Motion Planning and Inertia Based Control for Impact Aware Manipulation

Harshit Khurana[1] and Aude Billard[1]

***Abstract*—In this paper, we propose a metric called hitting flux which is used in the motion generation and controls for a robot manipulator to interact with the environment through a hitting or a striking motion. Given the task of placing a known object outside of the workspace of the robot, the robot needs to come in contact with it at a non zero relative speed. The configuration of the robot and the speed at contact matter because they affect the motion of the object. The physical quantity called hitting flux depends on the robot's configuration, the robot speed and the properties of the environment. An approach to achieve the desired directional pre-impact flux for the robot through a combination of a dynamical system (DS) for motion generation and a control system that regulates the directional inertia of the robot is presented. Furthermore, a Quadratic Program (QP) formulation for achieving a desired inertia matrix at a desired position while following a motion plan constrained to the robot limits is presented. The system is tested for different scenarios in simulation showing the repeatability of the procedure and in real scenarios with KUKA LBR iiwa 7 robot.**

***Index Terms*—Coupled Dynamical Systems, Dynamic Manipulation, Inertial Control, QP Control.**

I. Introduction

Pick-and-place tasks for fixed manipulators, are incredibly common in robotics. Although they are extremely helpful, they suffer from the following restrictions:

- the initial and final positions of the object are within the reach of the robot, i.e. in its workspace
- limitations exist on the object size, shape and mass due to hardware limitations of the robot such as payload capacities, gripper requirements and gripper size.

To move an object from one place to another, pushing primitive [1] is also used. Pushing primitive refers to the application of force on the object in the environment, producing the desired effect. The robot remains in contact with the object during pushing and hence at low velocities, many different works assume such motion to be quasi-static (ignoring the dynamic effects of the robot on the pushed object) [2], [3] [1]. Pushing helps in both: alleviating the limitations of pick-and-place operations for objects that are heavier than the lifting capacity of the robot as well as in manipulating objects that are hard to grasp, such as small or irregular shaped objects. Most of the pushing motion by the robot is quasi-static which normally does not impart large velocity to the object. Hence the motion of the object is primarily restricted to the reachable workspace of the robot. This motion primitive also neglects the inertial forces. We use a hitting or striking primitive to increase the space in which an object can be placed by a robot. Picking and throwing is another strategy to extend the workspace of the robot. Fast picking and throwing using a dual arm system has been discussed in [4] where the motion planner takes into account both fast grasping and the feasibility of throwing motions. The process of hitting differs in the way the robot comes in contact with the object. Hitting allows for a non zero relative speed between the robot and the object at contact. Hitting or striking dives into the realm of dynamic manipulation where the inertial properties of the robot and the environment must be considered. Such primitives can be used in many robot-environment interactions such as sliding an object on a table to place it in a given position; moving obstacles out of the way for mobile manipulators; exploring different strategies in sports such are baseball, football, boxing; learning environment properties for better robot-environment interaction, sorting different objects (varying in inertial properties); and interacting with objects with non-uniform mass distribution.

In our previous work, [5], we showed that hitting can produce repeatable and predictable post-impact object motion. The benefits of such dynamic manipulation are:

- The workspace of the robot is expanded. A fixed-base manipulator can use its dynamics to place an object outside its workspace.
- The robot can impart velocities to objects greater than its velocity - exceeding the hardware limits (depending on collision properties)
- The process is faster than quasi-static pushing.

In [5], in order to move different boxes, one needed to learn a hitting model for each box separately, since the same robot's motion would induce distinct motion on the object if these differed in their physical properties. In addition, the motion of the robot was restricted such that the joint configuration of the robot and thus its inertial properties on each impact were similar. This paper addresses this issue by decoupling the post-impact object behavior from the pre-impact robot motion. We propose a control metric called hitting flux, which used in a combination of DS-based motion planner and a control system for a fixed manipulator, allows us to produce similar post impact motion for an object with known inertial properties. We are interested in a manipulation strategy that exploits an intentional impact to impart motion to the object in order to place it in a desired location. For the entire process, the configuration of the robot is as important as its motion when it contacts the object. To have a desired

[1]Harshit Khurana and Aude Billard are with the Learning Algorithms and Systems Laboratory, EPFL, Lausanne, Switzerland, e-mail: harshit.khurana@epfl.ch and aude.billard@epfl.ch

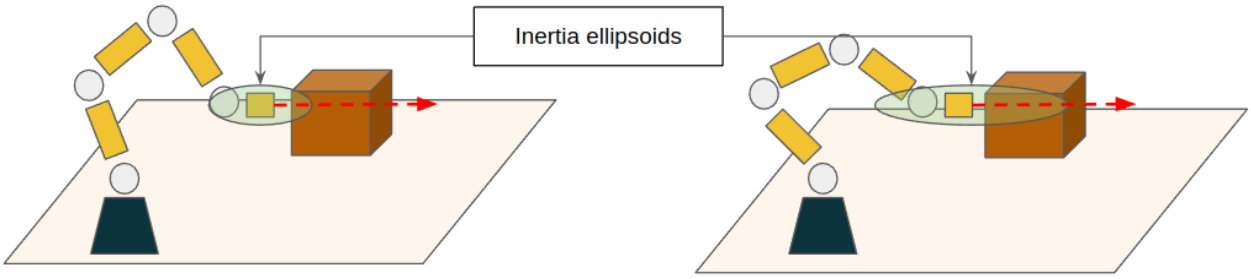


Fig. 1: As seen in Section III, the robot configuration changes the inertial properties of the robot and this leads to changes in the post-impact speed of the object, resulting in different motion of the object after impact. Thus, we deal with controlling the pre-impact robot motion such that the post-impact object behavior is the same.

effect on the environment, i.e., imparting similar motion on different-sized boxes, for different initial positions of the box, the physics of collision needs to be taken into account. This helps understand the control parameters of the problem, as shown in Section III.

Furthermore, the tasks for manipulating the environment are better suited to be formulated in the *task space* or *Cartesian space*. Writing the motion plan of the robot according to the task requirements is intuitive in the task space and hence can lead to easy adaptability to task changes and planning. In the task of hitting an object, we are aware of the current position of the object, where the object needs to be placed, and the robot's end effector position. The physics of the motion of the object can be used to understand the initial conditions (speed and direction of motion) of the object for the desired motion. This helps generate a motion plan for the robot in the task space, while the controller considers the constraints that come with the limitations of the robot.

Contributions of this paper:

- Proposition of a control metric called ***hitting flux*** that depends on robot's inertia, speed and object mass, and is proportional to the post impact object's speed and utilising it through a DS-based motion planner that generates a hitting motion in 3D task space ($\mathbb{R}^3$) and a control structure to achieve desired directional inertia and repeatable post impact object motion ($\in \mathbb{R}$)
- A control structure to achieve a desired inertia matrix ($\in \mathbb{S}^3_{++}$) while reaching a desired position $\in \mathbb{R}^3$ utilizing gradient of Stein distance [6]
- Validation through real robot implementation showing applicability of generating motion using hitting flux as a metric to hit box objects in a repeatable fashion (RMSE error in average object motion) and its applicability to boxes of different sizes and masses.

II. Related Work

A. *Pushing and Hitting Primitives for Manipulation*

Pushing and hitting manipulation techniques are not new to the manipulation literature, and are gaining popularity given the growth of geometry-aware motion planning for robots [7], [8], [9] and, data driven controls and planning algorithms [10], [11]. Humans use this skill in recreational sports such as Air-Hockey, football, table tennis, baseball, badminton and volleyball where the striking motion is used to generate a desired trajectory for the puck or ball. Applying those strategies in robotics has helped advance motion planning with intentional impacts. Robot-environment impacts exist wherever robots interact with the environment. Such impacts can be found in robots playing football [12], table tennis [13], golf [14], baseball [15] and in quadrotors juggling using a small ball [16]. In these applications, the robot does not plan its motion according to its knowledge of the post impact change in the environment. The contact normal between the robot and the object passes through the center of mass of the object due to its spherical nature and hence does not require convergence guarantees (tuning) of the motion model to pass though a specific area of the object.

In robotic table tennis, a substantial work exists on robot trajectory optimization ([17], [18], [19], [20]) and learning, such that the robot learns how to move so that the ball lands in an appropriate place. Such trajectory optimisation methods deal with high computational cost of solving analytical models, can suffer from modelling errors, and also depend on the initial guess for the solution. [21], [22] create a method to generate high velocities to play air hockey. The method includes creating a specific path for the end effector of the robot to traverse. It aims to hit the puck with the fastest achievable speed in the path designed to hit the puck by exploiting the redundancy of the robot. Since, the inertia of a hockey puck is negligible compared to that of the robot, this approach makes sense. In cases of hitting objects with higher inertia, the inertias of both, the robot and the object need to be considered.

B. *Dynamical Systems as Motion Plans*

Dynamical systems [23] represent evolving phenomena by a specific equation for the rate of change of a state. The state is a representation of the system. In the robotics regime, they have been used to create motion plans for robots such as manipulators [14] and mobile platforms. The advantage

of using DSs is that they are able to provide stable and closed-form motion plans that are easily controllable. They are heavily used in learning from demonstration [24], where the data from demonstrated trajectories is used to learn stable DSs. Stable autonomous DSs are robust to external disturbances, which allows for instantaneous recalculation of trajectories for a given purpose (stability depends on its specific definition). This makes them suitable for tasks such as interaction with the unstructured environment. Control strategies such as inverse kinematics control, inverse dynamics control and impedance control [25] can be applied to DSs, see [26]. DSs allow us to represent the motion of the robot in terms of its desired task space properties, such as inertia, end effector speed and the object position, thus not requiring us to create a motion plan for such changes every time.

Dynamical systems represent a closed form solution to a family of known trajectories, which allow fast replanning. They are chosen firstly, because we know the basic movement of hitting and secondly, to explicitly understand the effects of hitting flux on the post impact speed of the boxes. They provide fast computation and allow us to mathematically formulate guarantees for convergence. Other motion planning algorithms which combine motion generation and controls such as model predictive control are a completely valid approach for hitting and generating desired hitting flux. They are better suited in planning motion in unknown conditions where the solution is initially unknown and requires guaranteeing satisfaction of complex constraints at run time, such as controlling for robot's balance. They suffer from high computational cost, modeling inaccuracies and there is current research on the stability of MPC policies. Parallel computation allows sampling based MPC to work for manipulators but it needs to be tested for fast dynamic motions [27], [28]. In our work, we have an idea of the motion in the task space and the environment is not actively in motion, thus making the use of dynamical systems an easy choice.

C. *Symmetric Positive Definite (SPD) Properties of Robot Manipulators*

Manipulators possess properties which are SPD in nature such as manipulability matrices (velocity, dynamic) [29], joint space mass matrices and task space inertia matrices etc. We need to understand the behavior of SPD manifold since we want to understand task space inertia matrices. Extensive research has examined controlling manipulability metric values for the robot to ensure non-singular configurations during a task [30], achieving maximum end effector speed at a certain configuration and so on. [22] [7], [9] create a way of learning motion from human demonstrations, learning both the trajectory, and SPD properties of the robot. Since the data comes from the demonstrated trajectories, the task space end effector position and the corresponding desired SPD property are assumed to be feasible, and it does not help achieving a desired SPD property at a certain end effector position if they are inferred only from the task at hand. [8] can be used to generate trajectories that achieve a desired SPD property at a certain end effector position, provided that it is achievable but suffers from the difficulties of finding a suitable Riemannian metric. In this paper, we will use the SPD property of the inertia matrix to control the configuration of the robot to achieve a desired inertia matrix.

