IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Planning Impact-Driven Logistic Tasks

Ahmed Zermane[1], Niels Dehio[3], and Abderrahmane Kheddar[1,2]

***Abstract*—This letter proposes a decoupled two-level model-based planning strategy and control for a class of robotic tasks involving impact to be made with desired performance and constraints. The first part solves the problem of planning a robotic arm trajectory from a given state (position and velocity) to another desired one, enabling non-stop trajectory cycles. The second part is an impact-aware model-based plugin; it is specific to each task and links the desired task-space impact objective to a via-point in the joint-space. The two parts are then combined to achieve the entire task. Our approach is assessed with real-robot experiments demonstrating how this strategy can be used to perform tossing, grabbing, and boxing or any combination of them in sorting logistics-industry use-cases.**

I. Introduction

Impact-driven tasks are challenging to plan and to control due to non-smooth and fast contact dynamics. We are particularly interested in tasks found in the automated distribution and sorting logistics chains. Namely, fast and swift picking and tossing, grabbing, and boxing of objects of all sorts, see Fig. 1. The challenge is that the manipulated objects can undergo (open-loop) phases that are out of the robot control. For example, tossing an object requires accurate computation of its ballistic trajectory to meet landing specifications. Another example is manipulating (boxing) an object with a robot using impulses inducing an initial velocity (or, equivalently, an energy input) to bring it from an initial pose to a desired one: after the impulse, the object's motion is not 'controlled' by the robot. Our main contribution consists in devising a strategy in terms of planning and control of such tasks (defined by project partner Vanderlande) in two separate modules (Fig. 1):

○ **M1**: Plan for the robots a trajectory (with or without a load) to reach a target location (position and orientation) and velocity (linear and angular) from a given initial state (comprising both position and velocity). We allow the starting velocity to be non-zero for *the robot not need to stop at each cycle*. Our planning module accounts for hardware and task-space limits; see Sec. III.

○ **M2**: For each task (tossing in Sec. IV, grabbing in Sec. V, and boxing in Sec VI), we devise separate modular plugins representing specific inverse kinematics solvers for mapping desired impact objectives from task-space onto joint-space. The output of **M2** is the input of **M1**, i.e., the planner ensures bypassing the via-point with desired position and velocity.

With respect to related work (discussed in Sec. II), our novelty lies in defining task coordinates as a via-point, which are integrated into a time-constrained path. The latter is optimized for a generated samples' map. Throughout the entire letter, we assume rigid objects with known geometry and well-estimated inertial parameters [1].

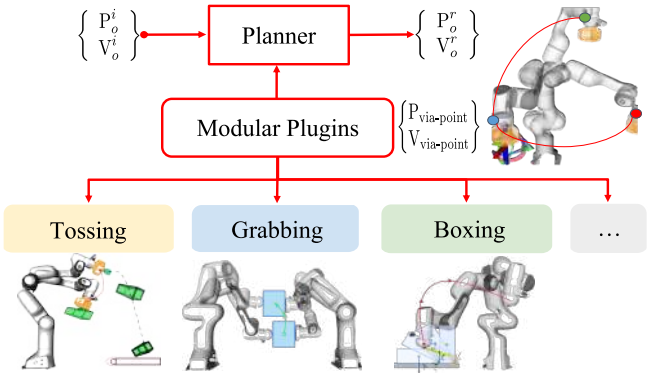

Fig. 1: Proposed motion generator for scenarios with impact.

We assessed our approach in several real-robot experiments, finally demonstrating a complex industry-driven task that could be customized to any sequence order, see Sec. VII.

II. A Brief Background

• *Planning and motion generation* are well-explored topics in robotics. A multi-mode trajectory optimization proposed in [2] combines hybrid control and dynamics to embrace impact for stopping moving objects in impact-aware manipulation tasks. In [3], a multi-physics formulation enables one to interact with objects at non-zero relative velocity to achieve desired motions. Alternatively, by relying on trajectory optimization under contact constraints, the robot can push objects [4]. Whereas [5] relies on learning from interactive demonstrations to execute picking at non-zero relative velocities. In [6], an impact-aware QP controller is formulated to be resilient to 3D frictional impacts in the task-space by defining new impact-related constraints and reformulating the usual ones to turn them robust against abrupt changes of the states at impacts. Other related works extend to impact for different robots and tasks within [7], [8] using dynamical systems and machine learning-based approaches.

• *Tossing* can be seen as an extension of the robot's workspace or for speeding pick-and-place task that do not require high placement precisions. Existing robotic tossing methods are found in model-based and data-driven approaches. The most outstanding work is reported in [9]. It is a machine learning-based system combined with residual physics designed to enable rapid picks and throws of arbitrary objects into selected boxes. In [10], tossing is formulated using reinforcement learning (RL), enabling generalization on various objects and fast reaction to dynamic environments. Combining

This work was supported by the Research Project I.AM. through the European Union H2020 program (GA 871899)

[1] A. Zermane and A. Kheddar are with the CNRS-University of Montpellier, LIRMM, Montpellier, France.

[2] A. Kheddar is also with the CNRS-AIST Joint Robotics Laboratory, IRL3218, Tsukuba, Japan.

[3] N. Dehio is with KUKA Deutschland GmbH, Augsburg, Germany.

decision transformers (DT) and RL enables tossing objects from handful experiments and far-from-reality trials [11]. Earlier work in [12] handled basic shapes (balls) grasping and tossing using a deep neural network architecture to perform grasping. Throwing a ball by a kinetic chain approach based on swinging motion and constraining its motion to the release timing and release direction is proposed in [13]. A dedicated gripper for grasping and throwing is proposed in [14], containing a latching mechanism that ensures both high accelerations for throwing by instantly discharging the stored elastic potential and low for placing.

• *Dual-am grabbing* enables higher payload handling and larger object grasp, e.g., [15]. In a previous work [16], this scenario was achieved using an impact-aware model preview control with learned dynamics of soft deformable pads mounted on the end-effectors. In [8], Modulated Dynamical Systems (MDS) generate motion for a dual-arm system to grab and release an object reactively swiftly. In [17], a time-invariant reference spreading controller is formulated to manipulate the object swiftly. Impact grabbing is studied more fundamentally and theoretically in [18], [19], where Linear Complementarity Systems (LCS) formulation is employed to synthesize controllers and study systems' stability in rich contact manipulation problems, and dual-grabbing as an example. In [20], grabbing is treated as a Stochastic Discrete-time LCS (SDLCS) solved by a chance-constrained based optimization.

