# Learning to Hit: A statistical Dynamical System based approach

Harshit Khurana, Michael Bombile and Aude Billard

*Abstract*—**This paper proposes a manipulation scheme based on learning the motion of objects after being hit by a robotic end-effector. This allows for the object to be positioned at a desired location outside the physical workspace of the robot. An estimate of the object dynamics under friction and collisions is learnt and used to predict the desired hitting parameters (speed and direction), given the initial and desired location of the object. Based on the obtained hitting parameters, the desired pre-impact velocity of the end-effector is generated using a stable dynamical system. The performance of the proposed DS is validated in simulation and and is used to learn a model for hitting using real robot. The approach is tested on real robot with a KUKA LBR IIWA robot.**

## I. Introduction

Nowadays, robots are widely used in industry for instance in repetitive tasks such as pick and place. Usually, in such a task, the initial and final positions of the object are within the reach of the robot, i.e. in its workspace. Robot swift manipulation of objects in unstructured and dynamic environment is crucial for the industry. For instance, in logistics, the booming of e-commerce and its related challenges have increase the need to speed up the pace of pick-and-place operations. In some cases, this need can be achieved by resorting to manipulation strategies that exploit impulsive actions (either through pushing, hitting or throwing) to move an object to its desired position. Such a strategy not only could speed up the process, but also will allow the object to be placed at a desired position well beyond the robot's natural manipulation boundaries. Thus in this work, we are interested in this kind of manipulation strategy, more particularly in those which exploit an intentional impact to impart a movement to the object in order to place it at a desired location.

When tossing objects, many challenges need to be overcome to ensure that the object will land at the appropriate location. The speed and orientation at impact are crucial. A slight offset in either of these two variables may lead the robot to fall short from its desired location, or conversely to land offsite. Such failure may lead to damages to the object or the environment. If humans are to work in the vicinity, this may lead to injuries.

Unlike pushing, where the action is continuous and can be adjusted before the object is released, when hitting or tossing the object, the action is instantaneous. This means that the action must be adjusted appropriately right from the start so that it will yield the desired displacement.

Additional challenges arise from the uncertainty linked to the dynamics of the object in the environment. The object in its motion is subjected to gravity forces, frictional forces and other aerodynamic forces such as the drag, which are generally only known approximately. If the object's dynamics is known, determining the evolution of its state from a known initial state and action is a trivial problem. However, this work is mainly concerned by the inverse problem which consists of determining the initial action which would cause the evolution of the state of the object to reach the desired position. This problem is not trivial, generally admits several solutions and is further complicated by uncertainties associated to physical phenomena involved. Thus, to address this problem, we will follow a machine learning-based approach, where the mapping between the desired position of the object and the initial angle and speed of the object's motion is learned and used in new situations. Thus, besides the generation of the hitting motion using dynamical systems (DS), we follow a two-step approach to realize hitting-based manipulation tasks by a robot. First, we learn the appropriate hitting parameters (initial angle and speed) of the object given its initial and desired positions. Second, we determine a strategy to modulate the pre-impact speed of the robot's end-effector to produce the desired post-impact velocity of the object regardless of the joint configuration at hitting time.

All authors are with the Learning Algorithms and Systems Laboratory, EPFL, Lausanne, Switzerland, e-mail: {firstname.lastname}@epfl.ch.

## II. Related Work

Manipulation outside the physical workspace of a robot has been an innovative and creative task that has led to wide research on motion of the robot and various controls associated. Combining this with robots working in a semi-structured environment with humans leads to the issue of safety and we discuss the work present in this scenario. A wide variety of methods have been implemented in the areas of mechanics, probabilistic modeling of object motion such as [1].