III. Preliminaries

A. *Breaking Down the Collision Problem*

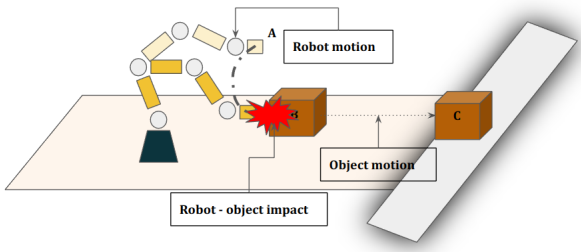


Fig. 2: The figure shows the process of collision between the robot and the object. The robot moves from A to B to create an impact. The robot-object collision imparts post-impact velocities to both the robot and the object. The object moves to desired location C after the impact.

The whole process of interacting with the environment through intentional impacts is broken down into three different physically independent phenomena shown in Fig. 2:

1) Robot motion: The robot moves intentionally to generate an impact with the object from position A to B.

2) Robot-Object impact: The robot-object interaction imparts post-impact velocities to both which are extremely hard to control but can be estimated [31]. These depend on the type of contact, frictional parameters between the colliding bodies, and coefficient of restitution.

3) Object motion: Object motion can be nominally described by equations of motion under forces given the initial state of the object (here, it is the post-impact state of the object).

In this paper, we discuss the process of creating impacts resulting in *similar initial conditions for the object* post impact. This allows for repeatable motion for the object and hence can make the system invertible, i.e., allowing us to later solve the following problem: given a desired motion of the object, how do we want the robot to come in contact with it. The following sections assume that the collision between the robot and the object is through a point contact and imparts velocity at the center of the mass of the box. Given the size of boxes and the end effector, the system can have line contacts in both horizontal and vertical directions, and surface contacts. The orientation of the end effector and tuning of the hitting motion as described in the Appendix (Section XI) allow the center of mass to be in the middle of the end effector of the robot allowing for such assumption to hold.

B. *Collision Mechanics*

We discuss the collision of the robot and the object. The cartesian position of the robot end effector and the object

are χ_e and χ_o respectively ($\chi_e, \chi_o \in \mathbb{R}^3$). Consider, at the moment of impact, that the task space inertia of the robot perceived at the end effector is $\Lambda \in \mathbb{S}^3_{++}$, the object mass is $M_o = m_o \mathbb{I}_3 \in \mathbb{S}^3_{++}$, and the pre- and post-impact velocities of the end effector and the object are $(\dot{\chi}_e^-, \dot{\chi}_o^-) \in \mathbb{R}^3$ and $(\dot{\chi}_e^+, \dot{\chi}_o^+) \in \mathbb{R}^3$ respectively. Let the coefficient of restitution for the impact be E. It is defined as the ratio of the final relative velocity between the colliding bodies to the initial relative velocity normal to the impact and can be represented as a scaled projection matrix along the hitting normal with a scaling factor < 1. We make the following assumptions for the collision equations:

Assumptions:

- The pre-impact velocity of the object, $\dot{\chi}_o^- = 0$.
- The robot-object contact is instantaneous, i.e. the configuration, and hence, $\Lambda(q)$, of the robot remains constant during the collision, and there is no contact between the robot and the object after the collision.
- The collision provides negligible tangential impulse (normal to the hitting / impact direction). Let $\epsilon \in [0, 1]$ be the coefficient of restitution along the line of impact. It determines the energy loss in the hitting direction or also called line of impact. This assumption requires the motion plan to generate pre-impact robot velocity in the hitting direction normal to the surface of the object i.e., $\dot{\chi}_e^- = \|\dot{\chi}_e^-\| \hat{h}$ (refer Section V, XI) and enables the motion of the object post impact to be in the impact direction, i.e., $\dot{\chi}_o^+ = \|\dot{\chi}_o^+\| \hat{h}$.
- If the inertia of the robot can be designed to be substantially larger in the hitting direction (refer Section VI), its post impact speed can be assumed to remain in the direction $\hat{h}$, because of low pre-impact momentum in the orthogonal directions $\hat{h}_{\perp_1}, \hat{h}_{\perp_2}$, implying $\dot{\chi}_e^+ \simeq \|\dot{\chi}_e^+\| \hat{h}$.

Using the above assumptions and from the principle of conservation of momentum during the impact we have:

$$\Lambda \dot{\chi}_e^- + M_o \dot{\chi}_o^- = \Lambda \dot{\chi}_e^+ + M_o \dot{\chi}_o^+ \tag{1}$$

Pre-multiplying Eq. 1 with $\hat{h}^T$ and using $\dot{\chi}_o^- = 0$, we have

$$\hat{h}^T \Lambda \dot{\chi}_e^- = \hat{h}^T \Lambda \dot{\chi}_e^+ + \hat{h}^T M_o \dot{\chi}_o^+ \tag{2}$$

Using $\dot{\chi}_e^- = \|\dot{\chi}_e^-\| \hat{h}$ and $\dot{\chi}_e^+ \simeq \|\dot{\chi}_e^+\| \hat{h}$ we have

$$\begin{aligned} \hat{h}^T \Lambda \hat{h} \left\|\dot{\chi}_e^-\right\| &= \hat{h}^T \Lambda \hat{h} \left\|\dot{\chi}_e^+\right\| + \hat{h}^T M_o \hat{h} \left\|\dot{\chi}_o^+\right\| \\ \lambda_h \left\|\dot{\chi}_e^-\right\| &= \lambda_h \left\|\dot{\chi}_e^+\right\| + m_o \left\|\dot{\chi}_o^+\right\| . \end{aligned} \tag{3}$$

We denote $\lambda_h = \hat{h}^T \Lambda \hat{h}$ as the *directional inertia*.

From the definition of restitution [32] along the impact normal ($\hat{h}$),

$$\epsilon \, \hat{h}^T (\dot{\chi}_e^- - \dot{\chi}_o^-) = \hat{h}^T (\dot{\chi}_o^+ - \dot{\chi}_e^+) \ . \tag{4}$$

This simplifies into:

$$\epsilon \left\|\dot{\chi}_e^-\right\| = \left\|\dot{\chi}_o^+\right\| - \left\|\dot{\chi}_e^+\right\| \ . \tag{5}$$

From Eq. 3 and 5, we have the post-impact object speed as:

$$\left\|\dot{\chi}_o^+\right\| = (1+\epsilon)(1 + \frac{m_o}{\lambda_h})^{-1} \left\|\dot{\chi}_e^-\right\| \ . \tag{6}$$

Reproducing similar post-impact conditions for the object translates to achieving a desired post-impact object velocity, $\dot{\chi}_o^+$. Eq. 6 depicts the requirements for the repeatable collision and interaction with the environment. Given the post-impact object behavior ($\dot{\chi}_o^+$) and mass of the object M_o, we need to control for the quantity given by $(1+\epsilon)(1 + \frac{m_o}{\lambda_h})^{-1} \|\dot{\chi}_e^-\|$.

The coefficient of restitution in the hitting direction, ϵ can range from 0 to 1. The coefficient of restitution scales the motion of the object from a completely inelastic ($\epsilon = 0$) to completely elastic ($\epsilon = 1$) collision. For a given object robot pair, the motion of the object for a given robot motion should remain statistically similar. We assume ϵ zero to provide a reference velocity to the robot's end effector. The post impact object velocity would depend on the real value of ϵ and for a given object-robot pair, the motion of the object for a given robot motion should remain statistically similar. The main idea that we want to convey here is that the quantity that we control for in the robot's motion does not depend on the coefficient of restitution. With the above assumption, we write the *directional hitting flux* as

$$\phi = (1 + \frac{m_o}{\lambda_h})^{-1} \left\|\dot{\chi}_e^-\right\| \ . \tag{7}$$

Estimating ϵ is a research problem for understanding environment behavior which we are working on in our current research. $\epsilon = 0$ will lead to hitting being the same as pushing and releasing, which as shown in Section IX is not the case, if $\epsilon \neq 0$.

We also define *Hitting Flux*, $\Phi \in \mathbb{R}^3$, based on the above assumptions. This derivation can be found in Appendix (Section XI).

IV. Problem Statement

Given an object placed at Cartesian position $\chi^* \in \mathbb{R}^3$, create and execute a motion for the end effector of the robot to hit the object of mass m_o with a desired directional hitting flux ϕ_{des}. Additionally, make this independent of the direction, i.e. achieve a desired inertia matrix, $\Lambda^* \in \mathbb{S}^3_{++}$ at a desired cartesian position, χ^*.

V. Robot Motion

A. *Pre-impact Dynamical System:*

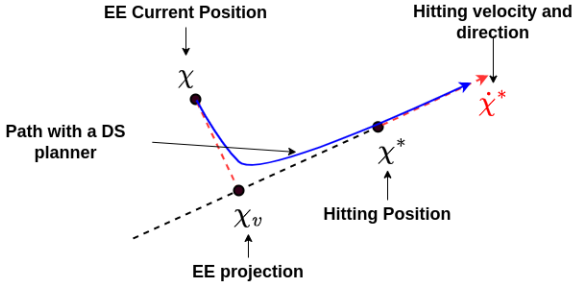


Fig. 3: Design intuition for the DS as a motion generator. The end effector (EE) moves toward its projection on the desired direction and a constant flow is added. The effect of the two different flows is controlled by a weighting parameter, which depends on the distance of the end effector from its projection.

An autonomous DS is used as a motion generator for the end effector of the robot. The current position of the robot end effector is given by χ. It needs to pass through χ^* with velocity $\dot{\chi}^*$ (refer Fig. 3). The DS is the reference velocity, $\dot{\chi}$ of the end effector of the robot. Consider the following DS:

$$\begin{aligned} \dot{\chi} = f(\chi) = \alpha(\chi)\dot{\chi}^* + (1-\alpha(\chi))[A(\chi - \chi_v)] \\ f(\chi), \chi, \dot{\chi}, \chi^*, \dot{\chi}^* \in \mathbb{R}^3, \alpha(\chi) \in \mathbb{R}, A \in \mathbb{S}^3_{--} \end{aligned} \tag{8}$$

where,

$$\begin{aligned} \alpha(\chi) = e^{\frac{-\|\chi - \chi_v\|}{\sigma^2}}, \quad \sigma \in \mathbb{R}_+ \\ \chi_v = \chi^* + \frac{< \chi - \chi^*, \dot{\chi}^* >}{\|\dot{\chi}^*\|^2}\dot{\chi}^* \quad . \end{aligned} \tag{9}$$

$< ., . >$ is the inner dot product, $\chi, \dot{\chi}, \chi^*, \dot{\chi}^* \in \mathbb{R}^3$. χ_v denotes a virtualized end effector. It is the projection of the current end effector position along the hitting direction. The tuning parameters depending on the robot and its workspace are σ. Eq. [8] is a combination of two vector fields, $\dot{\chi}^*$ represents a continuous flow at the desired hitting speed, and $A(\chi - \chi_v)$ represents a converging flow towards the virtual end effector position, if $A \prec 0$. $\alpha(\chi)$ is a weighting function that creates the flow with the desired speed $\dot{\chi}^*$ once the robot end effector is close to χ_v.

The DS depends on the tuning of the hyperparameter σ, which changes how the DS converges to the line joining the virtual end effector and the desired hitting position as shown in Fig. 4. We can impose an adaptive σ or have a higher bound on its value for the DS to have the desired behavior. For further analysis, refer to Section XI.