• *Boxing*, i.e., pushing and tilting objects can be treated as impact-contacts in motion control and planning [21]. For example, [22] used a suction cup end-effector to tilt an object by developing a framework that withstands deformation modeling errors. In [23], a robust pivoting formalism accounts for object inertia parameters. It is developed based on contact implicit bilevel optimization and enhanced by a tactile closed-loop control to deal with initializing issues and enable failure recovery. Impact-based interactions based on contact-implicit trajectory optimization (TO) are proposed in [4], assuming a hard contact model and imposing unilateral constraints as complementarity conditions. On the other hand [24] proposed a TO based on a variable smooth contact model and successive convexification to push objects to a desired pose, whereas for the same problem [7] used a learned statistical dynamical system to generate a non-zero relative contact velocity.

III. Impact-driven trajectory planning

Module **M1** builds on a sampling-based algorithm Bi-RTT [25]. It uses a steering function to plan minimum-time trajectories in the robot's configuration space subject to a via-point constraints in the task-space. The latter embeds explicitly impact postures and enforces back-to-safety configurations. Unlike any of the related works (e.g., [26]), this creates profile motions with way-points on position, velocity and acceleration levels. This Module relies on acceleration-limited trajectory generation where Double Integrator Minimum Time (DIMT) is the neighborhood criterion [27]. It allows having non-zero start and end velocities of the planned trajectories. If any jerk constraints exist, it can be used within the steering function.

To reach a target position/velocity (pose) state from a given initial one, Bi-RTT grows iteratively two trees: (i) forward in time starting from the initial state, and (ii) backward in time starting from the final state, see Fig. 2, blue-dotted area on the planning flowchart in Fig. 3. The sample-map segments with limited duration prevent numerical integration errors. Hence, there is no need to define an upper limit time [26] for the whole trajectory. For the acceleration-limited trajectory generation, each state Q is represented by the stacked vectors of joint states $Q = [q, \dot{q}]$, which extend to accelerations $Q = [q, \dot{q}, \ddot{q}]$ in the jerk-limited case.

The steering method, described in Algorithm 1 (lines 12 and 18) and detailed in Algorithm 2, is either the DIMT or the jerk-based one, see details later. This steering method relies on determining the closest neighbor using a distance function.

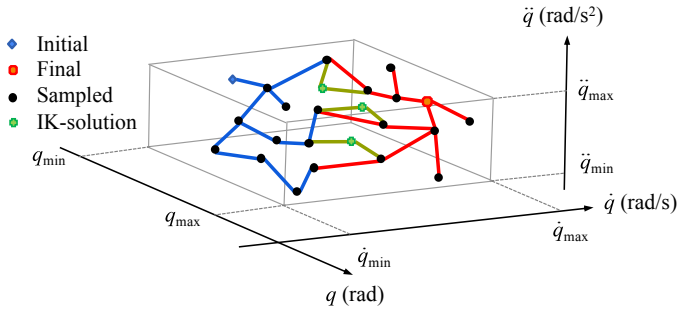

Fig. 2: Two grown trees (blue and red edges). Sampled states are chosen within the bounds for each DoF, defined by velocity, position, and acceleration limits in the jerk-limited case. Via-point constraints for a task-space pose and velocity are converted into joint-space states using inverse kinematics.

Algorithm 1 Extended BiRRT Algorithm

1: **procedure** BiRRT($q_{\max}, q_{\min}, \dot{q}_{\max}, \ddot{q}_{\max}, P_{\text{init}}, P_{\text{final}}$, N)
2: $\text{cost}_i \leftarrow 0$, $\text{cost}_f \leftarrow 0$
3: FinalPath $\leftarrow [\,]$; FinalCost $\leftarrow \infty$;
4: $V1 \leftarrow [P_{\text{init}}, Q_{\text{init}}, \text{cost}_i]$;
5: $V2 \leftarrow [P_{\text{final}}, Q_{\text{final}}, \text{cost}_f]$;
6: $E1 \leftarrow [\,]$; $E2 \leftarrow [\,]$;
7: d $\leftarrow$ true; % growing direction indicator
8: counter $\leftarrow 0$;
9: **while** counter $\leq$ N **do**
10: $Q_{\text{r}} \leftarrow$ SampleInJointSpace;
11: $P_{\text{r}} \leftarrow$ ForwardKinematics(Q_{r}), CollisionCheck(Q_{r});
12: $T_1, S_1, \text{cost}_{r_i}, \text{JointLimits} \leftarrow$ Steering$(Q_r, V1, \text{d})$;
13: **if** ¬JointLimits **then**
14: $Q_i, P_i \leftarrow$ SegmentsGeneration(T_1, S_1);
15: $E_{\text{temp}} \leftarrow$ BuildEdges(Q_i, P_i, S_1);
16: $V1 \leftarrow V1 \cup [P_r, Q_r, \text{cost}_r] \cup [P_i, Q_i, \text{cost}_i]$;
17: $E1 \leftarrow E1 \cup E_{\text{temp}}$;
18: $T_2, S_2, \text{cost}_{r_f}, \text{JointLimits} \leftarrow$ steering$(Q_r, V2, \bar{\text{d}})$;
19: **if** ¬JointLimits **then**
20: $Q_i, P_i \leftarrow$ SegementsGeneration(T_2, S_2);
21: $E_{\text{temp}} \leftarrow$ BuildEdges(Q_i, P_i, S_2) ;
22: $V2 \leftarrow V2 \cup [P_r, Q_r, \text{cost}_r] \cup [P_i, Q_i, \text{cost}_i]$;
23: $E2 \leftarrow E2 \cup E_{\text{temp}}$;
24: $\text{Cost}_{\text{Tmp}} = \text{cost}_{r_i} + \text{cost}_{r_f}$;
25: **if** $\text{Cost}_{\text{Tmp}} <$ FinalCost **then**
26: FinalCost $\leftarrow \text{Cost}_{\text{Tmp}}$;
27: FinalPath $\leftarrow$ Extractpath$(V_1, E_1, V_2, E_2, \text{d})$;
28: **end if**
29: **end if**
30: **end if**
31: Swap$([V_1, E_1], [V_2, E_2])$;
32: d $\leftarrow \bar{\text{d}}$;
33: counter $\leftarrow$ counter $+ 1$;
34: **end while**
35: **return** FinalPath, FinalCost;
36: **end procedure**

Different metrics exist to define distance functions. We use the go-to time between two states [27], [28]. Despite not being a metric because of its asymmetry, it is proven reliable, parameter-free, and accounts for both position and velocity. As shown in Algorithm 1, a connection between two trees can be made if the randomly generated state Q_r is successfully added to both trees (lines 16,17–22, 23).