A wide group of work have studied the problem of pushing an object [2]. The studies vary from analytical modelling [3] to data driven modelling of motion. Pushing alone is a hard task to predict and plan and hence early approaches tried to identify parameters such as friction on-line [4], [5]. More recent approaches gather dataset for learning the intricacies of pushing an object using a manipulator, covering different shapes, material properties, pushing speed, accelerations, contact position and direction [6] and use the dataset to model planar pushing [7], [8] using Gaussian Processes [9]. A probabilistic technique for modeling planar sliding of non uniform objects was shown in [10]. It focuses on comparing the real motion to different simulated motion with different friction and mass models and attempts to close the sim-to-real gap.

Pushing an object however remains constrained by the robot's workspace. Because of quasi-static assumptions it requires a constant application of force to move and place the object. Tossing has been offered as a means to extend this workspace [11]. Similar to tossing, hitting can vastly expand the robot's workspace and as been studied in the context of sport such as for playing golf [12], table tennis [13], juggling [14], football (like RoboCup) [15], and Volleyball [16]. The above have in common the aspect of hitting a ball, spherical in shape, and in which case the hitting is invariant to the orientation of the object. In terms of playing table tennis and juggling a table tennis ball, the In minigolf [12], although the final position of the ball matters, it is not required for the ball to stop at the position. The hitting parameters are learnt from the correct demonstrations which do not completely learn the motion of the golf ball.

Compared to pushing, tossing and hitting an object brings a number of additional challenges. Quasi-static assumption is no longer valid as neither the velocity of the end-effector nor the object are small. Hitting requires an impulse generation that makes it harder to predict the post impact object velocity. [14] estimates the coefficient of restitution and the drag coefficient to improve the process of hitting, but is bound by symmetric hitting of a spherical ball.

We propose a system which does not assume quasi-static motion. It uses stable DS to generate motion and produce non zero relative contact velocity between the robot and an object. Moreover, it models from data, the object's motion dynamics and predicts its post impact behaviour.

## III. Approach

We seek to generate robot motions to impact an object such that the post-impact state of the object reaches a desired state within some tolerance. This section describes how we build a probabilistic model of the object's motion and how we learn the hitting task parameters. We then show how we can combine the learned models with a sequence of dynamical systems control law. The latter offers great resilience to change in the target and the original location of the robot.

### A. Probabilistic modeling of the object motion

A physics based model capturing the motion of an object post-impact depends on the environment conditions such as friction, contact and impact dynamics, drag forces etc, that are hard to model. Given the fact that the post impact behaviour of an object depends on the pre-impact conditions of the environment, we create a regression model from the pre-impact hitting conditions to post-impact sliding observations. The pre-impact hitting conditions are direction of hitting, and speed of the robot end-effector represented as $(v_{ee}, \theta)^T$, initial position of the object $(x_i, y_i)$, and the post-impact observations are the final position of the object on the sliding surface, represented as $(x_f, y_f)$.

Let $G$ be the hitting function learnt. Using $(x_f, y_f)$ and $(x_i, y_i)$, we calculate the desired displacement of the object and the direction of the hit, which are the input to this regression model. $I$ and $O$ represent the input and the output of the regression model respectively. We have $G(I) = O$ with $I = (d_{des}, \theta)$ and $O = v_{ee}$.

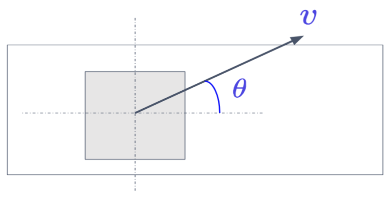


Fig. 1: Representation of the object on the sliding plane.

### B. Learning hitting task parameters

Hitting parameters are pre-impact end-effector velocity and the direction of hitting the object.

Calculation of hitting direction is straightforward. The object's initial position is represented as $\chi_i = [x_i; y_i; z_i]$ and the desired final position of the object is $\chi_f = [x_f; y_f; z_f]$, for planar sliding, we have:

$$\theta = tan^{-1} \frac{y_f - y_o}{x_f - x_o} \tag{1}$$

Calculation of the hitting speed is done by building a probabilistic estimate of the relationship between the hitting angle, speed and desired distance: Given the learned model of the data with features $(d_{des}, \theta, v_{ee})^T$, we create a regression model which separates the features as input and output of the regression function. This is done via Gaussian Mixture Regression (GMR), described in VII-A. GMR is a generative model which allows us to compute the conditional distributions of the variables at hand [17].