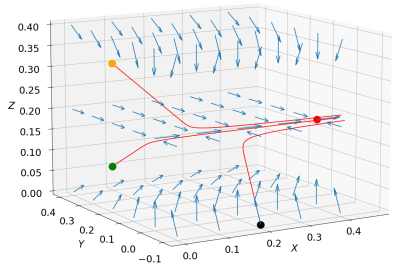


Fig. 4: The vector field showing the flow of the end effector as in Eq. 8. The red marker shows the position of the object that needs to be hit. The figure also shows three different paths the robot takes from three different initial positions (in orange, green and black)

Design intuition: Eq. 8 describes a DS that creates a hitting motion for an object placed at the cartesian position χ^*. Fig. 3 shows the position of the end effector and the virtual end effector. The virtual end effector is used for two different purposes:

- It reduces the collision problem from a 6 DoF motion collision to a collision of two objects moving along the direction provided by the DS i.e., the collision normal
- It provides robustness to change in position of the end effector through external disturbances. χ_v captures the variations in the position of the end effector caused by the disturbances.

This allows us to modify the dynamical system we used in [5] and simplify it by using a constant flow and one linear dynamical system instead of three individual dynamical systems. There are fewer tuning parameters, which makes it easier to implement it on the real robot.

VI. Robot Control

The dynamics of n degrees of freedom robot manipulator in task space can be written as

$$M(q)\ddot{q} + C(q,\dot{q}) + g(q) = \tau + J_c^T F_c \tag{10}$$

where $q \in \mathbb{R}^n$ denotes the joint space configuration vector. $M(q) \in \mathbb{R}^{n\times n}$ and $C(q,\dot{q}) \in \mathbb{R}^n$ denote the robot's joint space mass matrix, and the centrifugal and coriolis forces. $g(q) \in \mathbb{R}^n$ is the vector of the gravity wrench. $\tau \in \mathbb{R}^n$ represents an external wrench applied by the robot, and $F_c \in \mathbb{R}^6$ is the control wrench of the robot. The task space inertia, $\Lambda(q)$ of the robot at the end effector is given by:

$$\Lambda(q) = (J(q)M^{-1}(q)J^T(q))^{-1} \quad \in \mathbb{S}^6_{++} \tag{11}$$

where, q is the joint configuration of the robot, $J(q)$ is the Jacobian, $M(q)$ is the joint space mass matrix of the robot [33].

Here, we are primarily focusing on the end effector translational tasks. Thus, we have the translational task space inertia perceived at the robot's end effector:

$$\Lambda_t(q) = (J_t(q)M^{-1}(q)J_t^T(q))^{-1} \quad \in \mathbb{S}^3_{++} \quad . \tag{12}$$

The subscript t refers to the translational part of the robot properties. The different controllers used and introduced in this paper are as follows:

A. *Inverse Kinematics Control*

To follow the reference dynamical system $\dot{\chi} = f(\chi)$, we can use IK controller. This controller is a solution for the optimisation:

$$\begin{aligned} \dot{q} = \ & \underset{\dot{q}}{\text{argmin}} \ \|f(\chi) - J(q)\dot{q}\|_2^2 \\ \implies \dot{q} = \ & J^\dagger(q) f(\chi) \end{aligned} \tag{13}$$

where $J(q)^\dagger = J(q)^T(J(q)J(q)^T)^{-1}$ is the Moore-Penrose pseudo inverse of the Jacobian. This is used as a base controller.

B. *Manipulability Maximization Controller:*

The robot must be able to achieve high end effector velocities. This is ensured by prioritizing the configurations that have high velocity manipulability metrics, along the hitting direction, $\hat{h} \in \mathbb{R}^3$. Finding this configuration is formed as a least squares problem solved as SLSQP (Sequential Least SQuares Problem) using SciPy [34].

$$\begin{aligned} q_m = \ & \underset{q}{\text{argmax}} \left\| J(q)^T \hat{h} \right\|_2^2 \\ \text{s.t.} \quad & FK(q_m) = \chi^* \\ & q_l \leq q \leq q_u \end{aligned} \tag{14}$$

where, $FK(q)$ is the forward kinematics model of the robot, and q_l and q_u are the lower and upper bounds on the joint limits. χ^* is the point of hit. $\hat{h}$ can either be provided in a task to be achieved or calculated from the current and the desired position of the object (direction from current to the desired object position). The manipulability metric, $m = \left\| J(q)^T \hat{h} \right\|_2^2$ [29]. The following equation serves as the controller for achieving joint configurations close to the maximum manipulability configuration ($\beta_1 \in \mathbb{R}_+$ is the proportional gain). This controller is used later to exploit the redundancy of the 7DoF manipulator:

$$\dot{q} = \beta_1 (q_m - q) \quad . \tag{15}$$

C. *Increasing Directional inertia in the Null Space*

The hitting DS (Eq. 8) aligns in the direction of hitting at the object, and in the hitting direction, for some purposes, for As the robot end effector moves along $f(\chi)$, the following optimisation allows for increasing the inertia perceived at the end effector along the direction of the vector field $f(\chi)$.

$$q = \underset{q}{\operatorname{argmax}}\ \lambda_h(q) \quad . \tag{16}$$

The increasing inertia controller is formulated in terms of joint velocities through an iterative gradient ascent solution of Eq. 16 that runs at the controller frequency:

$$\dot{q} = \beta_2 \nabla_q \lambda_h(q) \quad . \tag{17}$$

Combining the controllers from Eqns. 13, 15 and 17 while exploiting redundancies, we have the final controller equation:

$$\dot{q} = J^\dagger(q) f(\chi) + N[\beta_1 (q_m - q) + \beta_2 \nabla_q \lambda_h(q)], \quad \beta_1, \beta_2 \in \mathbb{R}_+ \tag{18}$$

where, $N = I - J(q)^\dagger J(q)$ is the dynamically consistent Null space for joint velocities [35], [36], and β_1, β_2 are hyperparameters, which control the relative weights of achieving high manipulability configuration and moving in the direction of increasing directional inertia.

D. *Specific Directional Inertia in the Null Space:*

Instead of maximizing the directional inertia, here we want to achieve a desired directional inertia λ^* while tracking the dynamical system, with priority to the latter. As the robot end effector moves along $f(\chi)$, the following optimization allows for tracking the desired inertia perceived at the end effector along the direction of the vector field $f(\chi)$.

$$q = \underset{q}{\operatorname{argmin}}\ |\lambda_h(q) - \lambda^*|_2^2 \quad . \tag{19}$$

To control for the directional inertia in the null space, in the gradient ascent solution for Eq. [16], $\nabla_q \lambda_h(q)$ [1] is multiplied with $(\lambda_h(q) - \lambda^*)$.

$$\dot{q} = -\beta_2 \nabla_q \lambda_h(q) (\lambda_h(q) - \lambda^*) \quad . \tag{20}$$

[1]The derivation of $\nabla_q \lambda_h(q)$ is shown in Sec. XI.

This changes the joint velocities towards the configuration with the desired directional inertia if the redundancy allows for it. The final controller hence can be written as:

$$\dot{q} = J^\dagger(q) f(\chi) + N[\beta_1 (q_m - q) + \beta_2 (-\nabla_q \lambda_h(q) (\lambda_h(q) - \lambda^*))], \quad \beta_1, \beta_2 \in \mathbb{R}_+ \quad . \tag{21}$$

With the described control system leading to directional inertia values close enough to the desired values, here we formulate a strategy to achieve the desired directional flux, ϕ^*. To achieve the desired directional flux, ϕ^*, the desired velocity, $\dot{\chi}^*$ in the DS $f(\chi)$ (Eq. 8) depends on ϕ^*.

$$\|\dot{\chi}^*\| = \phi^* (1 + m/\lambda_h(q)) \tag{22}$$

where $\alpha(\chi)$ and χ_v are the same as in Eq. [9].

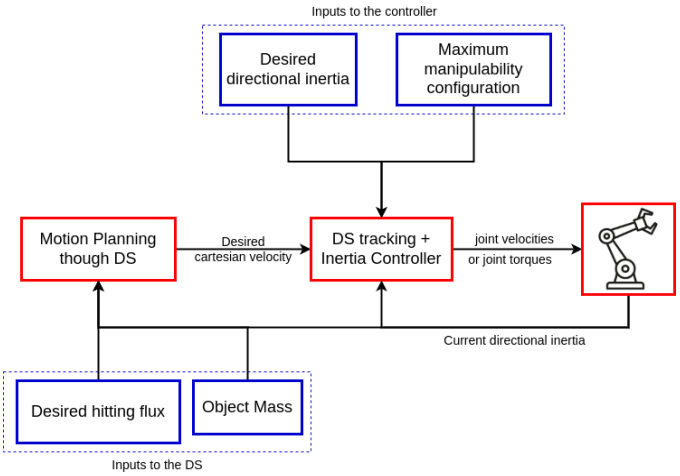


Fig. 5: The diagram shows the flow of the algorithms. The desired hitting flux and the object mass, along with the current inertia of the robot are inputs to the DS based motion plan. The desired cartesian velocity (output of the DS) along with the desired directional inertia and the configuration of maximum manipulability form the input to the controller which outputs joint velocity / torques to the robot.

VII. Motion in Inertial Manifold

In the previous section, the controller is designed to achieve the desired inertia and desired flux in a specific direction. This exploits the redundancy of the system and allows achieving a specific part of the inertia matrix that is a projection of the inertia matrix along a desired direction.

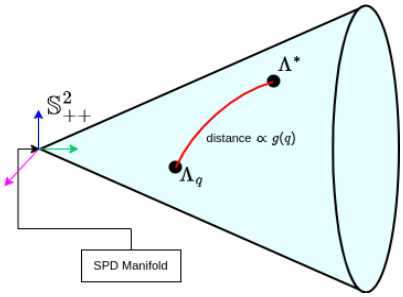


Fig. 6: An example of stein distance on the SPD manifold. Shown above is $\mathbb{S}^2_{++}$ manifold. It can be represented as a cone and Λ_q and Λ^* are two points on this manifold. The distance between the points is shown as the red curve. This distance is proportional to the stein distance g(q).

Task space inertia matrices belong to the SPD manifold. To calculate the distance between two SPD matrices, we use the *Stein divergence* [6] metric. This is an estimate of the distance two SPD matrices are on the manifold. If at a joint configuration q, the robot has a task space inertia Λ_q and the desired task inertia is Λ^*, then the Stein divergence, $g(q)$ is given as:

$$g(q) = |log(det(\frac{\Lambda^* + \Lambda_q}{2})) - \frac{1}{2}log(det(\Lambda^*\Lambda_q))| \in \mathbb{R} \quad . \tag{23}$$

We minimize the distance between the current inertia matrix and the desired inertia matrix, while following a DS under joint position and velocity constraints. The optimization is formulated to output joint velocities $\dot{q}$. To minimize $g(q)$, we use the gradient descent algorithm which finds the steepest gradient $\nabla_q g(q)$ in the direction of decreasing $g(q)$ and projects joint velocities along the direction of the gradient. For a high control frequency, the evolution of $g(q)$ can be written as $g_{t+dt}(q) = g_t(q) + \dot{g}(q)dt = g_t(q) + (\nabla_q g(q))^T \dot{q}dt$. This leads to $g(q)$ decreasing in value.

$$\begin{aligned} \dot{q} = &\underset{\dot{q}}{\text{argmin}} \quad \frac{1}{2}\left\|f(\chi) - J\dot{q}\right\|_2^2 + k_1\dot{g}(q) + k_2\left\|\dot{q}\right\|_2^2 \\ \text{s.t.} \quad &\dot{q}_{min} \leq \dot{q} \leq \dot{q}_{max}, \quad q_{min} \leq q + \dot{q}dt \leq q_{max} \\ &k_1, k_2 \in \mathbb{R}_+ \end{aligned} \tag{24}$$

where, q is the current joint position, dt is the time taken by one control loop, k_1 and k_2 are the weights for the penalties on derivative of the stein distance [2] and the joint velocity norm. For the real robot experiments, to not allow sudden jumps in the joint velocities, a penalty on the norm of the joint velocities is added to the objective function. In terms of reaching the desired inertia matrix, the solutions (joint configurations) become fewer and the physical limitations of the robot come into account. This leads the solver to be stuck in local minima. The solver used is qpOASES [37].