Each trajectory task can be generated by skipping path selection at the first phase of sample-map building for N iterations. For the same task, each via-point has multiple joint-space configurations competing to generate the best path, red-dotted area in the flowchart see Fig. 3. We start evaluating the desired number of IK solutions (loop-line 9) by considering it as the new random states (line 10). Finally, Algorithm 1 extracts the path that fulfills all the initial and final conditions under limits and collision constraints (line 27). It has a time complexity of $\mathcal{O}(2n)$, where $n = \max(N(V_1, V_2))$. To

Algorithm 2 Steering(Q_r, V, d)

1: $Q_{\text{near}}, T, S, \text{cost}_r \leftarrow$ NearestNeighbor(Q_r, V, d);
2: JointLimits $\leftarrow$ PositionLimitsViolation(S);
3: **return** T, S, cost$_r$, JointLimits;

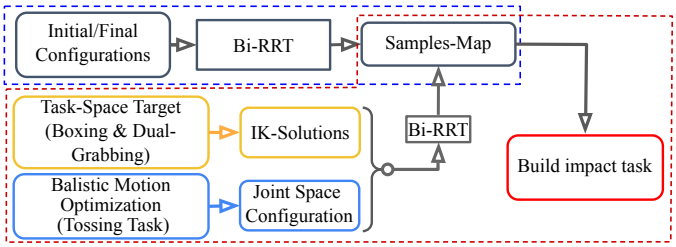

Fig. 3: Planning framework flowchart.

introduce both steering functions, we define:

- $(q_\star, \dot{q}_\star)$ position and velocity at $\star$;
- $(\ddot{q}_{\star_1}, \ddot{q}_{\star_2})$ acceleration values of a bang-bang profile;
- $(\dot{q}_{\max}, \ddot{q}_{\max}, \dddot{q}_{\max})$ velocity, acceleration, jerk bounds.

In DIMT steering function, there are four possible extremal motions for velocity and acceleration bounds, see Fig. 4: the parabolas P^+ and P^- accelerating at $\ddot{q}_{\max}$ and at $-\ddot{q}_{\max}$, respectively; and the lines L^+ and L^- traveling with zero acceleration at max velocity $\dot{q}_{\max}$ and at min velocity $-\dot{q}_{\max}$, respectively. The sign of the first acceleration segment is determined programmatically [27], using

$$\sigma = \text{sgn}(q_{i+1} - q_i - \delta q_{\text{acc}}), \tag{1}$$

where δq_{acc} is the distance crossed while accelerating at $\ddot{q}_{\max}$, which is defined as:

$$\delta q_{\text{acc}} = \frac{1}{2}(\dot{q}_i + \dot{q}_{i+1})\frac{|\dot{q}_i - \dot{q}_{i+1}|}{\ddot{q}_{\max}}. \tag{2}$$

thus we have $\ddot{q}_{i_1} = -\ddot{q}_{i_2} = \sigma\ddot{q}_{\max}$ and $\dot{q}_{\text{limit}} = \sigma\dot{q}_{\max}$.

The motion profile time parameters are defined as follows:

1) determine t_1 by solving the quadratic equation,

$$\ddot{q}_{i_1} t_1^2 + 2\dot{q}_i t_1 + \frac{\dot{q}_{i+1}^2 - \dot{q}_i^2}{2\ddot{q}_{i_2}} - (q_{i+1} - q_i) = 0; \tag{3}$$

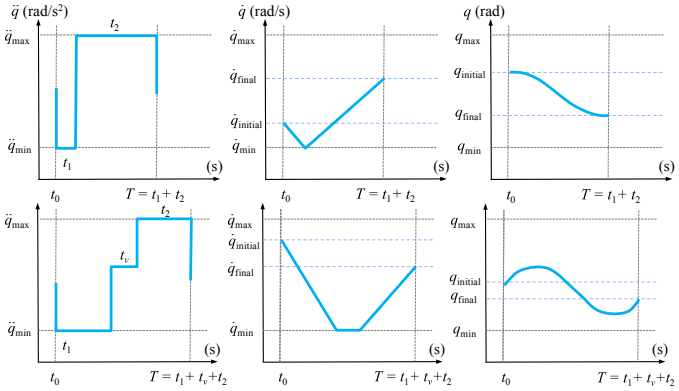

Fig. 4: Position-velocity-Acceleration profiles, Two segments profile, and Three segments profile.

2) check for velocity bounds violation, if $(\dot{q}_{t_1} < \dot{q}_{\min})$ or $(\dot{q}_{t_1} > \dot{q}_{\max})$, then it is a trapezoidal constant-velocity profile segment with duration t_v:

$$t_1 = \frac{\dot{q}_{\text{limit}} - \dot{q}_i}{\ddot{q}_{i_1}}, \tag{4}$$

$$t_v = \frac{\dot{q}_{i+1}^2 + \dot{q}_i^2 - 2\dot{q}_{\text{limit}}^2}{2\dot{q}_{\text{limit}}\ddot{q}_{i_1}} + \frac{q_{i+1} - q_i}{\dot{q}_{\text{limit}}}, \tag{5}$$

$$t_2 = \frac{\dot{q}_i - \dot{q}_{\text{limit}}}{\ddot{q}_{i_2}}. \tag{6}$$

else:

$$t_2 = \frac{\dot{q}_{i+1} - \dot{q}_i}{\ddot{q}_{i_2}} + t_1. \tag{7}$$

After obtaining the extremal profile for each DoF with duration T_i ($i = \overline{1,n}$ and n is DoFs number), and checking for its blocked time interval, the minimum-acceleration trajectory [27], [29] is then applied to synchronize T for all the DoFs by calculating new profiles parameters as follows:

$$T^2\ddot{q}_{i_1} + (2T(\dot{q}_{i+1} + \dot{q}_i) - 4(q_{i+1} - q_i))\ddot{q}_{i_1} - (\dot{q}_{i+1} - \dot{q}_i)^2 = 0, \tag{8}$$

For the Jerk-limited steering method. We use the algorithm proposed in [26] (available online). The resultant motion profile is an S-curve time scaling that consists of a maximum of seven segments.

IV. Tossing

Our approach differs from all previously cited work [9], [10], [11], [13], [14] in explicitly defining constraints reflecting any robot's ability to toss within a computed reach workspace, and more importantly, in integrating explicit constraints on objects' impact specifications. That is to say, we account for objects landing impact location, relative speed, or impulse. Then, it is plugged into the developed planner previously with the physical modeling of the tossing ballistic.

Prior to planning and control, we define the toss release reachable workspace (in both position and velocity). The Algorithm 3 describes the entire process, and Fig. 5 illustrates the outcome in a schematic way.

