated using a Savitzky-Golay [72] smoother and a Kalman filter. We used a 1.9-kg cubic box with dimensions of 0.26 m on each side.

The target speed ranged from 0.10 m/s to 0.450 m/s, determined experimentally to avoid collision between the moving object after landing and a robot while retracting after releasing the object. Speeds of up to 0.65 m/s were reached with a smaller box (see video). We used a flat tray with dimensions of 0.40 m × 0.30 m as the target. Figure 14 shows the robotic setup used for our experimental validation.

In the implementation, the DS and the feasible release state are updated every 5 ms and 100 ms, respectively. We used qpOASES [73] to solve the optimizations (20) and (22). The whole feasibility algorithm is solved between 3 to 25 ms on a $Intel^{(R)}$ *Core i7*, 3.4 GHz and 7.8 GB RAM PC.

The validation process was both simulated and conducted using an actual robot. Three main tasks were considered: (i) tossing an object to a target moving at a constant velocity; (ii) tossing an object to a target moving with a changing velocity; (iii) comparison of placing an object and tossing an object onto a moving target. A video (`https://youtu.be/8a4AFDYfrXo`) of the corresponding experiments is provided as supplementary material[8] and the code[9] made available.

[8] `https://youtu.be/8a4AFDYfrXo`
[9] `https://github.com/epfl-lasa/iam_dual_arm_control`

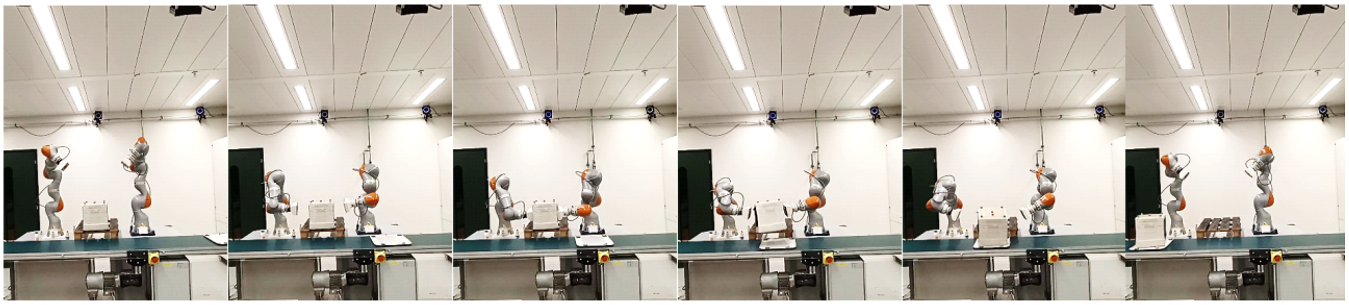


Figure 15: Snapshots of dual-arm grabbing and object tossing (white cubic box) onto a moving target (white tray on conveyor belt). At first, the robots wait for the target to reach the estimated state-to-go before moving toward the object to grab it and toss it to the target moving on the conveyor belt. From left to right, the snapshots show the dual-arm system waiting, approaching the object, grabbing the object, tossing the object, intercepting the moving target and landing, and both moving together.

7.1. Tossing object to target moving at constant velocities

The goal of this task is to evaluate the accuracy and repeatability of dual-arm based positioning through tossing of an object to a target that moves at different constant velocities. This task simulates a hypothetical depalletizing task on an already loaded conveyor belt, where the robot must place an object in available free space moving on the conveyor belt.

Initially, the robot remains stationary as the target approaches until the target reaches the state-to-go as determined in Eq. (36) in Section 6.2. Once reached, the dual-arm system moves, grabs the object, and throws it to the desired intercept position as shown in Figure 15; task sequences such as initiation of target movement and robot movement, throwing of the object, interception, and unified movement of the object and target are observed in the snapshots.

Corresponding plots of positions and velocities of the object and target are shown in Figure 16 (top) and (bottom), respectively. In Figure 16-(top), the initial position offset and correspondence between the object and target after landing are shown; the y-coordinates (representing the direction of conveyor belt movement) decrease continuously (negative speed). The constant offset of the z-coordinate results from measurement of the object position at the center, producing a height offset with the target on the conveyor belt. Small oscillations result from bouncing of the object on the target caused by landing impact before stabilizing. In Figure 16-(bottom), one can notice that the object's release velocity is relatively small[10]. Afterward, the object decelerates rapidly under gravity before changing suddenly its velocity direction from negative to positive at the landing impact.

A 3D illustration is shown in Figure 17, including the trajectories described by the robotic system during the task, the object, and the target, indicated in red, black, and green, respectively. The end-effector trajectories indicate complete cycles from standby positions and back after grabbing and tossing the object.

[10]In such a case, the aerodynamic drag force on the object is negligible.

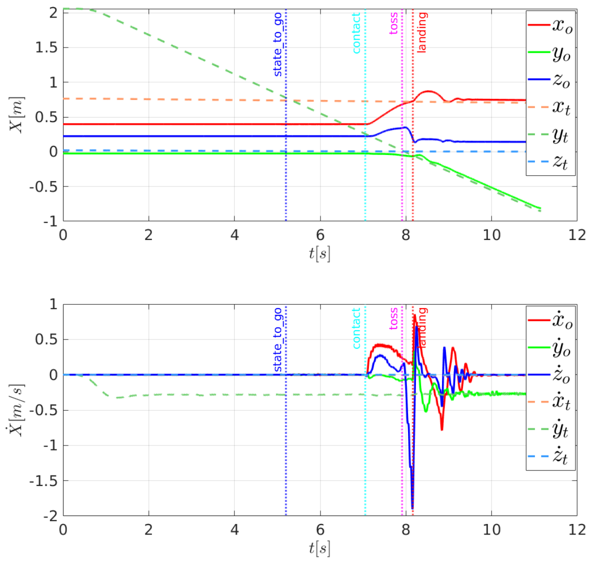


Figure 16: Position and velocity plots of object and target throughout the task. The target moves in the y-direction at a speed of -0.3 m/s (green dashed line in bottom plot); its starting position [0.75 m, 2.0 m, 0.25 m] decreases linearly in the y-direction (green dashed line in top plot); all other coordinates remain constant. The object starts moving only after contact with the robots.

The accuracy of the proposed scheme in intercepting moving targets is shown in Figure 18, which reports intercept position errors across the XY plane for ten experiments for each target speed. The norms of the errors per speed are shown in Figure 18-(a); the x and y components of the error contributing to the norm are shown in Figure 18-(b). The mean of the intercept error norm for each speed was less than 0.06 m, representing one-fifth of the target width (0.30 m). The variance in black indicates that few cases exceeded 0.06 m, remaining within a tolerance of 0.10 m.

The 3D distribution of intercept positions defined by the target and object at release and landing is shown in Figure 19. Intercept locations are indicated with respect to feasibility and tossable workspace described in Sections 4 and 5.

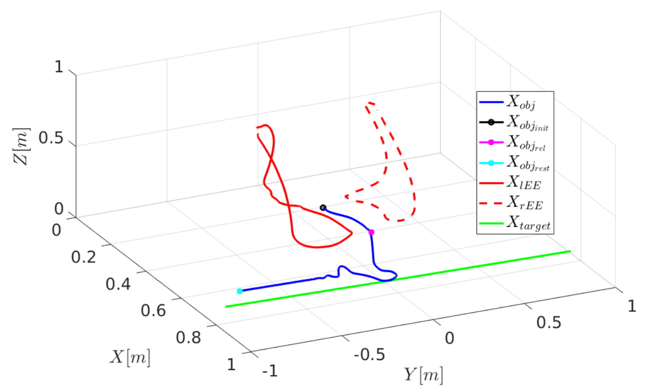


Figure 17: 3D trajectories of system in grabbing and tossing an object onto a moving target. The trajectories described by the end-effectors are shown in red (solid for left, dashed for right); the target trajectory is represented in green, the object trajectory in black. The initial position of the object is shown in black, the release position in magenta. The rest position when the conveyor belt stops is shown in cyan.