### C. DS-based motion generation

Once we have learnt the direction and the speed of the end effector, we need to create an autonomous hitting motion for the robot. To generate the flow, we need the end effector to approach the object towards the face which needs to be hit and move in the direction of the final desired position. This is achieved with a combination of three different linear dynamical systems.

- **Main Dynamical System** $(f_1(\chi))$ attracts the motion of the system towards to the main attractor, $(\chi_1^*)$ in the direction of the final desired position of the object. This allows the end effector to move in the direction of hitting.
- **Auxiliary Dynamical System** $(f_2(\chi))$ attracts the motion of the system to a phantom attractor $(\chi_2^*)$ placed in between the object and the robot. This allows the robot to first move towards the side of the object which needs to be hit.
- **Modulated Dynamical System** attracts the system to the line joining the object and the final desired position. This dynamical system generates motion to keep the robot's end effector from moving away the desired hitting path.

They are formulated below:

$$\begin{aligned} f_2(\chi) &= R_z(\theta) A_2 R_z(\theta)^T (\chi - \chi_2^*) \\ f_1(\chi) &= R_z(\theta) A_1 R_z(\theta)^T (\chi - \chi_1^*) \end{aligned} \tag{2}$$

$$f_m(\chi) = R_z(\theta) K_m R_z(\theta)^T \chi_{\perp} \tag{3}$$

where, $\chi_{\perp}$ is normal vector joining $\chi_{ee}$ and the line formed by $\chi_o$ and $\chi_1^*$. $R_z$ is a rotation matrix depicting rotation around $z$ axis, since the object is being hit in the horizontal / world $X - Y$ frame.

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\theta$ is given by eq. (1). $A_2$ and $A_1$ are the eigenvalue matrices of the respective dynamical systems, which ensure their stability ($A_1, A_2 < 0$).
$K_m$ is the modulation coefficient that allows the action of the modulated DS when the system is close to the line joining the object position and the main attractor.

$$K_m = diag(-e^{(-\|\chi_{\perp}\|/\sigma^2)}) \tag{4}$$

The final DS is a linear combination of the dynamical systems defined above in equation (2) and (3) and renormalised to the desired speed of the end effector.

$$f(\chi) = \alpha(\chi) f_2(\chi) + (1 - \alpha(\chi)) f_1(\chi) + f_m(\chi) \tag{5}$$

$$\dot{\chi} = f_h(\chi) = \frac{f(\chi)}{||f(\chi)||} v_d \tag{6}$$

where, $f_h(\chi)$ is the hitting dynamical system and $v_d$ is the desired end effector speed.

*Designing the weighting function* $\alpha(\chi)$: $\alpha(\chi)$ shifts the attractor of $f_h(\chi)$ from auxiliary attractor to the main attractor, depending on the current position of the end-effector. Fig. 2 depicts geometrically, the projections used in the calculation of $\alpha(\chi)$.

$$\alpha(\chi) = \frac{||proj(\chi_{ee} - \chi_o)||}{||proj(\chi_{ee_i} - \chi_o)||} \tag{7}$$

where, $\chi_{ee}$ is the current end effector position and $\chi_{ee_i}$ is the initial effector position in cartesian coordinates. At the start of the motion, $\alpha(\chi) = 1$, which enables the end effector to move towards the phantom attractor. As soon, as the end effector starts moving, $\alpha(\chi)$ moves from 1 to 0, at which point the motion is towards the main attractor.

The motion generated by the dynamical system as represented in Eq. 6 is first tested in simulation to see how it hits the object, with different initial positions of the robot, object and varying hitting directions.