First, we define the robot's reachable workspace with task velocity bounds (line 1 Algorithm 3). Using the latter, we estimate the tossing workspace by solving simple ballistic motion discarding post-impact effects (e.g., bouncing, sliding...) for each height layer (z). For each z we sample a launching

Algorithm 3 Tossing-Workspace (Robot, $Z_{\max}$, $Z_{\min}$, step)

1: $\text{RW}_{\text{space}} \leftarrow$ ReachableWorkspace(Robot, V, d);
2: $z = Z_{\min}$;
3: **while** $z \leq Z_{\max}$ **do**
4: $\quad \text{TW}_{\text{space}} \leftarrow$ TossingWorkspace(Robot, RW_{space}, z);
5: $\quad z = z + \text{step}$;
6: **end while**
7: **return** TW_{space};

position with different velocities (using an iterative algorithm based on Monte Carlo). We solve for each pose (position and velocity) (line 4 Algorithm 3) the landing time t_l such that $\text{P}_{t_l} = z_l$ from eq. (9) (see later).

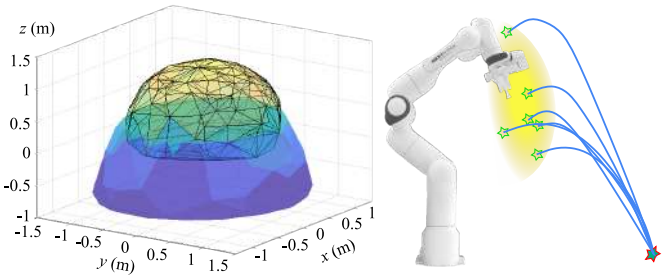

Fig. 5: example of a reachable workspace (sphere-like volume with black edges) and tossing workspace, for $z_{\max} = 0.5$ m and $z_{\min} = -0.7$ m (left). Tossing set for a given target (right).

We formulate the tossing ballistics as an optimization problem. The release state (a via-point in our planning) is determined by optimizing the ballistic motion, considering the robot's capabilities to generate a feasible one. Other ballistic related phenomena could be modeled, e.g., aerodynamics, bouncing, gripper dynamics... but would lead to a complex non-linear problem. Our primary goal is to solve tossing from industry-driven perspectives. Bouncing is challenging to predict. Therefore, one of the objectives we set is having low impact relative-velocity of the thrown objects when the precision of the landing spots prevails.

We define $P_d \in \mathbb{R}^3$, $\mathcal{Q}_d \in \mathbb{R}^4$ as the desired position and quaternion orientation of the object and P_{t_l}, $\mathcal{Q}_{t_l}$ the ones at time-step t_l of the ballistic motion defined by:

$$P_{t_l} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} v_x t_l + x_o \\ v_y t_l + y_o \\ -g t_l^2 + v_z t_l + z_o \end{bmatrix}, \tag{9}$$

and

$$\mathcal{Q}_{t_l} = \int_0^{t_l} \frac{1}{2} \Omega(\omega) \mathcal{Q} \, dt \text{ with } \dot{\omega} = \mathcal{I}^{-1}\tau - \mathcal{I}^{-1}(\omega \times \mathcal{I}\omega) \tag{10}$$

where $\Omega(\omega) \in \mathbb{R}^{4\times4}$ is a skew-matrix; g is the gravitational acceleration; $[x_o, y_o, z_o] \in \mathbb{R}^3$ and $\mathcal{Q}_o \in \mathbb{R}^4$ are position and orientation coordinates of the object's CoM at the moment of release; $v_{\text{ee}} = [v_x, v_y, v_z]$, $\omega_{\text{ee}} = [\omega_{x_o}, \omega_{y_o}, \omega_{z_o}]$ are the linear and angular velocities of the end-effector from which that of the picked object is determined as initial conditions for eq. (9) and eq. (10), respectively; $\mathcal{I} \in \mathbb{R}^{3\times3}$ is the object inertia matrix; and $\tau \in \mathbb{R}^3$ is the external torques for a free ballistic motion (assumed nil, $\tau = 0_{3\times1}$).

We denote the translation error $e_{\text{P}} = P_{t_l} - P_d$; used as design, evaluation criterion in [9], [11] respectively; and express the orientation error e_{O} as the imaginary part $\Im$ of the Hamilton product $\odot$ between desired and landing rotations:

$$e_{\text{O}} = \Im[\mathcal{Q}_d \odot \mathcal{Q}_{t_l}^{-1}] = \nu_{t_l} \cdot \epsilon_d - \nu_d \cdot \epsilon_{f_l} + \epsilon_d \times \epsilon_{t_f} \tag{11}$$

The problem of the IK for tossing accounting for impact constraints (location, orientation, and velocity) and robot limits, is formulated as a weighted optimization as follows:

$$\begin{aligned} \min_{q,\dot{q}} \; & w_1 \left\| e_{\text{P}} \right\|^2 + w_2 \left\| e_{\text{O}} \right\|^2 + w_3 \left\| v_{\text{desired}} - v_{\text{impact}} \right\|^2 \\ \text{s.t: } & \dot{q}_{\min} \leq \dot{q} \leq \dot{q}_{\max} \text{ and } q_{\min} \leq q \leq q_{\max} \\ & [v_{\text{ee}}, \omega_{\text{ee}}] = J(q)\dot{q} \\ & \left\| v_{\text{ee}} \right\| \leq v_{\max} \text{ and } \left\| \omega_{\text{ee}} \right\| \leq \omega_{\max} \end{aligned} \tag{12}$$

where decision variables $(q, \dot{q})$ are robot's joint position and velocity; J is the Jacobian matrix; and w_i the associated weight to each term. The object's impact velocity v_{impact} is considered in the ballistic optimization. The generated tossing pose, chosen from a constrained subset in Fig. 5, would induce a landing velocity as close as possible to v_{desired} at the landing time t_l, i.e., by the end of the ballistic motion trajectory. The solution of the optimization problem defined in eq. 12 is then considered a via-point state for the trajectory planner.

The target $\{P_d, \mathcal{Q}_d, v_{\text{impact}}\}$ belongs to the tossing workspace; else the solution found will be the closest to it.

V. Impact Grabbing

Swift grabbing of objects with impact using a dual-arm robotic system requires (i) knowing where to contact the object for each arm (e.g., could be driven by having the object balanced during lifting), (ii) specifying relative velocity at both impacts, and, more importantly, (iii) synchronizing both arms to impacts at the same time. To fulfill these requirements, we query the planner to find feasible collision-free configuration paths for both robots (which can be extended to multi-robots). Indeed, our proposed planner can straightforwardly handle dual manipulation and many configurations can be put into cascade to form a fully defined series of tasks, i.e., dual-grab, dual-toss, or dual-place, dual-re-grab, etc.