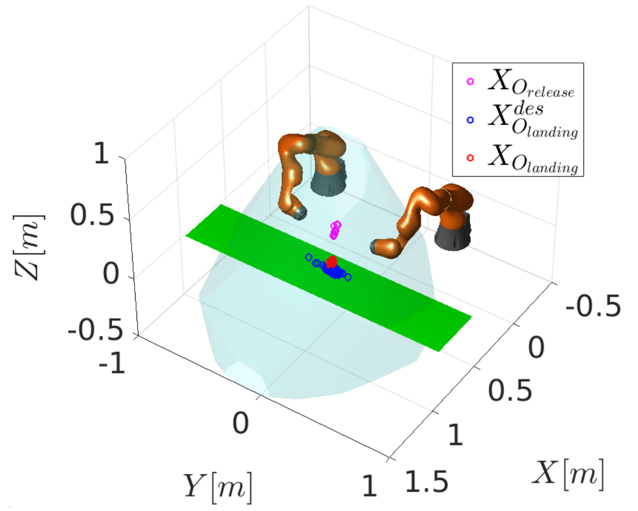
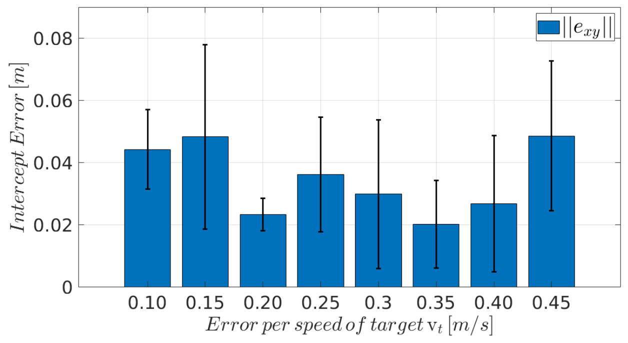
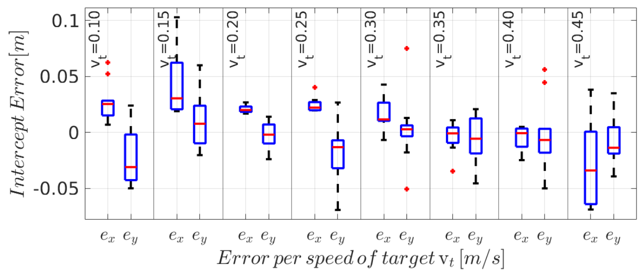


Figure 19: Relative 3D distribution of intercept positions defined by the target (black), the object release position (magenta), and the object landing location on the conveyor belt (red) for the dual-arm system and its kinematically feasible tossing workspace (light black)



(a)



(b)

Figure 18: Intercept position error per speed across the XY plane (the z coordinate is not considered as the object height offset from its center is constant): (a) norm of intercept error per target speed; (b) intercept error per coordinate and per target speed. The intercept error for x-coordinates is denoted as e_x; the error for y-coordinates is denoted as e_y

7.2. Tossing object on target moving with changing velocities

The goal of this experiment is to assess the robustness and adaptivity of the proposed algorithm to changes in the target velocity. In other words, we evaluate how the proposed control strategy adapts the motion of the dual-arm system carrying the object for successful interception in presence of changes in the target motion.

To that end, while changing the velocity of the target, we started by testing the algorithm without adaptation ($\beta(\mathbf{x}) = 1$ in Eq. (38)) and then activated the adaptation scheme.

The changing target speed was designed with a constant nominal component v^t_{nom} as in the previous experiment, and with a changing perturbation component v^t_{pert}, defined as $v^t_{pert} = a_{pert} \cdot sin((\omega_p + 0.2 \cdot \text{rand}(\omega_p))t)$, where a_{pert} denotes the maximum amplitude of the perturbation; ω_p denotes its angular frequency, and t is the running time of the algorithm. We have conducted 20 experiments for each case; we set the following target speed parameters: $v^t_{nom} = 0.30$ m/s, $a_{pert} = 0.15$ m/s and $\omega_p = 2\pi$.

7.2.1. *Case without adaptation*

Controlling robot motion based only on the state-to-go generally leads to failed interception of the target as soon as perturbation affects the system. Such a strategy amounts to open-loop control of interception, which only works if the conditions that predicted the state-to-go remain the same after the dual-arm system has initiated motion.

An example is illustrated in Figure 20. The X and Y position and velocity plots for the object and target are presented in Figure 20-(a). The norm of the linear velocities of the two robots triggered when the target reached the estimated state-to-go is shown in Figure 20-(b). The tossed object fails to intercept the target as the dual-arm motion generation ignores the changes in target speed.

7.2.2. *Case with adaptation*

In this case, the adaptation scheme modulates the motion of the robot, slowing it down or speeding it up based on a

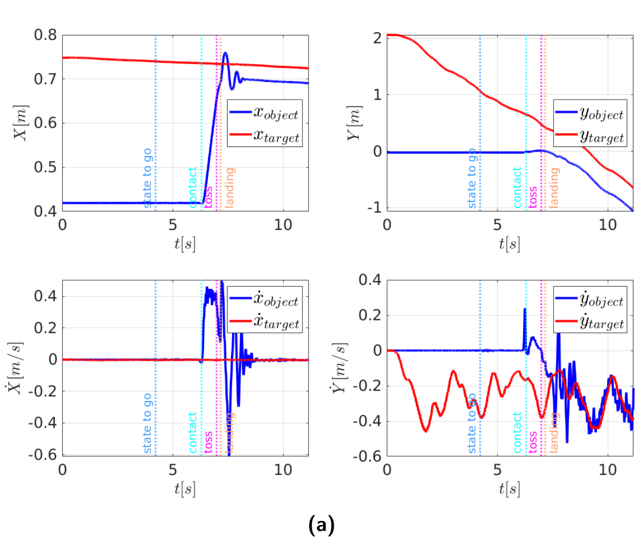
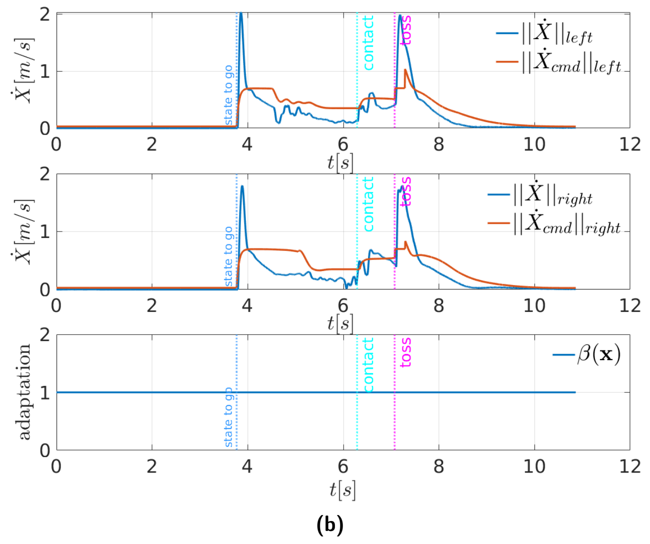



Figure 20: Position and velocity plots of dual-arm system grabbing and tossing an object onto a moving target with motion perturbation and **without adaptation** of robot motion: (a) X and Y evolution over time for position (top) and velocity (bottom) of the target (red) and object (black); (b) norm of linear velocities of left robot (top), right robot (middle), and adaptation factor $\beta(\mathbf{x})$ (bottom). The robot motion remains unaffected by changes in target speed.