### D. Stability proof for the Hitting Dynamical System

To prove the stability of $f(\chi)$, it is sufficient to prove the stability of $g(\chi)$, where $f(\chi) = R_z(\theta) g(\chi) R_z(\theta)^T$, since this is just a rotation of the dynamical system.

$$g(\chi) = \alpha(\chi) A_2(\chi - \chi_2^*) + (1 - \alpha(\chi)) A_2(\chi - \chi_1^*) + K_m(\chi_{\perp})$$

$K_m(\chi_{\perp})$ can be written as follows:

$$K_m(\chi_{\perp}) = K_m(\chi - \chi_3^*)$$

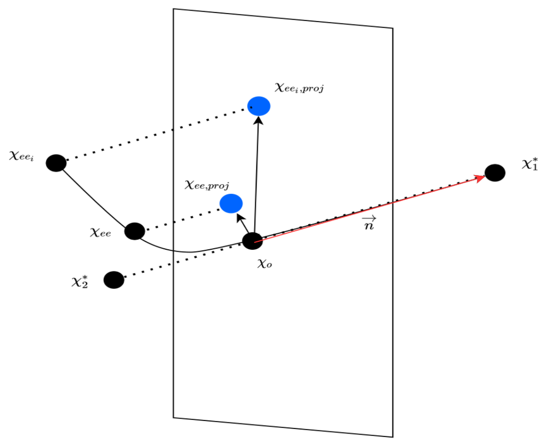


Fig. 2: Geometrical understanding of $\alpha(\chi)$: $\alpha(\chi)$ is the ratio of the length of vectors obtained by projecting relative position vectors of end-effector and object in different states before the impact on the plane with the normal as vector joining the pre-impact object position and desired final position.

For the sake of this proof, we denote $1 - \alpha(\chi) = \beta(\chi)$, and $\alpha(\chi)$ and $\beta(\chi)$ are represented as $\alpha$ and $\beta$.

$$A_1 = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{12} & 0 \\ 0 & 0 & a_{13} \end{bmatrix}, A_2 = \begin{bmatrix} a_{21} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{23} \end{bmatrix}$$

$$K_m = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix}$$

where, $, a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, k_1, k_2, k_3 < 0$.
$g(\chi)$ can be written as follows:

$$\begin{aligned} g(\chi) &= L\chi - LL^{-1}\chi_4^* \\ &= L(\chi - L^{-1}\chi_4^*) \end{aligned} \tag{8}$$

where

$$L = \begin{bmatrix} \alpha a_{21} + \beta a_{11} + k_1 & 0 & 0 \\ 0 & \alpha a_{22} + \beta a_{12} + k_2 & 0 \\ 0 & 0 & \alpha a_{23} + \beta a_{13} + k_3 \end{bmatrix} \text{ and}$$

$$\chi_4^* = \begin{bmatrix} \alpha a_{21}\chi_2^* + \beta a_{11}\chi_1^* + k_1\chi_3^* \\ \alpha a_{22}\chi_2^* + \beta a_{12}\chi_1^* + k_2\chi_3^* \\ \alpha a_{23}\chi_2^* + \beta a_{13}\chi_1^* + k_3\chi_3^* \end{bmatrix}$$

Eq. 8 is a linear time invariant system and is globally asymptotically stable if eigenvalues of $L < 0$. Since $L$ is a diagonal matrix, it means that all the diagonal entries of $L$ should be less than 0.
We have, $0 \leq \alpha, \beta \leq 1$, by design and $a_{ij} < 0$. This makes sure that individual diagonal entries of $L$ are less than 0, which means eigenvalues of $L < 0$, thus making $g(\chi)$ globally asymptotically stable [18].