Synchronizing both robots can be done at the planning level; time constraints are added to the planned path for each robot to match the arrival time for both of them. It can also be made at the control level since we use the same tracking strategy in Sec. VII-A, introducing the posture task of the fastest robot with a delay corresponding to the time difference between the planned paths.

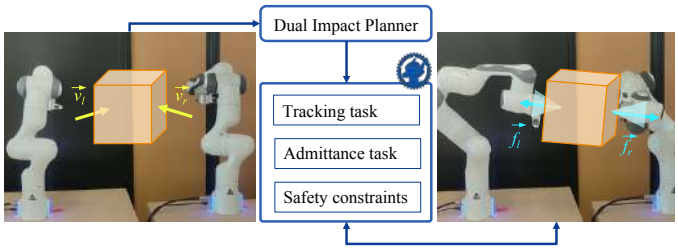

Fig. 6: Dual grabbing scenario: (left) define impact-grab velocity, (right) execute the planned trajectory.

Once the impact with the object occurs, the robot shall instantly switch the control objectives in terms of forces and

torques toward a stable grasp and swift motion control. Our multi-robot QP task-space control formulation enables parallel admittance tasks to monitor and control the contact state between the dual-arm system and the object, see Fig. 6. In order to account for impact uncertainties due to one robot contacting the object before the other one, we mitigate the impact velocity bounds (using momentum conservation) in the impact-aware QP controller w.r.t to the possible motion of the object in the direction of the opposite contact, which results in lowering impacting velocity bounds for both robots.

VI. Impact Boxing

Boxing involves applying an impulse on an object to induce an initial velocity that brings it to a desired state. Concretely, we examine how to rotate a resting box using a single impulse provided by a robotic arm, which is mainly different from what is described in related work, e.g., [4], [7], [21], [22], [23], [24]. We use an explicit model-based plugin that integrates our planning strategy by accounting for all forces acting on the box; see Fig. 7.

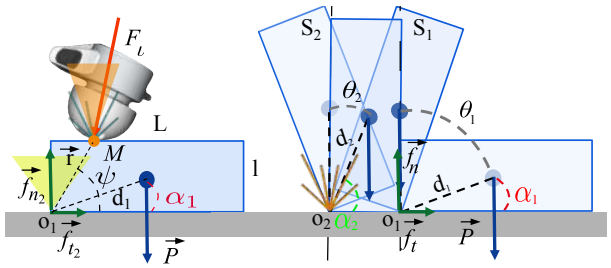

Fig. 7: On the left side, the force applied on the box exactly before starting to tilt. On the right one, the box and applied forces during tilting phases S_1 and S_2.

At impact time T, utilizing Newton's Laws circumvents the complexity linked to state discontinuities [30]. Thus, there is a need to use second-order Measure Differential Equation (MDE) (assuming no sliding):

$$\mathcal{I}_{\mathrm{yy}_{/o_1}}\dot{\omega} = \vec{\tau}_\iota \delta_T \vec{j} + \vec{\tau_P}\vec{j} \tag{13}$$

$$\text{and} \quad \vec{\tau}_\iota = \overrightarrow{OM}\times\vec{F}_\iota = [0, -z_M f_{\iota_x} + x_M f_{\iota_z}, 0]^T$$

$$\vec{\tau}_P = \overrightarrow{OG} \times \vec{P} = [0, mgx_G, 0]^T = [0, mgd_1\cos(\alpha_1), 0]$$

$$\overrightarrow{OM} = \left[l\cot(\psi), 0, l\right]^T, \overrightarrow{OG} = \left[d_1\cos(\alpha_1), 0, d_1\sin(\alpha_1)\right]$$

are the coordinates of the impact point M and the object's CoM; $\vec{F}_\iota$ is the impact force generated by the robot end-effector; $\vec{P}$ is the gravity force acting on the box; and δ_T is the Dirac measure at impact. $[\mathcal{I}, \dot{\omega}]$ is described in Sec. IV. We enforce non-sliding impact with

$$|\vec{f_t}| \leq \mu_0 f_n,\ |\vec{\tau}_\iota| \geq D_y f_n,\ \text{and} f_{\iota_z} > f_{\iota_x} > 0 \tag{14}$$

where μ_0 and D_y are friction coefficients. Post-impact dynamics can be expressed as

$$\mathcal{I}_{\mathrm{yy}_{/o_1}}\dot{\omega} = \vec{\tau}_P(\theta)\vec{j} = mgd_1\cos(\theta)\vec{j} \tag{15}$$

with $\omega_0 = \omega_0^+ \neq 0$ resulting from the impact, and $\theta^+ = 0$.

the post-impact velocity at M is written as:

$$\vec{\nu}_M^+ = \vec{\omega}^+ \times \overrightarrow{OM} = \omega^+[l\cot(\psi), 0, l]^T \tag{16}$$

we get ω^+ by solving the energy based equation eq. 17:

$$\frac{1}{2}\mathcal{I}_{\mathrm{yy}_{/o_1}}{\omega^+}^2 = mgd_1(1-\sin(\alpha_1)) \tag{17}$$

$$\omega^+ = \sqrt{\frac{2mgd_1}{\mathcal{I}_{\mathrm{yy}_{/o_1}}}(1-\sin(\alpha_1))} \tag{18}$$

we can then estimate the impact point and velocity from eq. 16 to plug it into the impact planner.

However there are situations where, even in perfect physics, it is not possible to reach a settled pose. The right amount of velocity to bring the object to reach its intended pose induces a rotational trajectory from the initial resting pose to the singular point at near-to-nil angular velocity (S_1 in Fig. 7). Having the right minimal energy making the rotating object pass the singular pose S_1, the remaining second phase trajectory is a free-falling one; which induces another impact at S_2 resulting in an energy gain. Depending on the impact physics, the geometry of the object and the landing ground, the desired pose can be reached after a few bounces or the object would fall on the other side.

In order to achieve boxing, some conditions must be satisfied, first

$$\tau_\iota > \tau_P \Leftrightarrow l\Big(-f_{\iota_x} + \cot(\psi)f_{\iota_z}\Big) > mgd_1\cos(\alpha_1) \tag{19}$$

From eqs. (14) and (19), we constraint x_M

$$f_{\iota_x} < \cot(\psi)f_{\iota_z} \Leftrightarrow \psi > \frac{\pi}{4} \Leftrightarrow x_m < l \tag{20}$$

Then from eq. 19, the influence of the box thickness is also clear, i.e., the thinner the box to be flipped, the more impact force is needed. At some point it will never be possible to flip it with any amount of normal force

$$l \to 0 \Rightarrow x_m \to 0 \Rightarrow \psi \to \frac{\pi}{2}$$

The CoM position to the rotation axis can also make this task difficult to achieve. By analyzing energy at singularity positions S_1 and S_2 (see Fig. 7), we seet two main conditions:
(i) $\theta_2 + \alpha_2 \leq \frac{\pi}{2}$, and
(ii) $E_2 = mgd_2\cos(\theta_2+\alpha_2) \leq \epsilon E_1 = \epsilon mgd_1\cos(\alpha_1)$, where ϵ is the coefficient of restitution.