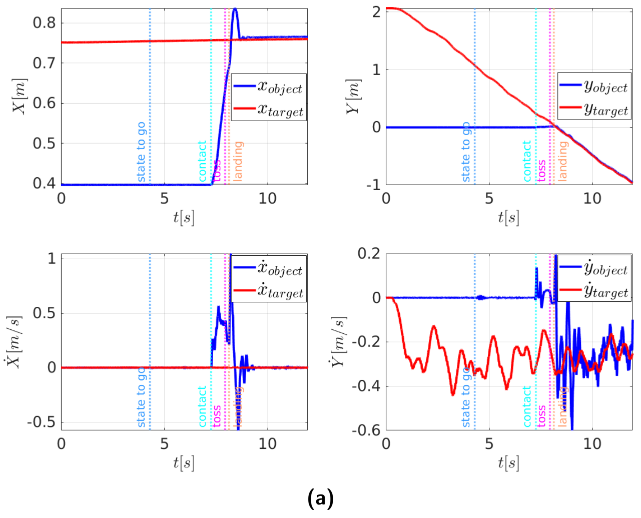
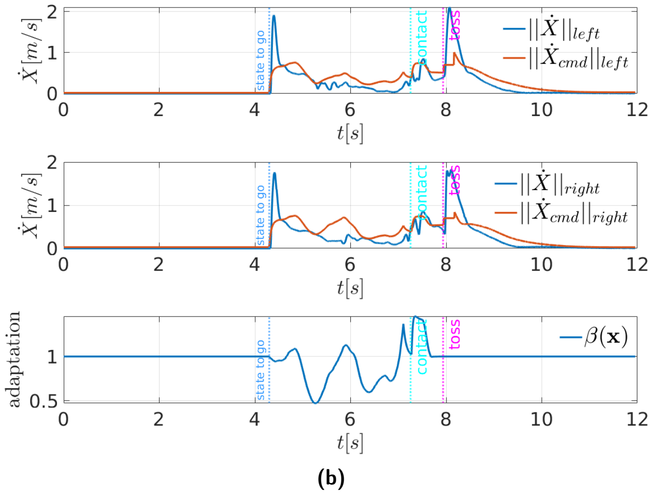



Figure 21: Position and velocity plots of dual-arm system grabbing and tossing an object onto a moving target with motion perturbation and **adaptation** of robot motion: (a) X and Y evolution over time for position (top) and velocity (bottom) of the target (red) and object (black); (b) norm of linear velocities of left robot (top), right robot (middle), and adaptation factor $\beta(\mathbf{x})$ (bottom). (bottom). The robot motion was modulated based on estimated changes in target speed to ensure successful interception.

continuously updated prediction of interception using current target and robot states. We evaluated adaptation with two types of perturbation. The first perturbation of target motion was similar to that in the case without adaptation; the second perturbation was caused by manually stopping, pulling back, or pushing forward the target as it moved on the conveyor belt.

Velocity-based perturbation The position and velocity plots of the object and target are shown in Figure 21-(a). The object successfully intercepts the target despite changes in its speed (see accompanying video). Velocity plots of the left and right robot end-effectors and the adaptation factor $\beta(\mathbf{x})$ are shown in Figure 21-(b), at the top, middle, and bottom, respectively.

Compared to the case without perturbation, the effect of $\beta(\mathbf{x})$ on the dual-arm velocities is clearly observed. The intercept position errors between the object and target with and without adaptation are shown in Figure 22; error histograms in the x-direction are shown in the left plot, and error histograms in the y-direction are shown in the right plot for 20 experiments for each case.

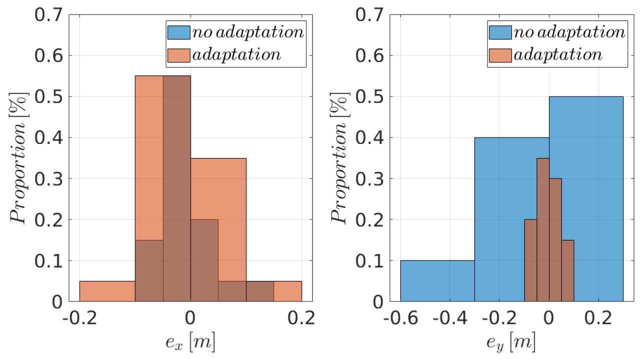


Figure 22: Distribution of intercept position errors in the x-direction (left) and y-direction (right) in tossing an object onto a moving target with motion perturbation without adaptation (black) and with adaptation (orange). The distributions were derived from 20 experiments with and without adaptation. The x-direction represents the main tossing direction; the y-direction represents the interception direction in which the target moves.

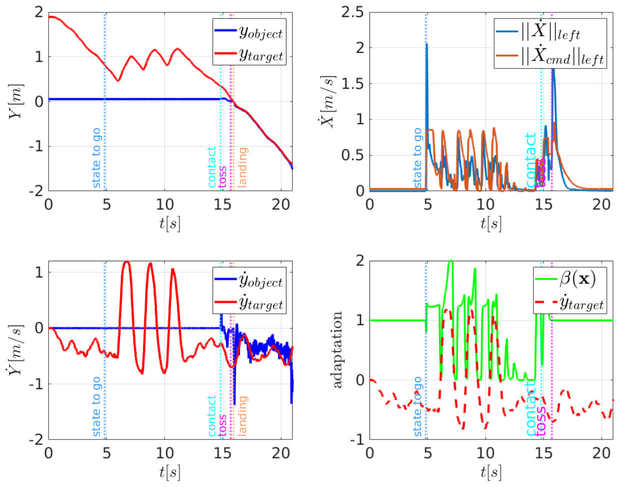


Figure 23: Position and velocity plots of dual-arm system with motion adaption while grabbing and tossing an object onto a moving target **with manual perturbation**: left: y-evolution of position (top) and velocity (bottom) of target (red) and object (black), respectively, over time; (right): norm of linear velocity of left robot (top) and adaptation factor $\beta(\mathbf{x})$ and y-velocity of target (bottom). The robot motion was modulated based on estimated changes in target speed to ensure successful interception.

In both cases, the mean of the error is about 0. However, considering the variance, the error in the interception direction (Y) ranges from -0.45 m to 0.20 m without adaptation, and between -0.10 m and 0.10 m with adaptation; 13 out of 20 Y-position errors are within [-0.05, 0.05] m. However, in the main tossing direction (X), the error with adaptation is slightly greater than the error without adaptation; 10 out of 20 experiments had an absolute error between 0.05 m and 0.10 m (two had errors of 0.11 m). This is mainly due to the change in momentum of the object before its release as the robot velocities are modulated. One solution to mitigate this effect is to stop modulation (set $\beta(\mathbf{x}) = 1$) as soon as the object is near the release position.

Without adaptation, the object successfully intercepted the target with a position error within the tolerance of 0.10 m in five out of 20 experiments. However, as the system was in open loop, the observed interceptions had a stochastic nature, stemming from the randomness introduced in the speed perturbation, and may have yielded target positions near the desired intercept location at the landing time of the tossed object.

Human interaction-based perturbation In this case, the target speed perturbation was manually induced through interaction with the target (see accompanying video). Figure 23 shows position and velocity plots for the target–object–robot system with perturbation and with adaptation to compensate for it. The y-components of the positions and velocities of the target and object are shown at the top-left and bottom-left, respectively. The linear velocity norm of the left end-effector is shown on the top-right (the right end-effector is not shown, but follows a similar pattern); the adaptation factor $\beta(\mathbf{x})$ and the y-velocity of the target that drives it are shown at the bottom-right.

Unlike the previous perturbation cases, the target velocity changes in sign from negative to positive and vice versa according to the perturbation. As $\beta(\mathbf{x}) \geq 0$, the modulation cannot reverse the motion direction of the robot (a negative $\beta(\mathbf{x})$ will make the system unstable). Thus, retraction of the robot is achieved by smoothly changing the attractors of the dual-arm system between the grabbing points on the box and the predefined standby position of the end-effectors.