VIII. Simulation Experiments

For the simulation, we use the PyBullet environment [38]. The robot motion and the controller scheme are implemented as in sections V and VI [3]. To understand the applicability of the theory described we perform the following experiments:

- Following a desired path with three different controllers analyzing the inertia changes through the trajectory (VIII-A)
- Hitting an object of known mass with different desired pre-impact hitting fluxes and observing the displacement of the object (VIII-B)
- Achieving the same desired pre-impact hitting flux at different task space positions (VIII-B)
- Testing the controller (Eq. 24) to achieve the full translational task space inertia matrix while following a trajectory (VIII-B2)

A. *Robot Motion:*

*1) **Comparison of controllers**:* We compare, in simulation here, the effect of including the inertia gradient in the Null Space control. The robot is commanded to go to a final desired position χ^* using a stable [4] linear DS $\dot{\chi} = A(\chi - \chi^*), A \prec 0$, with controllers defined in subsection VI-A, VI-C and VI-D. Desired final position, $\chi^* = [0.5, 0.5, 0.5]$ and $\hat{h} = [1, 0, 0]$ (hitting direction is along the X axis). The desired directional inertia commanded for the Specific inertia controller (Eq. 21) is 6 kg.

TABLE I: Quantitative comparison of the final achieved directional inertia for different controllers

No.	Final directional inertia (kg)	Desired directional inertia (kg)
IK	6.47	no control
IK + inc. dir. inertia	8.49	as large as possible
IK + sp. dir. inertia	6.17	specific value (6 kg here)

Fig 7a shows qualitatively, the three different configurations of the robot at the desired end effector position, with the different controllers. The IK control finds the joint configuration closest to the initial configuration of the robot that reaches the desired end effector position. The increasing directional inertia controller aligns the robot in the desired direction to maximize the inertia in the said direction. Reaching the desired inertia value of 6 kg is feasible with the desired end effector position. Hence, we see the robot reaching a similar value of inertia with both, the increasing directional inertia controller and the specific directional inertia controller. The comparison of the achieved final directional inertia for the different controllers is presented in Table I.

The evolution of directional inertia through the trajectory are shown in Fig. 8a. The Specific directional inertia controller maintains the directional inertia along the path taken by the robot when it is feasible, which is not the case with Inverse Kinematics controller, and and with the increasing directional inertia controller, the robot configuration changes to increase the directional inertia.

[2]The derivation of $\nabla_q g(q)$ is shown in SectionXI.

[3]The code for simulation is available here - code

[4]Exponential Stability [39]

B. Robot-Environment Interaction

*1) **Hitting with different pre-impact flux**:* This simulation shows the predictability with the described framework. Given the initial position of the object, placing the object at different positions is desired. The object is hit with different pre-impact fluxes by the robot and the distance moved by the object is compared. The initial position of the object is $[0.5m, 0.3m, 0.3m]$ and it is hit in the X direction. The motion data is shown in Fig. 8b. The displacement of the object follows almost a linear trend. The deviations from the linear trend can be understood by analyzing the scenario according to the achieved pre-impact flux. This depends on the tuning of the controllers and the path taken by the robot. Hence, the fluctuation in the achieved flux leads to some deviation in the distances moved by the object.

*2) **Achieving same pre-impact flux**:* Here we show that the robot can be controlled to have the same directional pre-impact flux at different task space positions, if feasible. This shows that the robot can interact with objects located at different positions and manipulate them in the same manner. For this experiment, the box is placed in 16 different positions in the Y-Z plane, sampled in the range $y \in [-0.4, -0.2] \cup [0.2, 0.4], z \in [0.2, 0.6], x = 0.5$ and the robot's end effector is initialized at 12 random points in the Y-Z plane, sampled in the range $y \in [-0.4, 0.4], z \in [0.2, 0.6], x = -0.2$. This results in 192 trajectories from multiple initial to multiple final positions with the same desired pre-impact flux. The hitting direction at the final positions is $\hat{h} = [1, 0, 0]$. The desired directional hitting flux $\phi_{des} = 0.6$ m/s so that it is known that such a quantity is feasible. [5]

The mean of the final achieved directional flux values is 0.579 m/s and the standard deviation is 0.038 m/s. Out of 192 trajectories, we have 10 trajectories that reach a directional hitting flux of $<$ 0.5 m/s. The distribution of the data shows that the directional hitting flux for the trajectories required to reach higher Z values is far from the desired directional hitting flux. This is because the inertia and velocity manipulability of the system are conflicting properties of the system. Since we want the directional inertia of the robot to align with the hitting direction, the physical geometry of the robot restricts the alignment of the robot in Y axis while reaching high Z values. This can be seen in Fig. 8c. The points in blue are those ones that show that the desired flux is not achieved at the desired final position.

Although on average, we achieve the directional flux, the following factors affect the final achieved hitting flux:

- Initial configuration of the robot
- The DS for the motion generation

Hence, for the experiments on the real robot, it is advised to have initial configurations where the inertia matrix is not far (in terms of stein distance) on the inertial manifold from the desired inertia.

[5]Since, optimization for the values of directional inertia and hitting velocity remains an open question, IK is used to provide an idea of achievable configurations at the desired end effector position. Through this self- motion manifold [40] samples, we have an idea of achievable directional inertia values.

C. Achieving Desired Inertia with Desired Position:

For this experiment in simulation, for different joint configurations, we store the inertia values since at a given end effector position, we can have multiple possible joint configurations, and hence, multiple possible inertia matrices (using IK and self motion manifold). We choose a random initial position and configuration, and a random final desired position and desired inertia matrix at that position. For the purpose of this experiment, one can choose the motion to be governed by the DS as in Eq. 22 or a simple stable linear DS such as,

$$f(\chi) = A(\chi - \chi^*)$$

where $A \prec 0$, and the desired translational inertia is

$$\Lambda^* = \begin{bmatrix} 3.2 & 3.07 & -0.32 \\ 3.07 & 3.0 & -0.31 \\ -0.32 & -0.31 & 0.03 \end{bmatrix} .$$

The robot then moves under the Inertia QP controller (Eq. 24).

TABLE II: Quantitative comparison of the final joint configurations for different controllers (also check Fig. 7c)

No.	final joint configuration	g
IK	(-0.96, 2.06, -0.17, -2.06, -1.63, 1.06, 2.97)	4.04
Inertia QP	(-1.94, 0.89, -1.34, -1.90, 0.54, 0.89, 1.56)	3.75

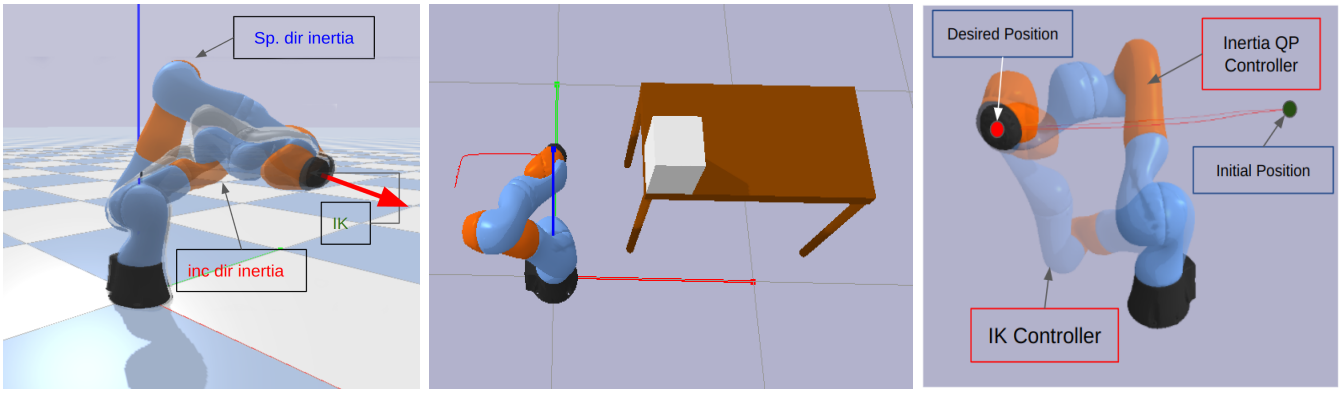
Fig. 7c shows the different configurations of the robot, given the same path it takes (in red) while getting closer to the desired inertia matrix. Since the QP is local optimization, the solution is not guaranteed to be the global solution. Hence, the robot can get stuck in a local minima or the joint limits.

IX. Robot Experiments

The experimental setup consists of a KUKA LBR iiwa 7 robot arm hitting a box (shape: cuboid) of known total mass on a planar table. The motion of the box is tracked using an Optitrack system that streams at 250 Hz using Prime 17W cameras. The DS described in Section V generates motion for the end effector of the robot controlled by the different strategies in Section VI . We use different kinds of objects that are used in industry and show the relevance of this paper in industries. The objects used are cardboard boxes of more or less uniform mass distribution and open boxes containing different objects, thus with a non uniform mass distribution as shown in Fig. 9a.