### E. Robot dynamics and control

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

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \mathbf{g}(\mathbf{x}) + \mathbf{f}_e = \mathbf{f}_c \tag{9}$$

where $\mathbf{x} \in \mathbb{R}^3 \times SO_3$ denotes the task space vector of position and orientation. $\mathbf{M}(\mathbf{x}) \in \mathbb{R}^{n\times n}$ and $\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}}) \in \mathbb{R}^{n\times n}$ denote the robot's inertia, and centrifugal and Coriolis matrices. $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^6$ is the vector of gravity wrench. $\mathbf{f}_e \in \mathbb{R}^6$ represents an external wrench applied by the robot, whereas $\mathbf{f}_c \in \mathbb{R}^6$ is the control wrench of the robot. Now assuming that the gravity wrench is already compensated and separating the position from the orientation task, to track a desired task space velocity $\dot{\mathbf{x}}_i^d$ while preserving passivity, the control force and torque $\mathbf{f}_{c_i} \in \mathbb{R}^3$ can be designed such that [19]

$$\mathbf{f}_{c_i} = -\mathbf{D}_i(\mathbf{x}_i)(\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_i^d) = d_{i1}\dot{\mathbf{x}}_i^d - \mathbf{D}_i(\mathbf{x}_i)\dot{\mathbf{x}}_i \tag{10}$$

with $i = (p, o)$ which stand for position and orientation. $\mathbf{D}_i(\mathbf{x}_i) \in \mathbb{R}^{3\times 3}$ is a state-damping matrix, whose first eigenvector $d_{i1}$ is aligned with $\dot{\mathbf{x}}_i^d$ and the other are orthogonal.

For the hitting motion considered in this paper, we focus essentially on the position component by generation $\dot{\chi}_p^d$ using autonomous dynamical system ($\dot{\chi}_p^d = f(\chi_p)$) to be described in section (III-C). Whereas for the orientation, the desired angular velocity $\dot{\chi}_o^d = \omega^d$ is computed using quaternion error between the current and the desired orientation (predefined). Thus, we have

$$\omega^d = k.(\delta\mathrm{q} - \mathrm{q}) \times \mathrm{q}^* \tag{11}$$

where $\mathrm{q}$ and $\mathrm{q}^*$ denotes respectively the current end-effector quaternion orientation and its conjugate. $\delta\mathrm{q} = Slerp(\mathrm{q}, \mathrm{q}^d, 0.5)$ with $\mathrm{q}^d$ the desired quaternion and $Slerp$ represents the spherical interpolation function. $k$ is a positive gain.

## IV. Experiments

### A. Setup

The proposed method is evaluated with the scenario of a robot hitting an object (a cardboard box) in a planar manner on a table with unknown friction value. Fig. 3 shows the setup consisting of a KUKA LBR IIWA7 which is a 7 DoF robot, with a $26cm \times 26cm \times 23cm$ box, weighing $0.363kg$. The dynamical system mentioned in Sec. III generates motion that passes through the center of mass of the object, which is assumed to be at the geometrical center of the object, considering uniform mass distribution. This allows for the hit to not generate unwanted roational effects on the object, so that it is able to slide on the surface. The motion of the box is tracked and recorded through Optitrack motion capture system with Prime 17W cameras with streaming frequency 250 Hz. Fig. 4 shows the difference between the commanded speed of the end effector and the achieved speeds. The achieved speed saturates around $1.1m/s$.

### B. Data Collection

We collect the data on how the object moves upon being hit by the robot with a certain velocity and in a certain direction using the Hitting DS, engineered to pass through the center of mass of the known object. The data is collected using a real KUKA iiwa 7 robot hitting an object, as shown in Fig. 3. The commanded speed and commanded direction are uniformly distributed in the space of $[0, 2.5]m/s$ and $[-0.4, 0.4]$ radians. The data collected for learning of the model are the commanded speed and direction, the end effector velocity at the time of hitting the object, initial position of the object and the final position of the object. The collected data consists of 60 data points which are uniformly spread around in the above mentioned space of direction and speed.

### C. Learning

• *Modeling the mapping from end effector speed to the final object position*: A Gaussian Mixture Model is used to model the entire dataset using Expectation - Maximisation. Since, this leads to a local optimal solution, the modelling is performed with different initialization to find the better fit. Bayesian Information Criterion is used to select the optimal number of Gaussians modelling the data. BIC calculates a tradeoff between the likelihood of the model and number of parameters used in the model. Fig. 5 shows one such model for the data with 3 Gaussians. Through this we obtain $P(d, \theta, v_{ee})$.