Assuming that, at S_1, $\omega_{S_1} \geq 0$; at S_2, $\omega_{S_2} = 0$; and $\epsilon = 1$,

$$mgd_1 = mgd_2\cos(\theta_2+\alpha_2) \tag{21}$$

since $\cos(\theta_2+\alpha_2) \in [0,1]$ we have

$$d_1 \leq d_2 \tag{22}$$

which means $\frac{\pi}{2} - \alpha_2 \leq \alpha_1$.

VII. Experiments

We conducted extensive experiments using 7 DOF Panda robots to assess our approach. We also used the open-source QP controller `mc_rtc` that runs at 1 kHz using the necessary dependencies [16] `mc_franka` and `mc_panda`. We report some results and data obtained and provide an accompanying video Check video: https://youtu.be/A8N0C3mE3TM as supplementation material.

A. Trajectory Tracking

Common to all tasks, the planner (Sec. VII-A) does not include the robot's dynamics. Hence, we cannot track the

planned trajectory obtained from Algorithm 1 in open-loop; therefore, we impose a posture task objective in the QP controller [31]. Let the triplet $\{q_{\text{ref}}, \dot{q}_{\text{ref}}, \ddot{q}_{\text{ref}}\}_t$ be position, velocity, and acceleration references at t time-step within the generated trajectory, and $\{q_{\text{cur}}, \dot{q}_{\text{cur}}\}_t$ the current position and velocity sensor readings. The desired task-space acceleration is formulated as a classical linear spring-damper feedback law $\ddot{q}_{\text{d}} = \ddot{q}_{\text{ref}} - K_s(q_{\text{cur}} - q_{\text{ref}}) - K_d(\dot{q}_{\text{cur}} - \dot{q}_{\text{ref}})$, with diagonal positive-definite matrices K_s, K_d that represent the stiffness and damping gains. The QP is formulated as:

$$\begin{aligned} &\min_{\ddot{q}\in\mathbb{R}^n} \|\ddot{q} - \ddot{q}_{\text{d}}\| \\ &\text{s.t: } \mathcal{C}(\dot{q}, q)\ddot{q} + d \leq 0 \end{aligned} \tag{23}$$

where $\mathcal{C}(\dot{q}, q)$ is the constraints vector related to the robot's limits (position, velocity, acceleration, torque, jerk); the jerk is included in the constraints by deriving generated acceleration; d is the bounds vector.

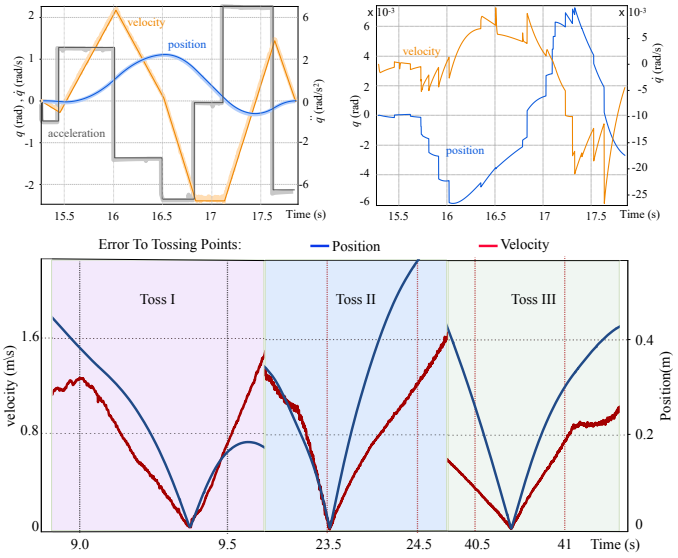

Fig. 8: Position, velocity, and acceleration tracking for joint 1: desired (dark) and real (shaded), (Top-left). Position and velocity tracking errors (Top-right). Measured Position and Velocity errors to the TS defined Tossing Points (Bottom).

Figure 8 shows a sample trajectory-tracking result, revealing a limited position and velocity error: $|e_{\text{position}}| \leq 6.10^{-3}$ rad, $|e_{\text{velocity}}| \leq 25.10^{-3}$ rad/s for all joints, and the corresponding task-sapce velocity and position errors to the via-point. Which is enough for the intended tasks and their application in a logistics context under the assumption made for each task.

B. Tossing

First, the robot's reachable and tossing workspaces are constructed. Then assuming that we know precisely the object inertia parameters, we toss to several chosen empty containers as landing targets (put at different heights) within the reachable tossing space. These tossing experiments are achieved either with a suction cup or a panda gripper as picking tools. For the suction cup case, the object is assumed to be rigidly attached to the suction cup for the planning, and the suction cup – object contact rupture delay is determined statistically by analyzing pressure drop down and actual release command at around ≈ 200 ms in preliminary experiments.

This is to show that we are able to plan various objects tossing on different locations. We also specified different landing velocities (equivalently different landing impacts) w.r.t to the surface normal for the same landing target of the same object. To highlight differences between ballistic motions, we compare the release configurations and measure the object impacts. For this purpose, we instrumented the target floor with ATI-force sensors; see result data in Fig. 9.

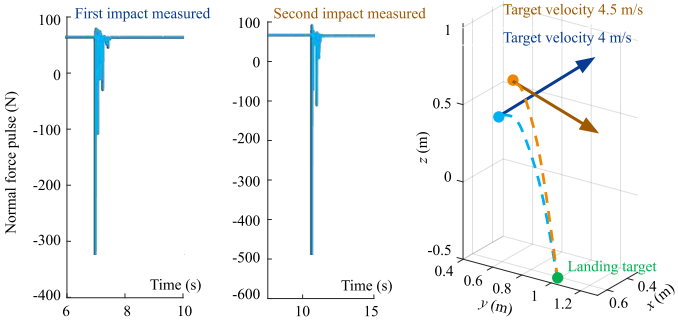

Fig. 9: Same landing target with different impact velocities.