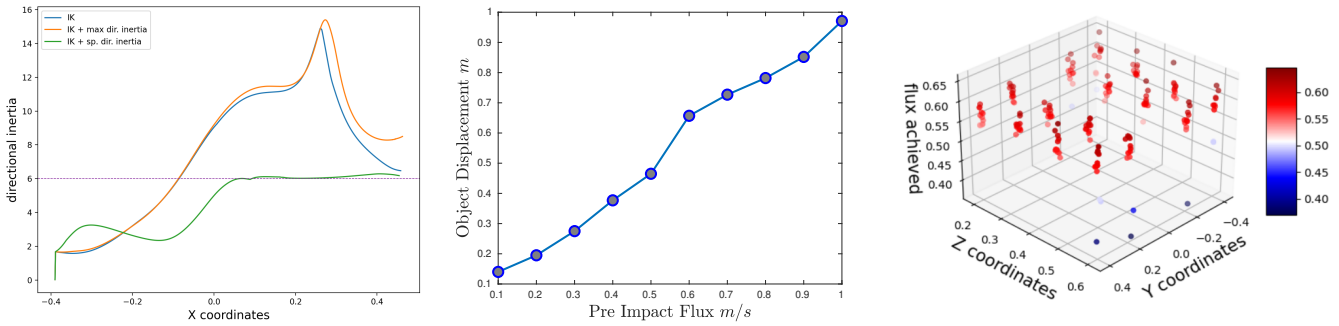
TABLE III: Objects / Boxes used in the experiments, refer to Fig. 9a

Object	1	2	3	4
Mass (kg)	0.34	0.95	0.4	2.0
Mass distribution	uniform	non-uniform	uniform	uniform
size (cm^3)	18x19x19	31x23x21	26x25x23	26x26x27



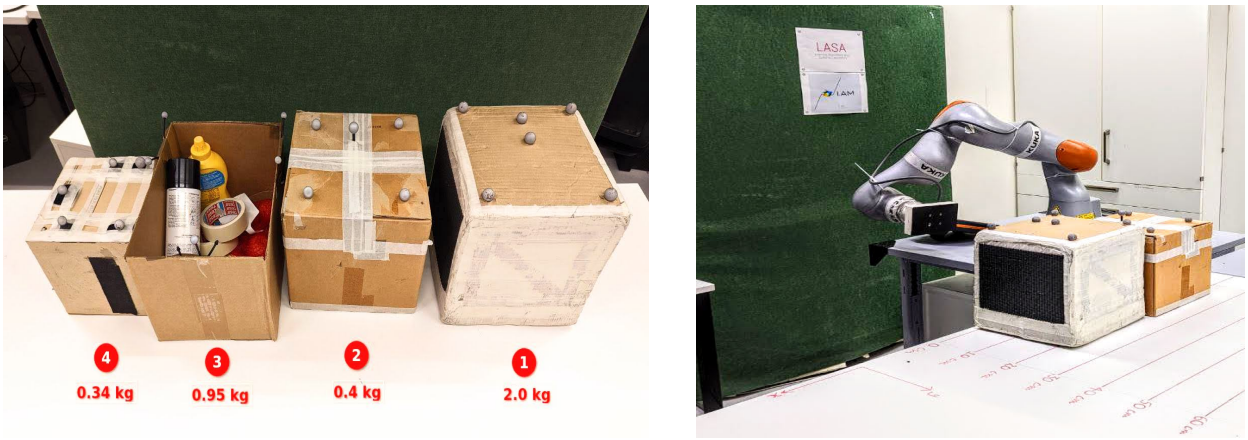
(a) (b) (c)

Fig. 7: Fig. (a): The three different control systems lead to different joint configurations at the desired final position. The IK controller leads to the joint configuration closest to the initial configuration. With the increasing directional inertia controller, the robot's configuration is aligned with the desired direction. Since the specific directional inertia is lower than the maximum achievable in the desired direction and the directional inertia achieved by the inverse kinematics controller, the configuration achieved with the specific inertia controller is least aligned with the hitting direction; Fig. (b): The simulation setup is shown - The robot follows the dynamical system to generate a pre-impact flux to interact with the object on a table; Fig. (c) The robot achieves different configurations with two different controllers (IK and inertia QP controller) while following the same path - The inertia QP controller tries to attain the desired inertia matrix subject to robot constraints.



(a) (b) (c) Achieved directional hitting flux

Fig. 8: Fig. (a): Quantitative comparison of the controllers - The three curves show the different directional inertia of the robot while following a trajectory with three different controllers. The green curve shows the controller trying to maintain the directional inertia along the trajectory equal to 6 kg while following the trajectory; Fig. (b): Displacement of the object with different pre-impact fluxes of the robot; Fig. (c): Final achieved directional hitting flux for the final position of the end effector in the 192 simulated trajectories.



(a) Objects used in the experiment (b) Experimental Setup

Fig. 9: Fig. (a) shows the different boxes used in the experiments with different properties such as size, masses, and mass distribution as indicated. Fig. (b) shows the experimental setup used: A KUKA LBR iiwa 7 arm that hits or intercepts the boxes moving on the table. The table is marked to better access the distance moved by the object.

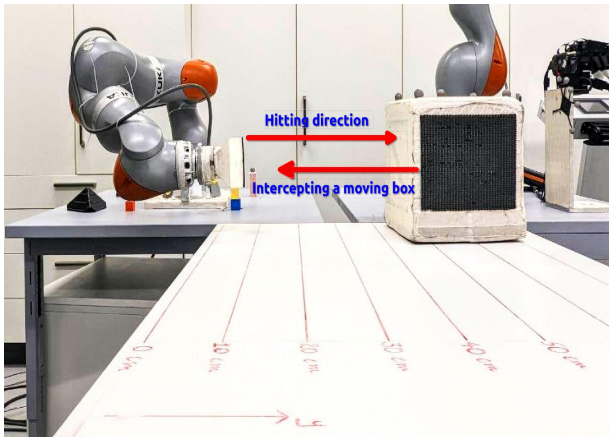


Fig. 10: Experiment for object hitting and interception - the arrows how the different direction of motion of the robot during hitting an object and intercepting the object.

We also compare the performance of our approach to a control system based solely on velocity modulation, as proposed in [22] (mentioned later as *high speed impact controller*). In the experiments, we use a simple block as an end effector that does not significantly extend the length of the robot and thus, the physical limits of the robot as in [22]. This allows us to see significant changes in the configuration of the robot when controlling for its inertia.

High Speed Impact Controller: In this method [22], we generate the motion using the DS and modulate the end effector speed using a scaling factor.

$$\dot{\chi}^*_{max} = \gamma \hat{h}, \quad \gamma \in \mathbb{R}_+$$

where γ is a scaling factor that can be given as an input to the system and $\hat{h}$ is the hitting direction. In the null space of the first task, the robot is trying to achieve the anchor configuration (q_m) obtained by manipulability maximization (refer Eq. 14).
Hence the joint velocities passed as an input to the controller can be written as:

$$\dot{q} = J^{\dagger}(q) f(\chi) + \beta N(q_m - q), \quad \beta \in \mathbb{R}_+ \ . \tag{25}$$

We create three scenarios that confirm the contributions of the paper described in Section I, and compare the performance of our method to the high-speed impact controller (Eq. 25). The difference in performance that we expect is hypothesized in the respective scenarios and then the results are discussed.

A. *Scenario 1 - Hitting stationary boxes*

We systematically test the hitting action on boxes. First, we see the difference between pushing and releasing, as compared to hitting the object. Hitting the object at high speed should impart larger motion to the object, as opposed to pushing and releasing at high speed. This is confirmed experimentally. We then compare the difference between using the high-speed impact controller and the directional inertia controller with the hitting flux-based DS. The repeatability of our method is also shown in these experiments.

Hypothesis - Since our method considers the dynamic properties of the objects and the robot, hitting different objects with the same pre-impact flux should lead the different objects to move similar distances (assuming the coefficient of restitution is similar and so are the friction coefficients between the objects and the table) whereas the same hitting speed will make different objects move differently.

1) Pushing and releasing: Here we compare the motion of the object after being pushed and hit with the same motion of the robot (Fig. 11). The expectation is that the object can achieve higher velocities than the robot velocity at the time of the hit as it is lower in mass than the robot, while in push and release it will have the same speed as the robot at the time of release. This should allow the hit object to move faster and more distance. The robot achieves the same end effector speed and directional flux during the hitting and the pushing tasks as shown in Fig. 11 To keep the experiment setup consistent, the motion of the robot remains the same. The object is released at the same point in the push and release, where it is hit in the hitting scenario. The accompanying supplemental video shows the two different processes.

TABLE IV: Object Mass = 0.4 kg, Pushing / Hitting speed = 0.95 m/s, Flux = 0.85 m/s, 10 repetitions

Push distance	Hit distance
0.37 m	0.54 m

From the Table. IV we see that the object moves on average larger distance when it's hit than pushed at a high speed.

2) Different objects and same hitting velocity: Table V shows the distances moved by different objects when they are hit with the same end effector speed (controlled using the high-speed impact controller). Since this velocity is limited by the torque limits of the robot, this is the largest distance the respective objects move. Since the distances covered by the object when the robot hits them with the same speed are different, this shows that the velocity of hit is not the only important factor to consider. To control the motion of an object that is similar in mass to the directional inertia of the robot (e.g., object 1 of mass 2 kg), we must include the masses of object and the robot in our calculation.

TABLE V: Distances moved by objects 1 and 2 at highest achieved velocity and highest achieved flux (experiment repeated 10 times each)

Highest speed achieved (average)	Box mass	Distance covered (average)
1.2 m/s	0.4 kg	0.65 m
1.2 m/s	2.0 kg	0.32 m

3) Different objects subject to hitting flux: Hypothesis - Since our method considers the dynamic properties of the

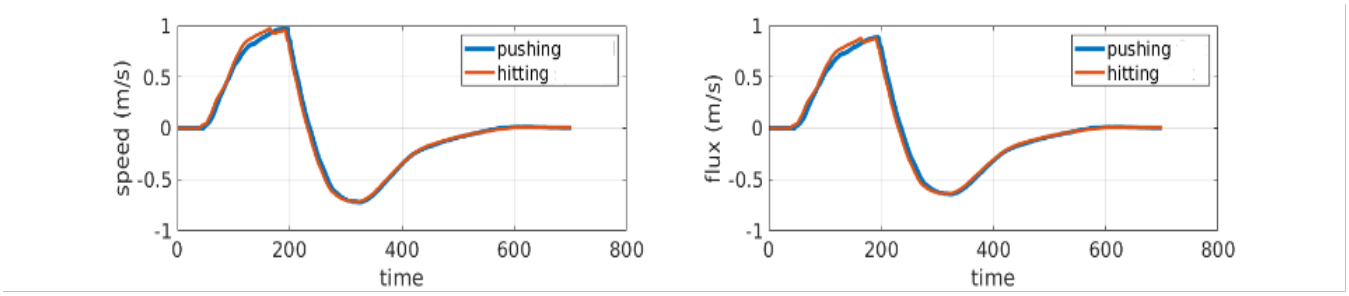


Fig. 11: The robot velocities and hitting fluxes during pushing and hitting the object

objects and the robot, hitting different objects with the same pre-impact flux should lead the different objects to move similar distances (assuming the coefficient of restitution is similar and so are the friction coefficients between the objects and the table) whereas the same hitting speed will make different objects move differently.

Fig. 13 shows the effect when object 1 (2.0 kg) and object 2 (0.4 kg) are hit with multiple fluxes. For object 2, we reach the velocity limit of the robot for hitting flux $\sim$ 0.6 m/s. Hence we cannot compare the motion of both objects after this flux value.

First we compare the difference between the desired flux and the achieved flux values. According to Table IX, we see slight errors in achieving the desired flux. This is because the directional inertia of the robot is being controlled in the null space of the first task, which is to achieve a desired velocity. In the total of 110 different hitting experiments, the RMSE (Root Mean Squared Error) of the flux achieved is 0.008 m/s, which in relative terms is 0.004 or 0.4% error.

Analysis of motion of different objects being subject to same hitting flux: Fig. 13 shows similarity in the motion of both objects when subjected to similar flux. Fig. 12 shows the mean and variance in the motion of objects after being hit (flux values used = 0.3 and 0.6 m/s for this plot). For a similar achieved flux, we have, on average, similar distances moved by objects that are similar in material. The differences in motion may also be affected by different friction and restitution parameters, the identification of which is another research problem.

Relation between object motion and hitting flux: Having seen the repeatability of the control of the robot to produce similar effects on similar objects with different masses, in theory scaling the hitting flux should result in a proportional change in the distances moved by the objects. From Table IX and Fig. 12, we see a small discrepancy. Double the hitting flux leads to slightly more than double the distance covered (for both objects). The linear effect on the motion of the object is visible through the plot in Fig. 13, although after a threshold of the hitting flux, the slope of the linear behavior changes. This requires further analysis, but one explanation is the dependence of friction coefficient on the speed of sliding of the object [41] and considering the stick-slip phenomenon [42]. The low flux values do not provide enough energy transfer to overcome the spontaneous jerking motion when the object starts sliding on the table. This creates a research problem of identifying the initial non-linearity between the hitting flux and the distance traveled by the object given the restitution and friction coefficients.