Once, we have the model of the data, we predict the desired hitting speed, given the initial and the final desired position of the object using GMR, which calculates desired speed as expectation of conditional probability $v_d = E(P(v|d, \theta))$. The calculation of expected mean and the variance of prediction has been previously shown in Section. VII-A and III.

The model has been implemented using the architecture in [20].

## V. Results

### A. Implementation

Table. I shows the desired final positions of the object, relative to its initial position. Using this, we calculate the distance and direction and predict the hitting speed. The desired final position spans across the table as shown in Fig. 7. For each of the final desired positions, the experiment is repeated 5 times.

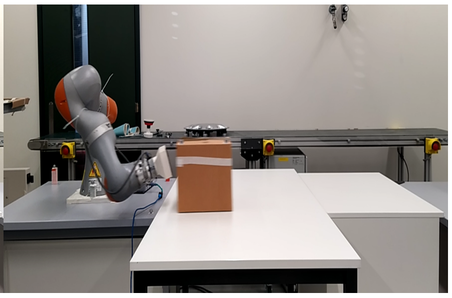
TABLE I: Experimental Data

| No. | $\chi_{rel}$ | $d_{des}$ | $\theta(rad)$ | $v_{ee}(m/s)$ |
|---|---|---|---|---|
| 1 | $(0, 0.5)^T$ | 0.5 | 0.0 | 0.817 |
| 2 | $(0.2, 0.5)^T$ | 0.54 | -0.38 | 0.91 |
| 3 | $(-0.1, 0.6)^T$ | 0.61 | 0.16 | 0.94 |
| 4 | $(-0.1, 0.7)^T$ | 0.71 | 0.14 | 0.98 |
| 5 | $(-0.2, 0.5)^T$ | 0.54 | 0.38 | 0.96 |

### B. Fastest contact speed

Feasible Fastest impact-safe hitting is currently achieved at $1.1m/s$ and is also shown in Fig. 4

(a) Approaching phase


(b) Hitting phase

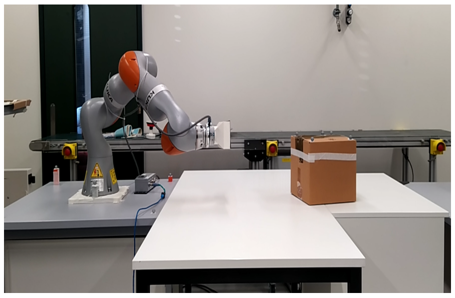

(c) Final position of the object

Fig. 3: Snapshots depicting three stages of the box hitting experiment. In (a) the robot is approaching the object. In (b) the robot hits the object. We can see here that the impacting with a flat end effector leads to further uncertainty due to differences in orientation of the box and the end-effector. In (c) we see the final position of the object being outside the physical workspace of the robot

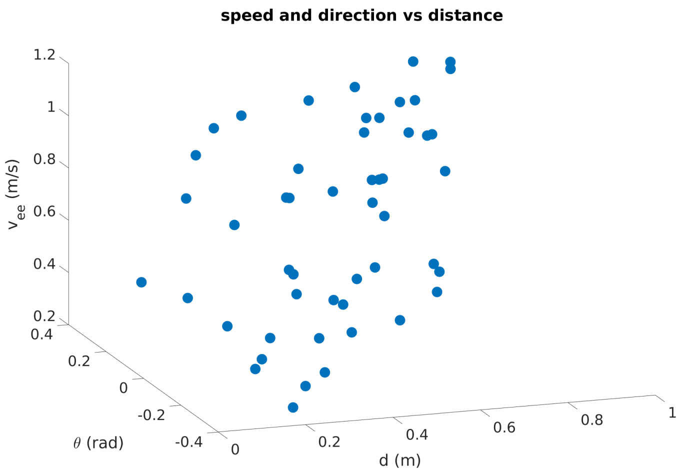


Fig. 4: Collected data: for different end effector speeds and direction of hit, we measure the displacement of the object.