Human interaction with the target, and which induced speed perturbation shown previously in Figure 23-(bottom) is illustrated in snapshots of Figure 24. The target is pulled back three times, as shown in Figures 24(b)–(c), (d)–(e), and (f)–(g), at $t = 5.95s$, $t = 8.08s$, and $t = 9.98s$, respectively, producing the three velocities shown in Figure 23-(bottom).

7.3. Comparison of placing and tossing object onto moving target

The goal of this experiment was to compare the kinetic and energy efficiencies of the widely used picking and placing operation with those of the proposed picking and tossing of objects onto a moving target (on conveyor belt). The overall energy consumption is estimated as follows

$$E_{dual} = \sum_{i}^{n_{DoF}} \left\{ \int_0^{T_c} |\dot{q}_i||\tau_i|dt \right\} \tag{42}$$

where $\dot{q}_i$ and τ_i represent the joint velocity and torque, respectively. T_c denotes the cycle time of the dual-arm task. Unlike our previous study [4], where such a comparison was conducted from the standby position to the release of the object, in this study, the comparison includes the entire

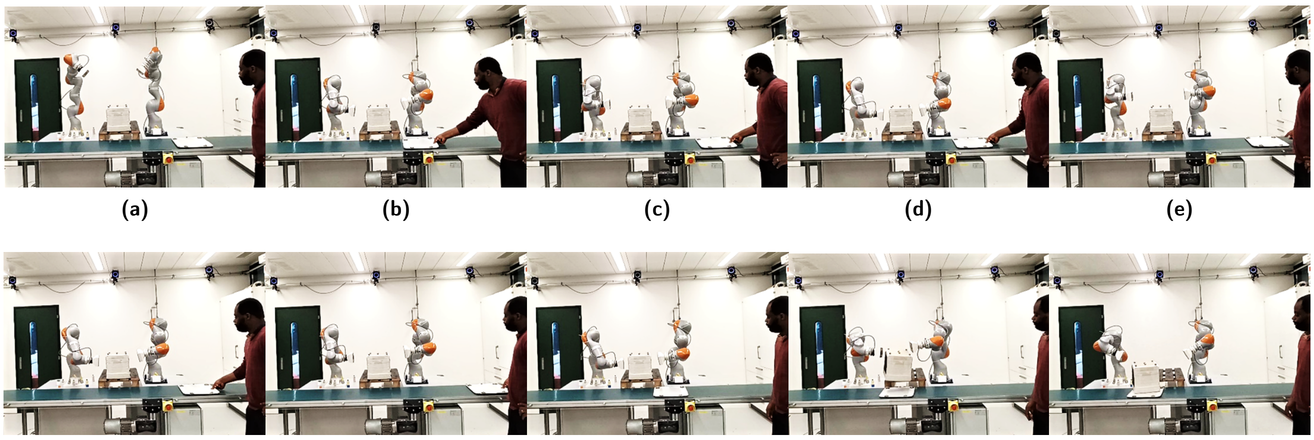


Figure 24: Snapshots illustrating adaptation of dual-arm system to manual perturbations of target motion in grabbing and tossing of an object onto a moving target: (a): dual-arm in standby, waiting for target to reach estimated state-to-go; (b)–(c), (d)–(e), and (f)–(g): perturbation introduced by manually pulling back moving target, causing retraction of robots; (h)–(i): grabbing and tossing of object as target moves; (j): motion of object and target after successful interception.

cycle (from the standby position and back after executing the motion), at target speeds ranging from 0.1 m/s to 0.450 m/s in increments of 0.05 m/s. We conducted ten experiments at each target speed and estimated the cycle time and energy consumption of the dual-arm robotic system. The comparison results are shown in Figure 25; Figure 26 shows the energy consumption comparison. The results indicate that the proposed picking and tossing produces a shorter cycle time and consumes less energy than the picking and placing operation, consistent with our previous research [4]. Similar consistent results were observed across target speeds for cycle time and energy expenditure. These experiments were conducted with no perturbation of target speed; thus, the main control variable was the estimated state-to-go, with adaptation having little effect. Slight variations in results were caused by noise, state estimation, and control errors.

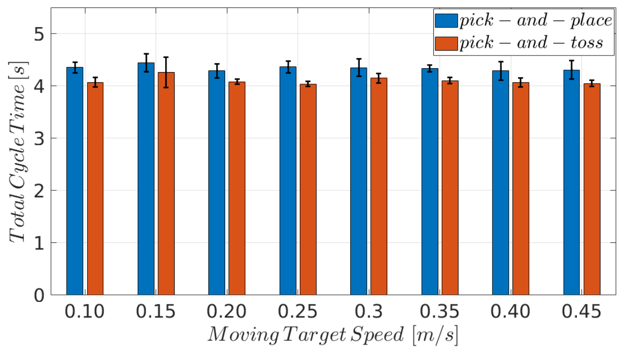


Figure 25: Comparison of cycle time for picking and placing and picking and tossing of an object onto a moving target at different speeds

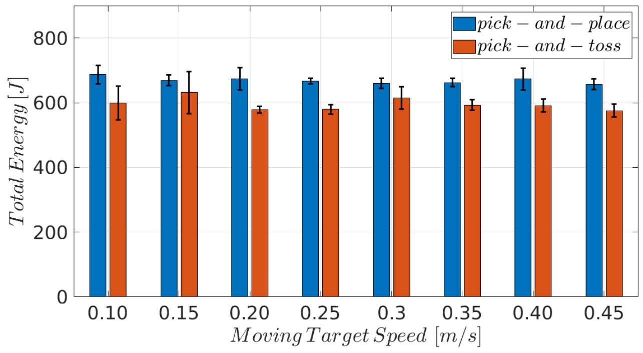


Figure 26: Comparison of total energy consumption for picking and placing and picking and tossing of an object onto a moving target at different speeds

The main results of the comparison of picking and placing and picking and tossing are summarized in the histogram shown in Figure 27. The proposed picking and tossing is approximately 5.5% faster and consumes approximately 11% less energy than the picking and placing operation for the same positioning task.

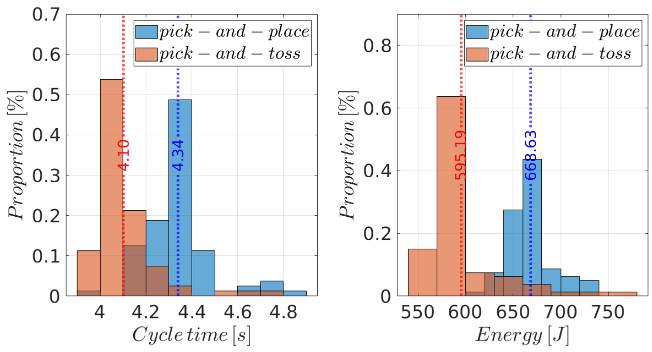


Figure 27: Histograms summarizing cycle time and energy expenditure comparisons in picking and placing and picking and tossing of an object onto a moving target