The obtained results demonstrate that we are able to toss objects (providing their inertia parameters are known) precisely to any reachable location surrounding the robot. Yet, contrarily to existing tossing approaches, see Sec. II (tossing) and Sec. IV (first paragraph), we are able to put specifications on the landing pose and relative velocity (or impact). We also didn't have to learn from any real or simulation data/trials or put constraints on the tossing configuration (shortcut used in the learning methods) that limits the performance of robots and reduces the solutions set. For now, it is still difficult to precisely predict object pose and location after impact, as such behaviors are yet extremely difficult to predict.

C. Grabbing

We have also replicated intensive grabbing scenarios, namely those already achieved in [16]. The only added-value with respect to our latter work is that now, we are able to automatically generate grabbing trajectories that were made ad-hoc by the user in [16], see results on grabbing in Sec. VII-E where all skills are combined. Compared to [8], we do not need to have an intermediary MDS learning phase, as we can simply define task-space targets. Moreover, in [8], there is no guarantee on constraint fulfillment; also, it is not possible to specify landing constraints other than object location.

D. Boxing

We tested boxing (tilting fully different boxed-objects) utilizing a 3D-printed rigid end-effector. Boxes are used in our experiments because a large quantity of parcels found in e-commerce sorting logistics are cardboard boxes; and we used boxes obtained from real deliveries.

Figure 10 shows the execution of the planned trajectory. The desired impact point M is reached with the corresponding

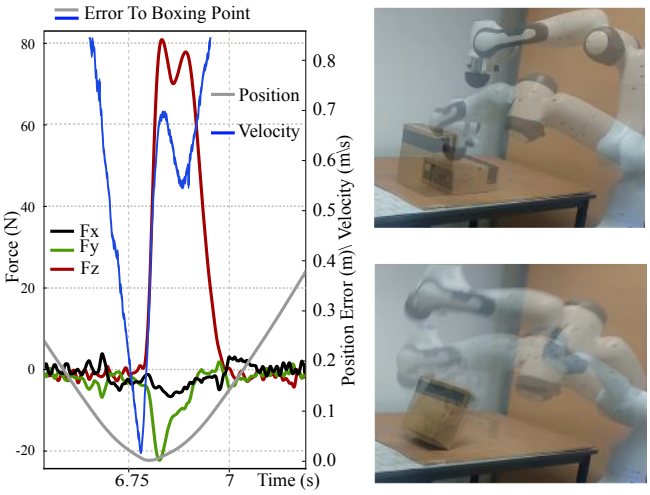

Fig. 10: Boxing sequence (right) and measured impact forces (consequent to desired impacting velocity) with position/velocity error to impacting point M (left).

velocity to induce the intended planned rotational motion (all obtained from computations in Sec. VI).

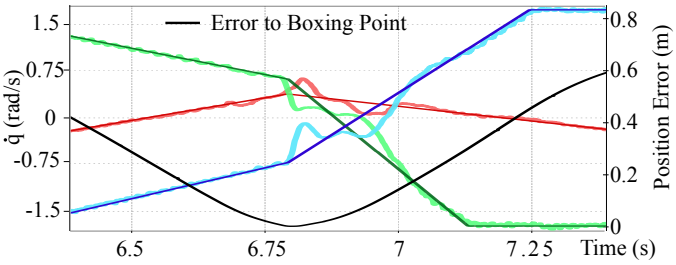

Fig. 11: Desired joint velocity for three joints, $i = \{2, 3, 4\}$, dark colors (red, bleu, green) and the real measured corresponding velocities by the encoders with light colors.

Figure 11 illustrates the velocity tracking performance in the boxing experiment (namely in the presence of velocity jumps produced by the impact), and how important it is to take the robot back to a safe configuration to proceed to with the next task objective.

In case of boxes that cannot reach the desired pose in a stand-alone motion (i.e., would bypass S_2 in Fig. 7), it is possible to plan another robot arm to meet the box a little after the singular configuration S_1 in Fig. 7 and still achieve boxing.

There is no similar existing work to which our approach is to be compared. Indeed, the boxing studied in this paper is an extreme dynamic motion case; in most cases, it is acceptable to gently tilt boxes to grab them with slower dynamics by applying a continuously sustained contact. This can be done as formulated in [32] using tactile feedback, where contact is sustained during the whole motion. In terms of performance, rotating a box in [32] took $\sim$ 160 s. Our implementation of contact-sustained boxing using force sensing took 45 s, whereas impact-boxing is made in around 1 s!

E. Combining different tasks

With our modular impact-aware planning and control approach, it is straightforward to combine previous individual impact-aware tasks. Since the planning can start from any state, no pause is necessary between two successive trajectories. We successfully experimented with a scenario that can be encountered in logistics were a given object at rest would first be boxed to change its pose, then grabbed and then tossed. Figure 12 illustrates such an example with snapshots from the multimedia video of this scenario . For the latter scenario,

Fig. 12: Sequencing boxing, grabbing, and tossing in one run.

Fig. 13 shows the force measurements for part of the task starting from boxing to dual arm tossing (notice that tossing is made in dual arm, that is adapting the computation of the working spaces) indicating boxing performed by the right panda robot (light red region) and the transition from dual grabbing impact (blue region) to sustained contact through an admittance task objective in the QP controller (orange region).

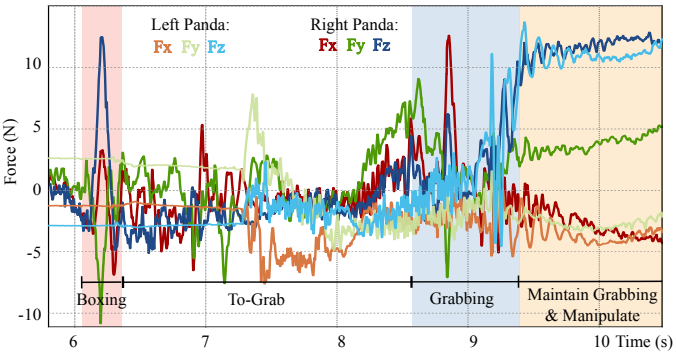

Fig. 13: Measured right panda (see Fig. 12) end-effector applied forces (along x, y, and z-axes) while impacting a box to change its initial rotation. In the grabbing interval, the end-effectors normal forces are shown (also during contact).