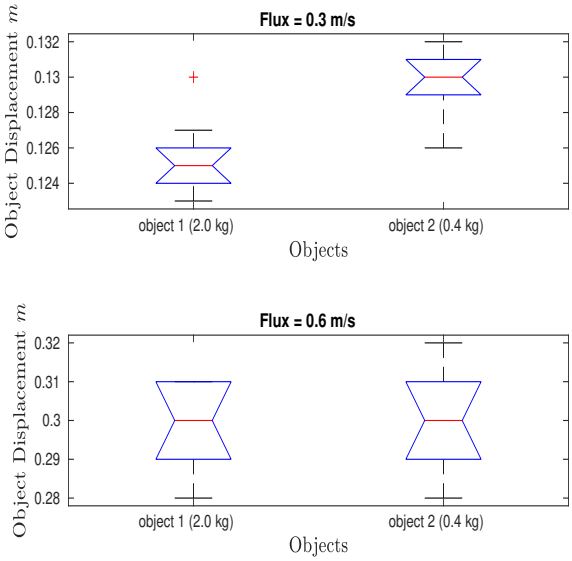


Fig. 12: The plots show the variability in the final position achieved by two different objects for two different flux values. For a similar achieved flux value, the displacement of two different boxes is similar.

4) Sensitivity analysis to the known mass: To produce similar post-impact object motion, we need to know the mass of the object. In the industrial setting, there can be small fluctuations in the masses of similar objects, or fluctuation in the known mass of an object. The estimation techniques can also produce errors in the known values of the masses. Thus, the desired hitting flux value may not lead to the motion of the object that is desired. Here, we analyze the sensitivity of the post-impact object motion on errors in the known mass.

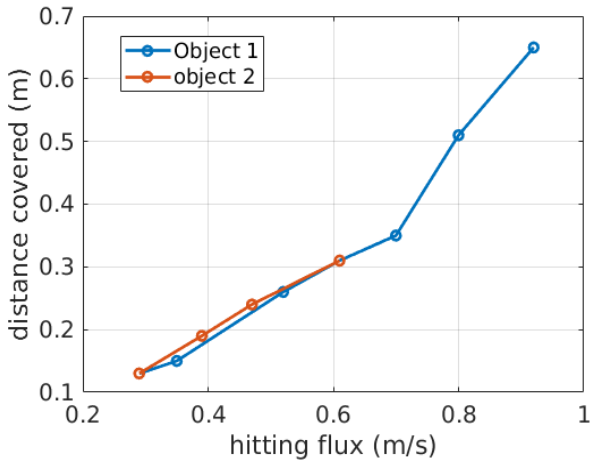


Fig. 13: Average distance covered by the two objects on being subjected to various hitting fluxes. Both objects behave similarly flow small flux values, which aren't the same motion for the robot. For high flux values for object 1, it deviates from the original linear relation, which can be attributed to velocity dependent friction behavior, but needs further research.

TABLE VI: Hitting Flux = 0.85 m/s, Actual mass = 0.4 kg, Actual distance = 0.59 m

Box mass known (kg)	Achieved distance (m)	Theoretical error (Rel RMSE)	Experimental error (Rel RMSE)
0.32	0.58	(-)0.016	(-)0.01
0.36	0.586	(-)0.008	(-)0.006
0.44	0.594	0.008	0.008
0.48	0.598	0.016	0.015

TABLE VII: Hitting Flux = 0.48 m/s, Actual mass = 2.0 kg, Actual distance = 0.26 m

Box mass known (kg)	Achieved distance (m)	Theoretical error (Rel RMSE)	Experimental error (Rel RMSE)
1.6	0.195	(-)0.06	(-)0.057
1.8	0.202	(-)0.03	(-)0.023
2.2	0.263	0.03	0.01
2.4	0.283	0.06	0.08

For a given desired flux, the error in known mass of the object changes the desired pre-impact speed of the end effector of the robot.
Theoretical sensitivity analysis (Relative RMSE):

$$\frac{\Delta \dot{\chi}_o^+}{\phi} = \left(\frac{\Delta m_o}{\lambda + m_o}\right) \tag{26}$$

where ϕ is the commanded flux, m_o is the correct mass and Δm_o is the error in the known mass value. Check Section XI for the derivation of the relative sensitivity. The theoretical and experimental relative RMSE errors can be found in Table VI and VII. The $(-)$ indicates that the object covers less distance than what is expected. We see from the Relative RMSE in Table VI and VII that difference in experimental errors and predicted errors is low. Hence, the physics model is quite robust to errors in the estimated mass of the object.

B. *Scenario 3 - Intercepting boxes:*

Instead of a robot imparting motion, a similar impact is generated when the robot stops an object by intercepting it in its trajectory. This allows for human - robot interactions with humans passing the objects to robots for further tasks (Fig. 10) and having low end effector velocities of the robot post-impact is desired. We compare the end effector speed of the robot with the two controllers (Eq. 18, Eq. 25).
Hypothesis - Higher intercepting inertia should lead to lower end effector velocities, and will lead to more precise human-robot interaction.

Fig. 14 and 15 show the difference in the post impact robot speed when a human passes an object of mass 2.0 kg moving in the y direction. When controlled with the increasing inertia controller, the robot has on average almost half the post impact speed (0.072 m/s) compared to the robot that intercepts a box while not changing its inertia or with the high speed impact controller (0.129 m/s). For faster and more precise human robot interaction, and then continuous task performance by the robot, it is desirable for the robot to not have high post-impact end effector velocities. This can be achieved by controlling the robot's inertia. Both intercepting and hitting objects can be controlled using the same control system that modulates the velocity and directional inertia of the robot.

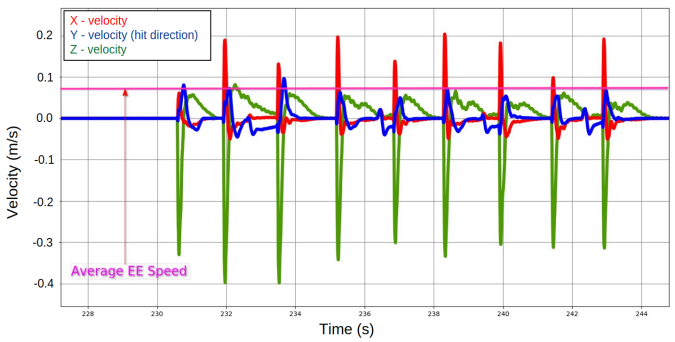
C. *Motion in the Inertial Manifold:*

The robot is tested to achieve a configuration that minimizes the error between the desired inertia matrix (Λ^*) and the robot's inertia, while achieving the desired position for the end effector. Here

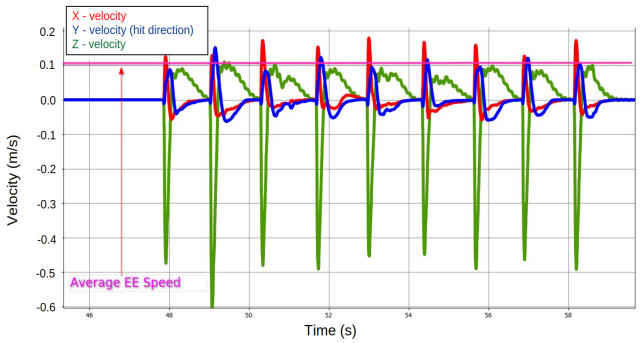
$$\Lambda^* = \begin{bmatrix} 4.0 & 1.0 & 3.0 \\ 3.0 & 3.0 & 2.0 \\ 3.0 & 2.0 & 3.0 \end{bmatrix} .$$

We compare the performance of the inverse kinematics (IK) controller (Eq. 13) and Inertia QP controller (Eq. 24). The desired inertia is a randomly designed inertia matrix to test how close the robot can get to the desired inertia matrix. The robot moves from a given initial position $[0.3, -0.2, 0.5]$ (in m) to a final desired position $[0.3, 0.3, 0.5]$ (in m) while being controlled by the above two controllers. We measure the Stein distance between the desired inertia and robot's inertia throughout its motion and compare the two controllers.

Fig. 16 shows the evolution of the Stein distance between the current inertia and the desired inertia while moving toward the desired end effector position. We see that, the IK controller leads to an increase of $g(q)$, from 1.28 to 1.4, and the final inertia is farther away than the desired inertia matrix on the inertial manifold. With the Inertia QP controller, the distance between the robot's inertia and the desired inertia decreases along the path, from 1.28 to 1.13. Fig. 17 shows the



(a) Directional Inertia Controller



(b) High Speed Impact Controller

Fig. 14: The two different graphs show the end effector velocity of the robot after intercepting a moving object with different inertias. In blue, we see the velocities of the end effector in y-direction (where the inertia is chosen to be high in value). In Fig. (a) the robot is controlled using increasing inertia controller (Eq, 18) and in Fig. (b), we have high speed impact controller (Eq. 25)

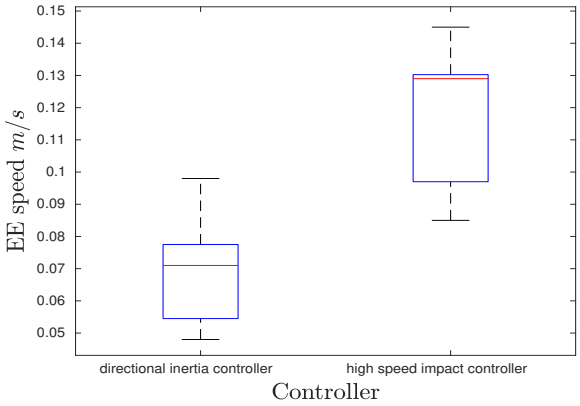


Fig. 15: Box plot showing the post-impact end effector speed after intercepting or being hit by an object of mass 2.0 kg

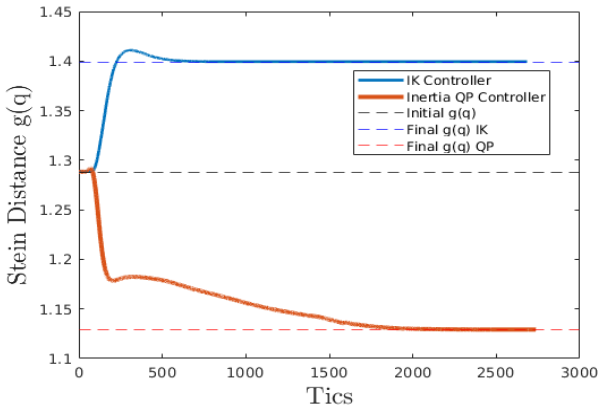


Fig. 16: The graph shows the evolution of the Stein distance while reaching a desired end effector position.

initial configuration of the robot and the final configurations of the robot achieved through IK controller and the inertia QP controller. Since QP solves a problem locally, the solution is susceptible to be a local minimum or reach joint limits. Hence, the final joint configuration achieves the task inertia that is closest to the desired inertia but depends on the initial configuration of the robot. This opens a research problem of designing motion planners that consider the dynamic properties of the robot, are stable and achieve the global minimum solution.