8. Discussion and Conclusion

In this paper, we have presented a control strategy enabling a dual-arm robotic system to pick up and toss an object onto moving target locations on a conveyor belt. Positioning objects on moving conveyor belts is common in industry, and can represent depalletizing of parcels in a sorting facility. To achieve precise dynamic positioning, we addressed this problem from an interception perspective; a bi-manually tossed object governed by projectile dynamics was intended to land at a desired position along the path of a moving target. Using GMR, we learned an inverse throwing map from the nonlinear projectile dynamics to determine the tossing parameters (release position and velocity) necessary to reach the desired landing position, which serve as reference inputs for the robotic system (Section 4.1). To ensure feasibility of the release state, we embedded the learned throwing map into a bi-level kinematics-based optimization framework. We translated and solved such a problem at the acceleration level allowing to enforce concurrently velocity and position feasibility constraints with off-shelves solvers (Section 4.3). Building upon the obtained release state feasibility algorithm, we proposed a method to model the tossable workspace of a dual-arm robot, representing the set of all positions reachable by an object tossed by the robot. We generated 10^5 desired positions within and outside the dual-arm system workspace and determined for each whether our algorithm could find a corresponding feasible kinematic release state. We derived a closed-form model of the tossable workspace by learning the distribution of all feasible landing positions using a GMM. The obtained model allows us to predict the the probability of reaching potential intercept positions of the object with the target before initiating the robot motion (Section 6). For robust tossing, we used our previously developed dynamical system-based control framework [4], we complemented it with an adaptation strategy to modulate the generated motion of the dual-arm system and enable grabbing and tossing of an object onto a moving target under live perturbations in speed or in position. Moreover, to accommodate objects with different masses and shapes, the learned inverse throwing map (Eq. (9) and Eq. (43)) was parameterized by η (an aerodynamic parameter defined in Eq. (8) that captures the object's characteristics). For each new object, one needs only to update the parameter η with the corresponding new mass, aerodynamic coefficient C_D, and the cross-sectional area A_o. Thus, the learned model can generalize as long as the obtained value of η is within its training range.

To demonstrate the validity of the proposed approach, in addition to simulations, we conducted experiments using a pair of actual KUKA robots. We evaluated the accuracy and repeatability of interception at different target speeds, with and without motion perturbations. To highlight the benefits of our approach with respect to classical positioning tasks based on pick-and-place operations, we implemented a picking and placing strategy for an object on a moving target and compared it in terms of cycle time and energy consumption with the proposed picking and tossing strategy. We found that in the absence of target's motion perturbation and for the selected release configurations shown in Figure 19, the object landed within a radius of 0.05 m from the center of the target in 67 out of 80 tossing cases, that is 77%; the remaining tosses were within a radius of 0.10 m.

With target motion perturbation and no adaptation of the robot motion, the ratio of tosses with interception error norms less than 0.10 m decreased to 20%, mainly due to increased errors in the intercept direction (direction of target velocity); 75% of tosses were between 0.10 and 0.45 m from the target; in the tossing direction (object's release velocity direction), 95% of tosses were within 0.10 m of the target center. With the proposed adaptation scheme, although errors in the tossing direction may slightly increase, we demonstrated that the intercept error norms decreased significantly; 90% of tosses were within a 0.10 m radius of target.

When compared to the traditional pick and place strategy, we found that the proposed pick and toss approach leads to a shorter cycle time (5.5% in our experiments). This is consistent with previous results in [3, 7, 4]. Moreover, as in [4], we also found that the proposed approach consumes less energy (11%).

It is worth noting that a systematic comparison of our approach to state-of-the-art pick-and-toss solutions could not be performed as none of them use a dual-arm robotic system for tossing. The same is true for the industry, where most of the automated depalletizing solutions use either specialized equipment or single robotic arms equipped with tools specially adapted to the types of products to be depalletized. The current dual-arm solutions to depalletizing rely mainly on human operators, whose average depalletizing speed of cases is 500 picks per hours (pph) [74]. This translates into a cycle time of about 7.2s, whereas our approach with its current setup and non-optimized robot trajectories has a cycle time of 4s.

Although the effectiveness of the proposed method was demonstrated using actual robots, there are some limitations in the approach and implementation that need to be considered. For example, to handle moving targets with different orientations, our current implementation focuses essentially on the landing position rather than the landing pose (position and orientation). It has been developed assuming a fixed release orientation of the thrown object with no rotation or tumbling motion. If the landing orientation of the thrown object is required to match a desired target's orientation, the proposed approach can only handle such a requirement if there exists a feasible release pose of the object that has the desired orientation of the target. In such a case, the best interception point (point with the highest likelihood of task success) obtained from the learned tossable workspace when ignoring the target's orientation might no longer be valid. To address the latter problem, a possible solution to consider in future work could be expanding the dimensionality of the learned tossable workspace (used for the determination of the interception point) from three dimensions (3D positions) to four dimensions by including the planar orientation of the target.

Moreover, while our current approach focuses on computing kinematically feasible release states, their dynamical feasibility is not guaranteed. Dynamical feasibility depends on the initial robot configuration and the inertia of the object. Moreover, the geometric properties of the object and dual-arm grabbers may limit the feasible states due to collisions at the release time if the robot cannot retract fast enough. The uncertainty in the release time should also be considered as it affects interception accuracy.

In learning the inverse throwing map, we assumed a more complex object's dynamics by considering the non-linear aerodynamic drag force. This was justified by the necessity to generate rich data capable of describing the behavior of objects with different aerodynamic characteristics (shapes and masses) captured by the parameter η. However, in our experiments, the tossing velocities were relatively low, thus yielding negligible drag forces for the considered objects. Therefore, for similar situations, one could have used a simpler linear throwing model without compromising accuracy.

In determining the release configurations, we considered the free-flying dynamics of the object after release but did not consider the impact dynamics of the object at landing. Thus, we observed that for some feasible release states determined by our algorithm, the tossed object bounced on the conveyor belt and fell from it instead of resting on it. To prevent this, we constrained the search space of feasible release configurations to a small set of task-space release positions near the conveyor belt. Thus, the associated release velocities were reduced and the landing impacts too. A more general solution in determining the release state should therefore consider the desired post-landing impact state of the object beyond its simple landing state. In the meantime, with our current scheme's inability to forecast the effects of landing impacts on the tossed objects, application-wise, it should be used only for objects that will not break under the induced impact or for cases where such damage has no serious implications for the objects, as discussed in [4]. When the landing impact is deemed unsuitable, we recommend using instead the dual-arm grabbing and placing strategy. Such a situation has been illustrated in the accompanying video, where a grabbing and tossing strategy was used for the open box of loose bottles (waste to be recycled or disposed of), whereas a grabbing and placing strategy was used for a pack of filled beer bottles.

Thus, to guarantee optimal and safe behavior using the proposed approach and promote its use in industry, future research should consider the full dynamics of the robots and the dynamics of their interactions with the environment. Moreover, the flexibility of dual-arm systems needs to be increased as they have a major drawback when mounted on fixed bases. Their joint workspace is reduced and this limits their sphere of operations. The solution to expand their joint workspace is to augment the degrees of freedom of their common base, or better endow the dual-arm system with mobility.

Acknowledgment

This work was supported in part by the European Union's Horizon 2020 research and innovation program under grant agreement No. 871899. I.AM. project: `https://i-am-project.eu/`.