F. Discussions and limitations

Although repeatable, our experiments are conducted in well-calibrated settings, blindly (i.e., without using vision for object tracking). Adding object tracking would enhance significantly the performance and robustness of our approach. Moreover, the shape and inertia parameters of the objects are known, which is not always the case in logistics sorting. Yet, online and fast inertia parameter estimation is possible [1] and has to be integrated as a prior task in our planning, which is certainly possible for grabbing or tossing. Yet, for boxing, we cannot escape from first impact that would allow identifying such parameters for unknown objects. We also did not reach the precision wanted for the tossing scenario when objects are picked with a suction cup. First, contact location uncertainties can occur, but this can be mitigated with online inertia identification (including the CoM). The challenging issue is the non-linear deformation of the suction cup. Also, a drop of

pressure at release induces non-negligible perturbations. It is crucial to integrate the suction-cup dynamics in the planning as an extension of the robot model, e.g., [22], [33]. Another issue is the uncertainty in the release moment that can lightly change the object's ballistic trajectory. Since the latter depends on the release pose and velocity, the target can become out of reach. We also noticed that extra care should be given to the end-effector orientation motion at release to prevent undesired interaction between the object and the end-effector after it is released. The experiments also confirmed that for boxing the impacting location and impacting velocities are very sensitive parameters for the success of the tilting.

VIII. Conclusion and Future Work

We propose a decoupled approach to plan motions for a class of impact tasks based on kinodynamic sampling in joint-space. The new planner is plugged into modules for tossing, grabbing, and boxing, each of which links its target impact objective to via-point configuration parameters (in terms of joint-space position and velocity), which our planner takes into account. The entire task is executed by a standard task-space QP controller, which is proven to be highly efficient for generating intentional impacts. This work will be further investigated by benchmarking our developed methods together with others (data driven [8], [7] and reference spreading [17]) in real logistic cases that include robotics with our industrial partner in the frame of the I-AM project. We will also investigate the possibility of casting the entire tasks as a global optimization problem, with more robustness to uncertainties by incorporating suction cup models and additional tasks such as impact assemblies and on-the-fly catching of objects.

References

[1] A. Kurdas, M. Hamad, J. Vorndamme, N. Mansfeld, S. Abdolshah, and S. Haddadin, "Online payload identification for tactile robots using the momentum observer," in *IEEE International Conference on Robotics and Automation*, 2022, pp. 5953–5959.

[2] T. Stouraitis, L. Yan, J. Moura, M. Gienger, and S. Vijayakumar, "Multi-mode trajectory optimization for impact-aware manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 9425–9432.

[3] M. Toussaint, J.-S. Ha, and D. Driess, "Describing physics for physical reasoning: Force-based sequential manipulation planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6209–6216, 2020.

[4] J.-P. Sleiman, J. Carius, R. Grandia, M. Wermelinger, and M. Hutter, "Contact-implicit trajectory optimization for dynamic object manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 6814–6821.

[5] A. Mészáros, G. Franzese, and J. Kober, "Learning to pick at non-zero-velocity from interactive demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6052–6059, 2022.

[6] Y. Wang, N. Dehio, A. Tanguy, and A. Kheddar, "Impact-aware task-space quadratic-programming control," *International Journal of Robotics Research*, 2023.

[7] H. Khurana, M. Bombile, and A. Billard, "Learning to hit: A statistical dynamical system based approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 9415–9421.

[8] M. B. Bombile and A. Billard, "Dual-arm control for coordinated fast grabbing and tossing of an object: Proposing a new approach," *IEEE Robotics Automation Magazine*, pp. 2–13, 2022.

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

[10] H. Kasaei and M. Kasaei, "Throwing objects into a moving basket while avoiding obstacles," in *IEEE International Conference on Robotics and Automation*, 2023, pp. 3051–3057.

[11] M. Monastirsky, O. Azulay, and A. Sintov, "Learning to throw with a handful of samples using decision transformers," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 576–583, 2023.

[12] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman, "Deep predictive policy training using reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 2351–2358.

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

[14] N. Govindan, B. Ramachandran, P. H. V. Sai, and K. M. Krishna, "A novel hybrid gripper capable of grasping and throwing manipulation," *IEEE/ASME Transactions on Mechatronics*, pp. 1–12, 2023.

[15] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation—a survey," *Robotics and Autonomous systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[16] N. Dehio, Y. Wang, and A. Kheddar, "Dual-arm box grabbing with impact-aware mpc utilizing soft deformable end-effector pads," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5647–5654, 2022.

[17] J. J. van Steen, A. Coşgun, N. van de Wouw, and A. Saccon, "Dual arm impact-aware grasping through time-invariant reference spreading control," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1009–1016, 2023.

[18] A. Aydinoglu, P. Sieg, V. M. Preciado, and M. Posa, "Stabilization of complementarity systems via contact-aware controllers," *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1735–1754, 2022.

[19] B. Brogliato, "Modeling, analysis and control of robot–object nonsmooth underactuated lagrangian systems: A tutorial overview and perspectives," *Annual Reviews in Control*, 2022.

[20] Y. Shirai, D. K. Jha, A. U. Raghunathan, and D. Romeres, "Chance-constrained optimization in contact-rich systems," in *American Control Conference*, 2023, pp. 14–21.

[21] M. T. Mason, "Toward robotic manipulation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 1–28, 2018.

[22] X. Cheng, Y. Hou, and M. T. Mason, "Manipulation with suction cups using external contacts," in *International Symposium of Robotics Research*. Springer, 2022, pp. 692–708.

[23] Y. Shirai, D. K. Jha, and A. U. Raghunathan, "Robust pivoting manipulation using contact implicit bilevel optimization," *arXiv preprint arXiv:2303.08965*, 2023.

[24] A. O. Onol, P. Long, and T. Padir, "Contact-implicit trajectory optimization based on a variable smooth contact model and successive convexification," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 2447–2453.

[25] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning." in *IEEE International Conference on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.

[26] L. Berscheid and T. Kröger, "Jerk-limited real-time trajectory generation with arbitrary target states," in *Robotics: Science and Systems*, 2021.

[27] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *IEEE International Conference on Intelligent Robots and Systems*, 2014, pp. 3713–3719.

[28] K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 2493–2498.

[29] C. Lau and K. Byl, "Smooth RRT-connect: An extension of RRT-connect for practical use in robots," in *IEEE International Conference on Technologies for Practical Robot Applications*, 2015, pp. 1–7.

[30] M. Halm and M. Posa, "A quasi-static model and simulation approach for pushing, grasping, and jamming," in *Workshop on the Algorithmic Foundations of Robotics*, 2020, pp. 491–507.

[31] K. Bouyarmane and A. Kheddar, "On weight-prioritized multitask control of humanoid robots," *IEEE Transactions on Automatic Control*, vol. 63, no. 6, pp. 1632–1647, 2017.

[32] Y. Shirai, D. K. Jha, A. U. Raghunathan, and D. Hong, "Tactile tool manipulation," in *IEEE International Conference on Robotics and Automation*, 2023.

[33] H. Pham and Q.-C. Pham, "Critically fast pick-and-place with suction cups," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 3045–3051.