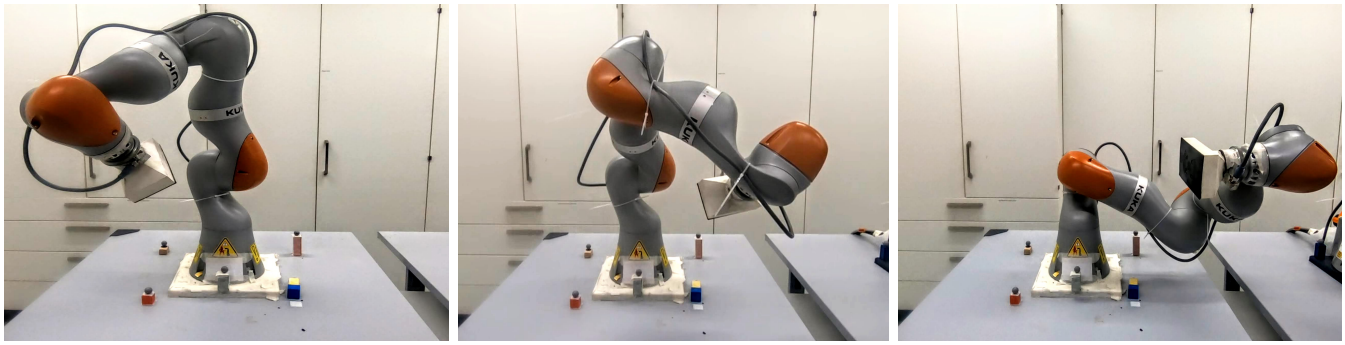
D. Further applications:

Further applications of this process for handling objects not suitable for suction cups, such as open boxes with uneven mass distribution and collecting different sized objects are shown in the video attached.

X. Discussion and Future Work

In this paper, we propose a metric, *hitting flux* (Φ), which we control to enable a manipulator to interact with the environment focusing on the hitting or striking motion between the robot and the environment. This is an example of dynamic manipulation because it considers the inertial properties of both, the environment and the robot. This provides the robot with the skill to create motions that have similar effects on the environment with different inertial properties. This is achieved using a DS-based motion planner that scales velocity according to the current value of the robot's directional flux and an inertia based controller that tries to track directional inertia values by exploiting the redundancy of the manipulator. Furthermore, a QP-based controller is proposed to reach a target by following a dynamical system and minimize the robot's translational inertia matrix to a desired inertia matrix under the joint position and velocity constraints.

This approach relies on assuming that the directional inertia and end effector speed are achievable at a certain end effector position. Thus, motion planning by the DS considers the current directional inertia value and assumes that the path followed by the end effector leads to a feasible directional inertia value. The redundancy of the robot allows it to achieve multiple different inertias at the same end effector position. Motion planning and achieving the desired inertia matrix are not independent tasks and they can be adversarial to each other if not planned properly. Thus, the desired directional hitting flux may not be achievable on a path if the velocity and the directional inertia cannot be achieved on the planned



(a) Initial position of the robot (b) Motion with IK Controller (c) Motion with Inertia QP Controller

Fig. 17: Fig. (a): This is the starting configuration of the robot; Figure (b): Final configuration of the robot under the IK Controller. This configuration is moving away from the desired inertia matrix as showing in Fig. 16; Fig. (c): Final configuration of the robot under the Inertia QP Controller. This configuration is closer to the desired inertia configuration (Fig. 16) while achieving the final position

path. This will be tackled in the future. Future work entails the planning of robot motion or DSs to achieve the desired inertia (or another SPD property) matrix at a desired task space position. The main purpose of using a dynamical system based motion planner is to have a closed form solution that plans the trajectory depending on where the robot and the object is. Further research includes understanding how can we include dynamics of the robots such as inertia in the DS based motion plan. Other methods that can be applied to get out of the local minima of the inertia matrix is to have Model Predictive Control, both analytical or sampling based. The downside for such method is high computational cost [43] and still some potential local minima. Another approach could be to have multiple motion plans (Graph based motion planning) that plan the motion according to achievable inertia matrices. But these methods will suffer from re-planning in the event of disturbances. This helps in the intuitive programming of robots, where the task at hand is programmed using the task space properties of the robot and understanding how humans perform a certain task.

The repeatability of the robotic motion to interact with the environment through intentional impacts must be adaptive to interact predictably with other objects. This includes learning properties such as object mass, object inertia, friction parameters, coefficient of restitution, and residual errors in the motion equations and real life motion. Even with the approximate knowledge of these parameters, the DS must be adaptive to not just the motion of the robot, but also the differences in the actual and predicted environment motion to enable desired environment change. This poses the research question of learning not just object properties, but their behavior after being subjected to impulsive forces. We intend to use the hitting flux formulation along with data driven motion models estimate the object and collision properties and adapt the DS for motion generation.

Acknowledgments

This work was supported by the Research Project I.AM. through the European Union H2020 program under GA 871899. The authors would like to thank Dr. Farshad Khadivar for his help in robot experiments.

XI. Appendix

A. Hitting Flux Derivation

Here we derive the expression of Hitting Flux $\Phi \in \mathbb{R}^3$. From the principle of conservation of momentum during the impact, and the definition of restitution [32] along the impact normal ($\hat{h}$), we have:

$$\Lambda\dot{\chi}_e^- + M_o\dot{\chi}_o^- = \Lambda\dot{\chi}_e^+ + M_o\dot{\chi}_o^+ \tag{27}$$

$$\epsilon\ \hat{h}^T(\dot{\chi}_e^- - \dot{\chi}_o^-) = \hat{h}^T(\dot{\chi}_o^+ - \dot{\chi}_e^+)\ . \tag{28}$$

$\epsilon \in [0,1]$ is the coefficient of restitution in hitting direction, $\hat{h}$. Using the assumption that $\dot{\chi}_e^+ \simeq \|\dot{\chi}_e^+\|\,\hat{h}$, we can write

$$E(\dot{\chi}_e^- - \dot{\chi}_o^-) = \dot{\chi}_o^+ - \dot{\chi}_e^+ \tag{29}$$

$$E = \begin{bmatrix} \hat{h}^T \\ \hat{h}_{\perp_1}^T \\ \hat{h}_{\perp_2}^T \end{bmatrix} \begin{bmatrix} \epsilon & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{h} & \hat{h}_{\perp_1} & \hat{h}_{\perp_2} \end{bmatrix} \tag{30}$$

$\hat{h}, \hat{h}_{\perp_1}, \hat{h}_{\perp_1}$ form an orthogonal basis. From Equations 27 and 29, we have the pre-impact end effector velocity expressed in terms of the pre- and post- impact velocities of the object:

$$\dot{\chi}_e^- = [\Lambda(I+E)]^{-1}[(\Lambda + M_o)\dot{\chi}_o^+ + (\Lambda E - M_o)\dot{\chi}_o^-]\ . \tag{31}$$

Simplifying Eq. 31 further and using $\dot{\chi}_o^- = 0$ we have:

$$\dot{\chi}_e^- = (I+E)^{-1}(I + \Lambda^{-1}M_o)\dot{\chi}_o^+\ .$$

Let $(I+E) = C$, since it is constant for a particular collision pair. The above equations become the following:

$$(I + \Lambda^{-1}M_o)^{-1}C\dot{\chi}_e^- = \dot{\chi}_o^+\ . \tag{32}$$

Reproducing similar post-impact conditions for the object translates to achieving a desired post-impact object velocity, $\dot{\chi}_o^+$. Eq. 32 depicts the requirements for the repeatable collision and interaction with the environment. Given the post-impact object behavior ($\dot{\chi}_o^+$) and mass of the object M_o, we need to control for $(I + \Lambda^{-1}M_o)^{-1}C\dot{\chi}_e^-$. In this paper, we refer to $(I + \Lambda^{-1}M_o)^{-1}C\dot{\chi}_e^-$ as Φ or **hitting flux**.

$$\Phi = (I + \Lambda^{-1}M_o)^{-1}C\dot{\chi}_e^-\ . \tag{33}$$

B. Mathematical tuning of the Hitting Dynamical System:

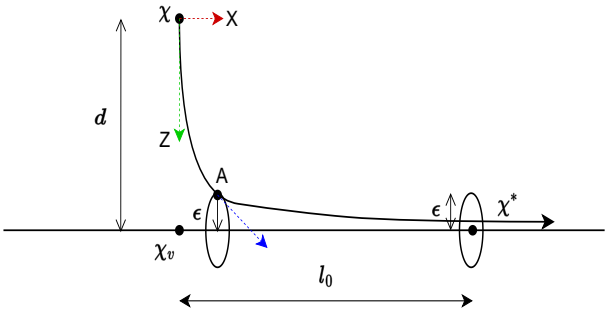


Fig. 18: Diagram showing the motion of the robot end effector's motion from the initial to the desired hitting position. The time constraint to reach point A is less than time to cover the distance l_0 is imposed by Eq. 34.

Here we explain how to tune the parameter σ in the hitting dynamical system given by Eq. 8, 9. The following analysis is performed in a vertical plane w.l.o.g. because the hitting motion generated by the dynamical system is planar and mathematical analysis can be done for any motion by applying a suitable rotation matrix. Let the dynamical system be required to pass through the hitting point within an error margin of ϵ. l and d are the distances between the end-effector and the object in the X and Z direction respectively, with l_0, and d_0 being the initial distance between the end effector and the object in the respective directions. v represents the desired hitting velocity. From the DS equations 8, the equations of motions in the X and Z directions can be written as follows:

$$\dot{l} = v \exp^{(-d/\sigma^2)}$$
$$\dot{d} = -(1 - \exp^{(-d/\sigma^2)})kd$$

where $-k$, $k > 0 \in \mathbb{R}$ is the eigenvalue of the gain matrix in Eq. 8 in Z direction. We want the robot end effector to reach the hitting radius of ϵ before it reaches the desired hitting position. The slowest Z dynamics are:

$$\dot{d} = -(1 - \exp^{(-\epsilon/\sigma^2)})kd \quad .$$

The solution for this ODE is as follows:

$$d = d_0 \exp^{-(1-\exp^{(-\epsilon/\sigma^2)})kt} \quad .$$

The fastest dynamics in the X axis would be:

$$\dot{l} = v \exp^{(-\epsilon/\sigma^2)} \quad .$$

The time to reach ϵ in the Z axis should be less than the time to reach l_0 in the X axis.

$$\frac{l_0}{v \exp^{(-\epsilon/\sigma^2)}} > \frac{ln(d_0/\epsilon)}{k(1 - \exp^{(-\epsilon/\sigma^2)})}$$
$$\exp^{(-\epsilon/\sigma^2)} < \frac{1}{1 + \frac{v}{kl_0} ln(\frac{d_0}{\epsilon})}$$
$$\implies \sigma < \sqrt{\frac{\epsilon}{ln(1 + \frac{v}{kl_0} ln(\frac{d_0}{\epsilon}))}} \quad . \tag{34}$$

Eq. 34 provide the constraints for the robot end effector to align with the hitting path before actually hitting the object.