9. Appendix

9.1. Jacobian of Inverse throwing map

To account for objects with different mass and aerodynamic properties, the inverse throwing map is parameterized by η and can be written as

$$\mathbf{v}_r \approx \sum_{k=1}^{K} h^k(\bar{\mathbf{x}}^{\mathrm{o}};\eta)\tilde{\boldsymbol{\mu}}^k_{\dot{\mathbf{x}}^{\mathrm{o}}|(\bar{\mathbf{x}}^{\mathrm{o}};\eta)}(\bar{\mathbf{x}}^{\mathrm{o}};\eta) \tag{43}$$

where $\tilde{\boldsymbol{\mu}}^k_{\dot{\mathbf{x}}^{\mathrm{o}}|(\bar{\mathbf{x}}^{\mathrm{o}};\eta)} = \boldsymbol{\mu}^k_{\dot{\mathbf{x}}^{\mathrm{o}}} + \Sigma^k_{\dot{\mathbf{x}}^{\mathrm{o}}\bar{\mathbf{x}}^{\mathrm{o}}}\Phi^k_{\bar{\mathbf{x}}^{\mathrm{o}}\bar{\mathbf{x}}^{\mathrm{o}}}(\bar{\mathbf{x}}^{\mathrm{o}} - \boldsymbol{\mu}^k_{\bar{\mathbf{x}}^{\mathrm{o}}}) + \xi^k_{\dot{\mathbf{x}}^{\mathrm{o}}\eta}$ and where $\xi^k_{\dot{\mathbf{x}}^{\mathrm{o}}\eta}$ is defined as

$$\begin{aligned}\xi^k_{\dot{\mathbf{x}}^{\mathrm{o}}\eta} &= \Sigma^k_{\dot{\mathbf{x}}^{\mathrm{o}}\eta}\Phi^k_{\eta\bar{\mathbf{x}}^{\mathrm{o}}}(\bar{\mathbf{x}}^{\mathrm{o}} - \boldsymbol{\mu}^k_{\bar{\mathbf{x}}^{\mathrm{o}}}) \\ &\quad + [\Sigma^k_{\dot{\mathbf{x}}^{\mathrm{o}}\bar{\mathbf{x}}^{\mathrm{o}}}\Phi^k_{\bar{\mathbf{x}}^{\mathrm{o}}\eta} + \Sigma^k_{\dot{\mathbf{x}}^{\mathrm{o}}\eta}\Phi^k_{\eta\eta}](\eta - \boldsymbol{\mu}^k_{\eta})\end{aligned}$$

The Jacobian of $\mathbf{v}_r$ with respect to $\bar{\mathbf{x}}^{\mathrm{o}}$ and parametrized by η can be written as

$$\mathbf{J}_{\mathbf{v}_r}(\bar{\mathbf{x}}^{\mathrm{o}}) = \frac{\partial \mathbf{v}_r}{\partial \bar{\mathbf{x}}^{\mathrm{o}}} = \sum_{k=1}^{K_{\mathrm{v}}}\left[(c^k_{\mathbf{v}\bar{\mathbf{x}}} + \mathrm{S}^k_{\mathbf{v}\bar{\mathbf{x}}}\bar{\mathbf{x}}^{\mathrm{o}})\frac{\partial h^k(\bar{\mathbf{x}}^{\mathrm{o}})}{\partial \bar{\mathbf{x}}^{\mathrm{o}}} + h^k(\bar{\mathbf{x}}^{\mathrm{o}})\mathrm{S}^k_{\mathbf{v}\bar{\mathbf{x}}}\right] \tag{44}$$

where the matrices $\mathrm{S}^k_{\mathbf{v}\bar{\mathbf{x}}}$ and vectors $c^k_{\mathbf{v}\bar{\mathbf{x}}}$ are given by

$$\begin{aligned}\mathrm{S}^k_{\mathbf{v}\bar{\mathbf{x}}} &= \Sigma^k_{\dot{\mathbf{x}}^{\mathrm{o}}\bar{\mathbf{x}}^{\mathrm{o}}}\Phi^k_{\bar{\mathbf{x}}^{\mathrm{o}}\bar{\mathbf{x}}^{\mathrm{o}}} + \Xi^k_{\mathbf{v}\eta} \\ c^k_{\mathbf{v}\bar{\mathbf{x}}} &= \boldsymbol{\mu}^k_{\dot{\mathbf{x}}^{\mathrm{o}}} - \mathrm{S}^k_{\mathbf{v}\bar{\mathbf{x}}}\boldsymbol{\mu}^k_{\bar{\mathbf{x}}^{\mathrm{o}}} + \zeta^k_{\mathbf{v}\eta}\end{aligned}$$

with

$$\begin{aligned}\Xi^k_{\mathbf{v}\eta} &= \Sigma^k_{\dot{\mathbf{x}}^{\mathrm{o}}\eta}\Phi^k_{\eta\bar{\mathbf{x}}^{\mathrm{o}}} \\ \zeta^k_{\mathbf{v}\eta} &= \mathrm{S}^k_{\mathbf{v}\eta}(\eta - \boldsymbol{\mu}^k_{\eta})\end{aligned}$$

where $\mathrm{S}^k_{\mathbf{v}\eta} = \Sigma^k_{\dot{\mathbf{x}}^{\mathrm{o}}\bar{\mathbf{x}}^{\mathrm{o}}}\Phi^k_{\bar{\mathbf{x}}^{\mathrm{o}}\eta} + \Sigma^k_{\dot{\mathbf{x}}^{\mathrm{o}}\eta}\Phi^k_{\eta\eta}$ with the Φ^k_{ii} defined as

$$(\Sigma^k_{\bar{\mathbf{x}}})^{-1} = \begin{bmatrix} \Sigma^k_{\bar{\mathbf{x}}^{\mathrm{o}}\bar{\mathbf{x}}^{\mathrm{o}}} & \Sigma^k_{\bar{\mathbf{x}}^{\mathrm{o}}\eta} \\ \Sigma^k_{\eta\bar{\mathbf{x}}^{\mathrm{o}}} & \Sigma^k_{\eta\eta} \end{bmatrix}^{-1} \triangleq \begin{bmatrix} \Phi^k_{\bar{\mathbf{x}}^{\mathrm{o}}\bar{\mathbf{x}}^{\mathrm{o}}} & \Phi^k_{\bar{\mathbf{x}}^{\mathrm{o}}\eta} \\ \Phi^k_{\eta\bar{\mathbf{x}}^{\mathrm{o}}} & \Phi^k_{\eta\eta} \end{bmatrix} \tag{45}$$

The expressions of $\pi^k(\bar{\mathbf{x}})$ and $\frac{\partial \pi^k(\bar{\mathbf{x}})}{\partial \bar{\mathbf{x}}}$ in the Jacobian are computed as follows

$$h^k(\bar{\mathbf{x}}^{\mathrm{o}}) = \frac{\alpha^k p(\bar{\mathbf{x}}^{\mathrm{o}}|\boldsymbol{\mu}^k_{\bar{\mathbf{x}}}, \Sigma^k_{\bar{\mathbf{x}}})}{\sum_{k=1}^{K}\alpha^k p(\bar{\mathbf{x}}^{\mathrm{o}}|\boldsymbol{\mu}^k_{\bar{\mathbf{x}}}, \Sigma^k_{\bar{\mathbf{x}}})} \tag{46}$$

and

$$\frac{\partial h^k(\bar{\mathbf{x}}^{\mathrm{o}})}{\partial \bar{\mathbf{x}}^{\mathrm{o}}} = \frac{1}{D^2}(\frac{\partial N}{\partial \bar{\mathbf{x}}^{\mathrm{o}}}.D - N.\frac{\partial D}{\partial \bar{\mathbf{x}}^{\mathrm{o}}}) \tag{47}$$

where the terms $N(\bar{\mathbf{x}})$, $D(\bar{\mathbf{x}})$, $\frac{\partial N^k}{\partial \mathbf{x}}$, and $\frac{\partial D^k}{\partial \mathbf{x}}$ are respectively given by

$$N(\bar{\mathbf{x}}^{\mathrm{o}}) = \frac{\alpha^k}{(2\pi)^{\frac{N}{2}}|\Sigma^k|^{\frac{1}{2}}}e^{-\frac{1}{2}(\chi^k_{\mu})^{\top}(\Sigma^k)^{-1}(\chi^k_{\mu})} \tag{48}$$

$$D(\bar{\mathbf{x}}^{\mathrm{o}}) = \sum_{k=1}^{K}\frac{\alpha^k}{(2\pi)^{\frac{N}{2}}|\Sigma^k|^{\frac{1}{2}}}e^{-\frac{1}{2}(\chi^k_{\mu})^{\top}(\Sigma^k)^{-1}(\chi^k_{\mu})} \tag{49}$$