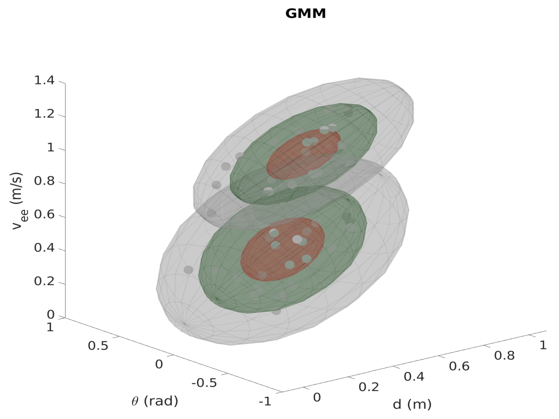


Fig. 5: GMM Model: Here we see the data modeled using two gaussians in a GMM. Each gaussian is shown with three iso-contour ellipsoids.

## C. Error in prediction of Hitting Speed

While we select the number of gaussians in the model using cross validation and BIC criteria, the RMSE in the predicted speeds from the test data are calculated to have an idea of the accuracy of the prediction. The average RMSE for hitting speed prediction in the cross validation prediction is $0.085 m/s$. The results of GMR are visualised in Fig. 6.

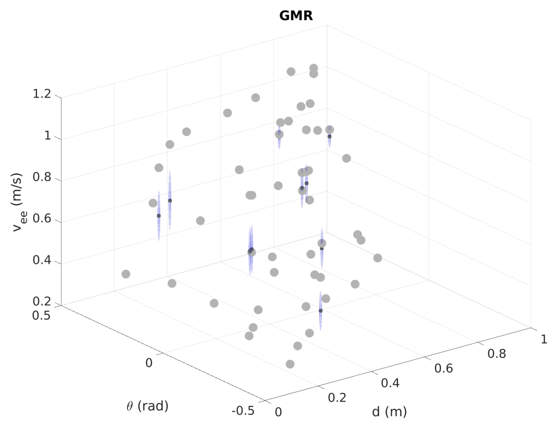


Fig. 6: An example of speed prediction for the cross validation test data - ellipsoid in blue depicts one standard deviation in the prediction value.

## D. Position Accuracy and Repeatability

The experiment is repeated five times for each desired final position of the object. The measure employed is the relative RMSE error in the distance covered by the object. Fig. 7 shows in black, the desired final positions of the box and the bounding rectangles show the achieved final positions of the box, using the hitting speed estimates from the GMR model. All locations are reached within the tolerance except for one outlier for the desired position $[-0.1; 0.7]$. Table II shows the data comparing the achieved distances with the desired distance. We test the model by RMSE and relative RMSE on the distance covered by the object after being hit, which are

TABLE II: Desired and achieved distances (in $m$)

| No. | $d_{des}$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ |
|---|---|---|---|---|---|---|
| 1 | 0.5 | 0.49 | 0.54 | 0.53 | 0.52 | 0.54 |
| 2 | 0.54 | 0.47 | 0.66 | 0.53 | 0.63 | 0.69 |
| 3 | 0.61 | 0.63 | 0.66 | 0.69 | 0.68 | 0.69 |
| 4 | 0.71 | 0.97 | 0.71 | 0.71 | 0.67 | 0.74 |
| 5 | 0.54 | 0.48 | 0.50 | 0.64 | 0.61 | 0.46 |