C. Inertia Gradient:

The inertia derivative is calculated numerically as shown in Alg. 1 and the algorithm is parallelizable:

Algorithm 1: Inertia Gradient

Input : Joint position - q
Output: $\nabla_q \Lambda(q)$
$dq = 0.001$
$\nabla_q \Lambda(q) \leftarrow zeros(6, 6, size(q))$
for $i \leftarrow 0$ **to** $size(q)$ **do**
  $q^{'} = q$
  $q^{'}[i] = q[i] + dq$
  $\nabla_q \Lambda(q)[:, :, i] \leftarrow (\Lambda(q) - \Lambda(q + dq))/dq$
end
return $\nabla_q \Lambda(q)$

D. Stein Distance Derivative for Inertia:

From Eq. 23, we have the distance between Λ^* and Λ_q

$$g(q) = log(det(\frac{\Lambda^* + \Lambda_q}{2})) - \frac{1}{2} log(det(\Lambda^* \Lambda_q))$$

$g(q) \in \mathbb{R}$, and we require $\dot{g}(q)$.

$$\dot{g}(q) = \nabla_q g(q)^T \dot{q} \tag{35}$$

$$\nabla_q g(q) = \nabla_\Lambda g(q) \nabla_q \Lambda_q$$
$$\nabla_q g(q) = (\frac{1}{2}(\frac{\Lambda^* + \Lambda_q}{2})^{-1} - \frac{1}{2}(\Lambda_q)^{-1}) : \nabla_q \Lambda_q$$
where, $A : B = Tr(B^T A)$ for matrices A, B
using properties, $\frac{\partial}{\partial \Lambda} ln\,|det(\Lambda)| = (\Lambda^{-1})^T$ and, $\Lambda^T = \Lambda$
$$\nabla_q g(q) = ((\Lambda^* + \Lambda_q)^{-1} - \frac{1}{2}(\Lambda_q)^{-1}) : \nabla_q \Lambda_q \quad . \tag{36}$$

For the derivatives used above, refer to [44].

The derivative can be calculated numerically as well.

E. Sensitivity Analysis:

We discuss the change in the motion of the object induced if the initial hypothesis on the mass of the object is wrong. Assume the actual mass of the object is m_o and the hypothetical mass of the box to be $m_k = m_o \pm \Delta m_o$. ϕ is the desired hitting flux, through which we can find the hitting speed of the end effector by $\phi = \frac{\lambda}{\lambda + m_k} \dot{\chi}_e^-$:
Theoretical sensitivity analysis:

$$\dot{\chi}_o^+ = [1 + \lambda^{-1} m_o]^{-1} \dot{\chi}_e^-$$
$$= (\frac{\lambda}{\lambda + m_o})(\frac{\lambda + m_k}{\lambda})\phi = \frac{\lambda + m_k}{\lambda + m_o}\phi \tag{37}$$

An estimate of the mass larger than the actual mass should hence lead to a larger speed induced on the object, hence leading the object to cover a longer distance.

$$\begin{aligned}\Delta\dot{\chi}_o^+ &= \frac{\lambda + m_k}{\lambda + m_o}\phi - \phi \\ &= (\frac{\Delta m_o}{\lambda + m_o})\phi \ .\end{aligned} \tag{38}$$

Relative RMSE error is given by

$$\frac{\Delta\dot{\chi}_o^+}{\phi} = (\frac{\Delta m_o}{\lambda + m_o}) \tag{39}$$

F. Results for experiments in simulation and on real robot:

TABLE VIII: Object Motion Data in Simulation - The simulation has differences in terms of the behaviour of the world and shows a linear relation between the hitting flux and motion of the object

No.	**Pre-impact Flux**	**Final Position** (m)	**Displacement** (m)
1	0.1	[0.640, 0.3, 0.5]	0.140
2	0.2	[0.695, 0.3, 0.5]	0.195
3	0.3	[0.775, 0.3, 0.5]	0.275
4	0.4	[0.877, 0.3, 0.5]	0.377
5	0.5	[0.965, 0.3, 0.5]	0.465
6	0.6	[1.157, 0.3, 0.5]	0.657
7	0.7	[1.227, 0.3, 0.5]	0.727
8	0.8	[1.282, 0.3, 0.5]	0.782
9	0.9	[1.353, 0.3, 0.5]	0.852
10	1.0	[1.471, 0.3, 0.5]	0.971

TABLE IX: Average distances moved by objects 1 and 2 subject to multiple fluxes (experiment repeated 10 times each)

Hitting Flux desired (m/s)	Hitting Flux achieved(m/s)	Distance covered 0.4 kg (m)	Distance covered 2.0 kg (m)
0.3	0.29	0.125	0.13
0.4	0.39	0.19	0.18
0.5	0.52	0.20	0.24
0.6	0.61	0.31	0.31
0.7	0.7	0.35	-
0.8	0.8	0.51	-
0.9	0.92	0.65	-

REFERENCES

[1] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing," *Frontiers in Robotics and AI*, vol. 7, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2020.00008

[2] D. Ma and A. Rodriguez, "Friction variability in auto-collected dataset of planar pushing: Data-collection bias and anisotropic friction," in *IROS*, 2018.

[3] M. Bauza and A. Rodriguez, "A probabilistic data-driven model for planar pushing," in *ICRA*, 2017.

[4] M. Bombile and A. Billard, "Dual-arm control for coordinated fast grabbing and tossing of an object: Proposing a new approach," *IEEE Robotics and Automation Magazine*, vol. 29, no. 3, pp. 127–138, 2022.

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

[6] S. Sra, "A new metric on the manifold of kernel matrices with application to matrix geometric means," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[7] N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon, "Geometry-aware manipulability learning, tracking, and transfer," *The International Journal of Robotics Research*, vol. 40, no. 2-3, pp. 624–650, 2021, pMID: 33994629.

[8] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," 2018.

[9] F. Abu-Dakka and V. Kyrki, "Geometry-aware dynamic movement primitives," 01 2020.

[10] M. Bauza and A. Rodriguez, "A probabilistic data-driven model for planar pushing," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3008–3015.

[11] N. F. KT. Yu, M. Bauza and A. Rodriguez, "More than a million ways to be pushed: A high-fidelity experimental dataset of planar pushing," in *IROS*. IEEE, 2016, pp. 30–37.

[12] H. Kitano, M. Asada, I. Noda, and H. Matsubara, "Robocup: robot world cup," *IEEE Robotics and Automation Magazine*, vol. 5, no. 3, pp. 30–36, 1998.

[13] J. Tebbe, Y. Gao, M. Sastre-Rienietz, and A. Zell, "A table tennis robot system using an industrial kuka robot arm," in *Pattern Recognition*, T. Brox, A. Bruhn, and M. Fritz, Eds. Cham: Springer International Publishing, 2019, pp. 33–45.

[14] S. M. Khansari-Zadeh, K. Kronander, and A. Billard, "Learning to play minigolf: A dynamical system-based approach," *Advanced Robotics*, vol. 26, no. 17, pp. 27. 1967–1993, 2012. [Online]. Available: http://infoscience.epfl.ch/record/181052

[15] Y.-B. Jia, M. Gardner, and X. Mu, "Batting an in-flight object to the target," *The International Journal of Robotics Research*, vol. 38, no. 4, pp. 451–485, 2019. [Online]. Available: https://doi.org/10.1177/0278364918817116

[16] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 5113–5120.

[17] R. L. Anderson, *A Robot Ping-Pong Player: Experiment in Real-Time Intelligent Control*. Cambridge, MA, USA: MIT Press, 1988.

[18] J. Billingsley, "Robot ping pong," *Practical Computing*, vol. 6, no. 5, 1983.

[19] Y. Huang, B. Schölkopf, and J. Peters, "Learning optimal striking points for a ping-pong playing robot," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4587–4592.

[20] O. Koç, G. Maeda, and J. Peters, "Online optimal trajectory generation for robot table tennis," *Robotics and Autonomous Systems*, vol. 105, pp. 121–137, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889017306164

[21] A. AlAttar, L. Rouillard, and P. Kormushev, "Autonomous air-hockey playing cobot using optimal control and vision-based bayesian tracking," in *Towards Autonomous Robotic Systems*, K. Althoefer, J. Konstantinova, and K. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 358–369.

[22] P. Liu, D. Tateo, H. Bou-Ammar, and J. Peters, "Efficient and reactive planning for high speed robot air hockey," 2021.

[23] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. MIT Press, 2022.

[24] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[25] N. Hogan, "Impedance control: An approach to manipulation," in *American Control Conference, 1984*. IEEE, 1984, pp. 304–313.

[26] K. Kronander and A. Billard, "Passive interaction control with dynamical systems," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 106–113, 2016.

[27] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017. [Online]. Available: https://doi.org/10.2514/1.G001921

[28] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 164. PMLR, 2022, pp. 750–759. [Online]. Available: https://proceedings.mlr.press/v164/bhardwaj22a.html

[29] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985. [Online]. Available: https://doi.org/10.1177/027836498500400201

[30] J. Haviland and P. Corke, "A purely-reactive manipulability-maximising motion controller," 2020.

[31] M. Jongeneel, N. van de Wouw, and A. Saccon, "Identification and validation of impact models," Jul. 2022, 10th European Nonlinear Dynamics Conference, ENOC 2022, ENOC 2022 ; Conference date: 17-07-2022 Through 22-07-2022. [Online]. Available: https://enoc2020.sciencesconf.org/

[32] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of Physics*, ser. Halliday & Resnick Fundamentals of Physics. John Wiley & Sons Canada, Limited, 2010. [Online]. Available: http://books.google.co.uk/books?id=49h2cgAACAAJ

[33] O. Khatib, "Inertial properties in robotic manipulation: An object-level framework," *The International Journal of Robotics Research*, vol. 14, no. 1, pp. 19–36, 1995.

[34] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[35] Y.-C. Chen and I. Walker, "A consistent null-space based approach to inverse kinematics of redundant robots," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 1993, pp. 374–381 vol.3.

[36] A. Dietrich, C. Ott, and A. Albu-Schäffer, "An overview of null space projections for redundant, torque-controlled robots," *The International Journal of Robotics Research*, vol. 34, no. 11, pp. 1385–1400, 2015. [Online]. Available: https://doi.org/10.1177/0278364914566516

[37] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[38] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[39] H. Khalil, *Nonlinear Systems*, ser. Pearson Education. Prentice Hall, 2002. [Online]. Available: https://books.google.ch/books?id=t_d1QgAACAAJ

[40] J. Burdick, "On the inverse kinematics of redundant manipulators: characterization of the self-motion manifolds," in *Proceedings, 1989 International Conference on Robotics and Automation*, 1989, pp. 264–270 vol.1.

[41] D. Ma and A. Rodriguez, "Friction variability in planar pushing data: Anisotropic friction and data-collection bias," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3232–3239, 2018.

[42] B. Feeny, A. Guran, N. Hinrichs, and K. Popp, "A Historical Review on Dry Friction and Stick-Slip Phenomena," *Applied Mechanics Reviews*, vol. 51, no. 5, pp. 321–341, 05 1998. [Online]. Available: https://doi.org/10.1115/1.3099008

[43] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, Nov 2021. [Online]. Available: https://doi.org/10.1007/s00170-021-07682-3

[44] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," nov 2012, version 20121115. [Online]. Available: http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html

Harshit Khurana received an M.Sc. in Robotics, Systems and Control from ETH Zurich, in 2019. He is currently conducting research towards a Ph.D. degree with the Learning Algorithms and Systems Laboratory (LASA), Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland. His research interests currently are in motion planning and control for robots with intentional impacts and learning to improve the motion plan through understanding the motion of the environment.

Aude Billard received an M.Sc. in physics from the Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland, in 1995, and an M.Sc. in knowledge-based systems and a Ph.D. degree in artificial intelligence from the University of Edinburgh, Edinburgh, U.K., in 1996 and 1998, respectively. She is currently a full professor in the Institutes of Micro and Mechanical Engineering and the head of the Learning Algorithms and Systems Laboratory, School of Engineering, EPFL. Her research interests are in machine-learning methods for making robots adaptive and able to learn through human guidance and practice.