$$\frac{\partial N^k}{\partial \bar{\mathbf{x}}^o} = -\frac{\alpha^k}{(2\pi)^{\frac{N}{2}}\,|\Sigma^k|^{\frac{1}{2}}} e^{-\frac{1}{2}(\chi_\mu^k)^\top(\Sigma^k)^{-1}(\chi_\mu^k)}(\chi_\mu^k)^\top(\Sigma^k)^{-1}S_{\bar{x}} \tag{50}$$

$$\frac{\partial D^k}{\partial \bar{\mathbf{x}}^o} = -\frac{1}{(2\pi)^{\frac{N}{2}}}\sum_{k=1}^{K}\frac{\alpha^k}{|\Sigma^k|^{\frac{1}{2}}} e^{-\frac{1}{2}(\chi_\mu^k)^\top(\Sigma^k)^{-1}(\chi_\mu^k)}(\chi_\mu^k)^\top(\Sigma^k)^{-1}S_{\bar{x}} \tag{51}$$

where $\mathbf{x}_\mu^k$ is defined as $\chi_\mu^k = \begin{bmatrix} \bar{\mathbf{x}}^o - \boldsymbol{\mu}_{\bar{\mathbf{x}}^o}^k \\ \eta - \boldsymbol{\mu}_\eta^k \end{bmatrix}$ and $S_{\bar{x}} = \begin{bmatrix} I_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 0_{1\times1} \end{bmatrix}$

References

[1] P. Britt, How growing e-commerce demand is driving growth in mobile robotics, Robotics Business Review (2020).

[2] C. Mims, As e-commerce booms, robots pick up human slack (2020). URL https://www.wsj.com/articles/as-e-commerce-booms-robots-pick-up-human-slack-11596859205

[3] F. Raptopoulos, M. Koskinopoulou, M. Maniadakis, Robotic pick-and-toss facilitates urban waste sorting, in: 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), IEEE, 2020, pp. 1149–1154.

[4] M. Bombile, A. Billard, Dual-arm control for coordinated fast grabbing and tossing of an object: Proposing a new approach, IEEE Robotics Automation Magazine 29 (3) (2022) 127–138. doi:10.1109/MRA.2022.3177355.

[5] H. Frank, N. Wellerdick-Wojtasik, B. Hagebeuker, G. Novak, S. Mahlknecht, Throwing objects–a bio-inspired approach for the transportation of parts, in: 2006 IEEE International Conference on Robotics and Biomimetics, IEEE, 2006, pp. 91–96.

[6] B. Systems(BHS), Max-aiaqc-c recycling cobot (2019). URL https://www.youtube.com/watch?v=tEAr1w1Jxww

[7] G. Hassan, M. Gouttefarde, A. Chemori, P.-E. Hervé, M. El Rafei, C. Francis, D. Sallé, Time-optimal pick-and-throw s-curve trajectories for fast parallel robots, IEEE/ASME Transactions on Mechatronics.

[8] R. R. Burridge, A. A. Rizzi, D. E. Koditschek, Toward a dynamical pick and place, in: Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots, Vol. 2, IEEE, 1995, pp. 292–297.

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

[10] G. Bätz, A. Yaqub, H. Wu, K. Kühnlenz, D. Wollherr, M. Buss, Dynamic manipulation: Nonprehensile ball catching, in: 18th Mediterranean Conference on Control and Automation, MED'10, IEEE, 2010, pp. 365–370.

[11] S. Kim, A. Shukla, A. Billard, Catching objects in flight, IEEE Transactions on Robotics 30 (5) (2014) 1049–1065.

[12] M. M. Schill, M. Buss, Robust ballistic catching: A hybrid system stabilization problem, IEEE Transactions on Robotics 34 (6) (2018) 1502–1517.

[13] K. Dong, K. Pereida, F. Shkurti, A. P. Schoellig, Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 6718–6725.

[14] A. C. Satici, F. Ruggiero, V. Lippiello, B. Siciliano, A coordinate-free framework for robotic pizza tossing and catching, in: Robot Dynamic Manipulation, Springer, 2022, pp. 207–227.

[15] R. L. Anderson, A robot ping-pong player: experiment in real-time intelligent control, MIT press, 1988.

[16] L. Acosta, J. Rodrigo, J. A. Mendez, G. N. Marichal, M. Sigut, Ping-pong player prototype, IEEE robotics & automation magazine 10 (4) (2003) 44–52.

[17] T. Senoo, A. Namiki, M. Ishikawa, Ball control in high-speed batting motion using hybrid trajectory generator, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., IEEE, 2006, pp. 1762–1767.

[18] C. Lai, T. J. Tsay, Self-learning for a humanoid robotic ping-pong player, Advanced Robotics 25 (9-10) (2011) 1183–1208.

[19] K. Mülling, J. Kober, O. Kroemer, J. Peters, Learning to select and generalize striking movements in robot table tennis, The International Journal of Robotics Research 32 (3) (2013) 263–279.

[20] D. Serra, A. C. Satici, F. Ruggiero, V. Lippiello, B. Siciliano, An optimal trajectory planner for a robotic batting task: the table tennis example, in: International Conference on Informatics in Control, Automation and Robotics, Vol. 3, SCITEPRESS, 2016, pp. 90–101.

[21] Y.-B. Jia, M. Gardner, X. Mu, Batting an in-flight object to the target, The International Journal of Robotics Research 38 (4) (2019) 451–485.

[22] E. W. Aboaf, S. M. Drucker, C. G. Atkeson, Task-level robot learning: Juggling a tennis ball more accurately, in: Proceedings, 1989 International Conference on Robotics and Automation, IEEE, 1989, pp. 1290–1295.

[23] M. Buhler, D. E. Koditschek, P. J. Kindlmann, A family of robot control strategies for intermittent dynamical environments, IEEE Control Systems Magazine 10 (2) (1990) 16–22.

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

[25] K. M. Lynch, C. K. Black, Recurrence, controllability, and stabilization of juggling, IEEE Transactions on Robotics and Automation 17 (2) (2001) 113–124.

[26] A. Akbarimajd, M. N. Ahmadabadi, Manipulation by juggling of planar polygonal objects using two 3-dof manipulators, in: 2007 IEEE/ASME international conference on advanced intelligent mechatronics, IEEE, 2007, pp. 1–6.

[27] P. Reist, R. D'Andrea, Design and analysis of a blind juggling robot, IEEE Transactions on Robotics 28 (6) (2012) 1228–1243.

[28] D. Serra, F. Ruggiero, V. Lippiello, B. Siciliano, A nonlinear least squares approach for nonprehensile dual-hand robotic ball juggling, IFAC-PapersOnLine 50 (1) (2017) 11485–11490.

[29] K. L. Poggensee, A. H. Li, D. Sotsaikich, B. Zhang, P. Kotaru, M. Mueller, K. Sreenath, Ball juggling on the bipedal robot cassie, in: 2020 European Control Conference (ECC), IEEE, 2020, pp. 875–880.

[30] M. T. Mason, K. M. Lynch, Dynamic manipulation, in: Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93), Vol. 1, IEEE, 1993, pp. 152–159.

[31] B. Hove, J.-J. E. Slotine, Experiments in robotic catching, in: 1991 American Control Conference, IEEE, 1991, pp. 380–386.

[32] H. Frank, Design and simulation of a numerical controlled throwing device, in: 2008 Second Asia International Conference on Modelling & Simulation (AMS), IEEE, 2008, pp. 777–782.

[33] H. Frank, Determination of launching parameters for throwing objects in logistic processes with direct hits, in: 2008 IEEE International Conference on Emerging Technologies and Factory Automation, IEEE, 2008, pp. 58–61.

[34] T. Senoo, A. Namiki, M. Ishikawa, High-speed throwing motion based on kinetic chain approach, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2008, pp. 3206–3211.

[35] S. Ichinose, S. Katsumata, S. Nakaura, M. Sampei, Throwing motion control experiment utilizing 2-link arm passive joint, in: 2008 SICE Annual Conference, IEEE, 2008, pp. 3256–3261.

[36] H. Frank, A. Mittnacht, J. Scheiermann, Throwing of cylinder-shaped objects, in: 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, IEEE, 2009, pp. 59–64.

[37] W. Mori, J. Ueda, T. Ogasawara, A 1-dof dynamic pitching robot that independently controls velocity, angular velocity and direction of a ball, Advanced robotics 24 (5-6) (2010) 921–942.

[38] W. August, S. Waeldele, B. Hein, H. Woern, G. Wyeth, Accurate object throwing by an industrial robot manipulator, in: Proceedings of the Australasian Conference on Robotics and Automation 2010, ACRA 10, Brisbane, Australia, 2010, pp. 74–81.

[39] Y. Zhang, J. Luo, K. Hauser, Sampling-based motion planning with dynamic intermediate state objectives: Application to throwing, in:

2012 IEEE International Conference on Robotics and Automation, IEEE, 2012, pp. 2551–2556.

[40] A. Zeng, S. Song, J. Lee, A. Rodriguez, T. Funkhouser, Tossingbot: Learning to throw arbitrary objects with residual physics, IEEE Transactions on Robotics 36 (4) (2020) 1307–1319.

[41] A. Gallant, Optimisation de trajectoire pour l'augmentation des capacités des manipulateurs robotiques, Ph.D. thesis, Université Laval (2020).

[42] J. H. Kim, Y. Xiang, R. Bhatt, J. Yang, J. S. Arora, K. Abdel-Malek, Throwing motion generation of a biped human model, in: 2008 2nd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics, IEEE, 2008, pp. 587–592.

[43] A. C. Satici, F. Ruggiero, V. Lippiello, B. Siciliano, A coordinate-free framework for robotic pizza tossing and catching, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 3932–3939. doi:10.1109/ICRA.2016.7487582.

[44] Y. Liu, A. Nayak, A. Billard, A solution to adaptive mobile manipulator throwing, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 1625–1632. doi:10.1109/IROS47612.2022.9981231.

[45] F. Ruggiero, V. Lippiello, B. Siciliano, Nonprehensile dynamic manipulation: A survey, IEEE Robotics and Automation Letters 3 (3) (2018) 1711–1718.

[46] A. Sintov, A. Shapiro, A stochastic dynamic motion planning algorithm for object-throwing, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 2475–2480.

[47] M. Okada, A. Pekarovskiy, M. Buss, Robust trajectory design for object throwing based on sensitivity for model uncertainties, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 3089–3094.

[48] M. Asgari, A. Nikoobin, A variational approach to determination of maximum throw-able workspace of robotic manipulators in optimal ball pitching motion, Transactions of the Institute of Measurement and Control 43 (10) (2021) 2378–2391.

[49] W. B. Sturek, H. A. Dwyer, L. D. Kayser, C. J. Nietubicz, R. P. Reklis, K. O. Opalka, Computations of magnus effects for a yawed, spinning body of revolution, AIAA Journal 16 (7) (1978) 687–692.

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

[51] J. Kober, E. Oztop, J. Peters, Reinforcement learning to adjust robot movements to new situations, in: Twenty-Second International Joint Conference on Artificial Intelligence, 2011.

[52] S. Kim, S. Doncieux, Learning highly diverse robot throwing movements through quality diversity search, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2017, pp. 1177–1178.

[53] M. Mehrandezh, M. Sela, R. G. Fenton, B. Benhabib, Proportional navigation guidance in robot trajectory planning for intercepting moving objects, in: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), Vol. 1, IEEE, 1999, pp. 145–150.

[54] R. Sharma, J.-Y. Herve, P. Cucka, Dynamic robot manipulation using visual tracking, in: Proceedings 1992 IEEE International Conference on Robotics and Automation, IEEE Computer Society, 1992, pp. 1844–1845.

[55] M. D. Mikesell, R. J. Cipra, Development of a real time intelligent robotic tracking system, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 12860, American Society of Mechanical Engineers, 1994, pp. 213–222.

[56] S. W. Holland, L. Rossol, M. R. Ward, Consight-i: a vision-controlled robot system for transferring parts from belt conveyors, in: Computer Vision and Sensor-Based Robots, Springer, 1979, pp. 81–100.

[57] X. Mo, L. Liu, A robot system with vision touch and slide sensors for the grip onto a moving conveyor belt, in: Proc. 15th Int. Symp. Industrial Robots, 1985, pp. 129–136.

[58] P. K. Allen, A. Timcenko, B. Yoshimi, P. Michelman, Automated tracking and grasping of a moving object with a robotic hand-eye system, IEEE Transactions on Robotics and Automation 9 (2) (1993) 152–165.

[59] T. H. Park, B. H. Lee, An approach to robot motion analysis and planning for conveyor tracking, IEEE transactions on systems, man, and cybernetics 22 (2) (1992) 378–384.

[60] E. A. Croft, R. G. Fenton, B. Benhabib, Optimal rendezvous-point selection for robotic interception of moving objects, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 28 (2) (1998) 192–204.

[61] E. A. Croft, R. G. Fenton, B. Benhabib, Time-optimal interception of objects moving along predictable paths, in: Proceedings. IEEE International Symposium on Assembly and Task Planning, IEEE, 1995, pp. 419–425.

[62] I. Akinola, J. Xu, S. Song, P. K. Allen, Dynamic grasping with reachability and motion awareness, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 9422–9429.

[63] S. S. M. Salehian, N. Figueroa, A. Billard, Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty., in: Robotics: Science and Systems, 2016.

[64] S. S. Mirrazavi Salehian, N. B. Figueroa Fernandez, A. Billard, Dynamical system-based motion planning for multi-arm systems: reaching for moving objects, Tech. rep. (2017).

[65] S. S. Mirrazavi Salehian, N. Figueroa, A. Billard, A unified framework for coordinated multi-arm motion planning, The International Journal of Robotics Research 37 (10) (2018) 1205–1232.

[66] S. S. Mirrazavi Salehian, N. Figueroa, A. Billard, Dynamical system-based motion planning for multi-arm systems: Reaching for moving objects, In Proceedings of International Joint Conference on Artificial Intelligence 2017, Melbourne, Australia (2017).

[67] F. Caccavale, M. Uchiyama, Cooperative manipulation, in: Springer handbook of robotics, Springer, 2016, pp. 989–1006.

[68] H. G. Sung, Gaussian mixture regression and classification, Ph.D. thesis, Rice University (2004).

[69] T. Laetsch, J. Keller, The number of solutions of a nonlinear two point boundary value problem, Indiana University Mathematics Journal 20 (1) (1970) 1–13.

[70] P. T. Boggs, J. W. Tolle, Sequential quadratic programming, Acta numerica 4 (1995) 1–51.

[71] B. Colson, P. Marcotte, G. Savard, An overview of bilevel optimization, Annals of operations research 153 (1) (2007) 235–256.

[72] A. Savitzky, M. J. Golay, Smoothing and differentiation of data by simplified least squares procedures., Analytical chemistry 36 (8) (1964) 1627–1639.

[73] H. Ferreau, C. Kirches, A. Potschka, H. Bock, M. Diehl, qpOASES: A parametric active-set algorithm for quadratic programming, Mathematical Programming Computation 6 (4) (2014) 327–363.

[74] H. Eto, H. Nakamoto, T. Sonoura, J. Tanaka, A. Ogawa, Development of automated high-speed depalletizing system for complex stacking on roll box pallets, Journal of Advanced Mechanical Design, Systems and Manufacturing 13 (3). doi:10.1299/jamdsm.2019jamdsm0047.