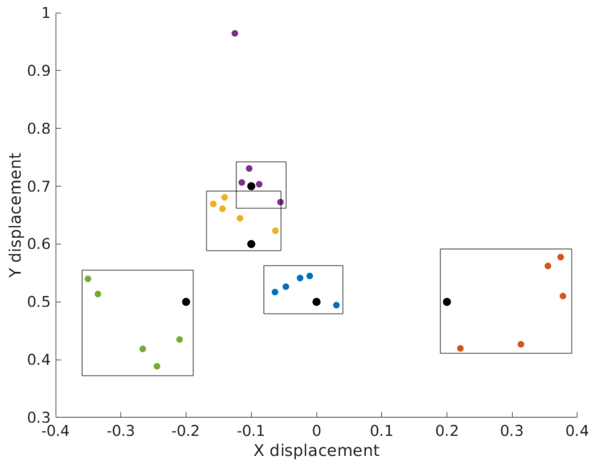


Fig. 7: Precision with which the box reached the desired target. In black, the desired final positions of the box. The bounding rectangles show the achieved final positions of the box, using the hitting speed estimates from learned model.

TABLE III: Accuracy of distance coverage (in $m$)

| No. | $d_{des}(m)$ | $RMSE(m)$ | rel $RMSE(\%)$ |
|---|---|---|---|
| 1 | 0.5 | 0.03 | 6.35 |
| 2 | 0.54 | 0.10 | 18.9 |
| 3 | 0.61 | 0.06 | 10.67 |
| 4 | 0.71 | 0.12 | 17.03 |
| 5 | 0.54 | 0.07 | 13.84 |

reported in table III. From Fig. 8, we see that there exists the outlier we notice in Fig. 7. The outlier is attributed to double hit from the end-effector due to a late joint orientation. Including the outlier in the calculations, we have maximum relative RMSE of $18.9\%$. We see more variance in reaching the desired positions which have higher $\theta$ values. This is because the end effector hits the object on its edge which leads to higher uncertainty due to rotational effects induced.

### E. Directional Accuracy

Table IV shows the errors in the desired and the achieved direction of hitting and Fig. 9 depicts the variance in the directional accuracy for the hit. The hitting accuracy is lowest in test cases 2 and 5, which correspond to the desired positions of $(0.2, 0.5)^T$ and $(-0.2, 0.5)^T$. This is because of the end effector hitting the object on its edge that leads to higher uncertainty. This can be overcome by either more data collection or first aligning the object with the direction of hitting so that the end effector hits the object as desired.

TABLE IV: Accuracy of direction of motion (in $rad$)

| No. | $\theta_{des}(rad)$ | $RMSE(rad)$ | rel $RMSE(\%)$ |
|---|---|---|---|
| 1 | 1.57 | 0.077 | 4.87 |
| 2 | 1.19 | 0.21 | 17.3 |
| 3 | 1.74 | 0.05 | 2.91 |
| 4 | 1.71 | 0.03 | 1.74 |
| 5 | 1.95 | 0.18 | 8.86 |

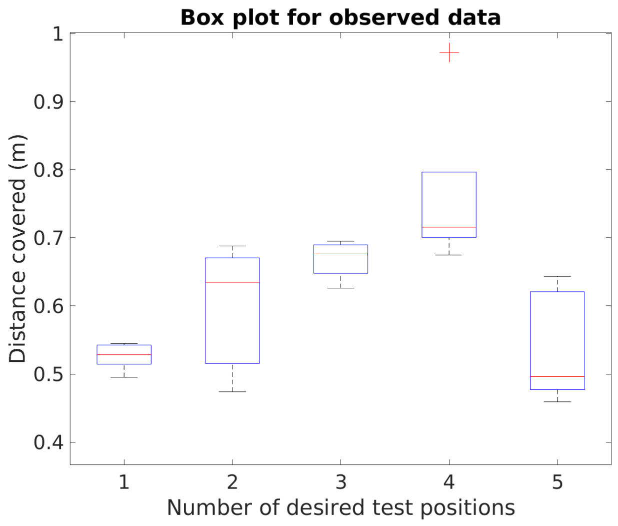


Fig. 8: Box Plot for the Test data: Shows the variance for the different desired positions for the object. Higher variance in noted when high directionality of hit is required.

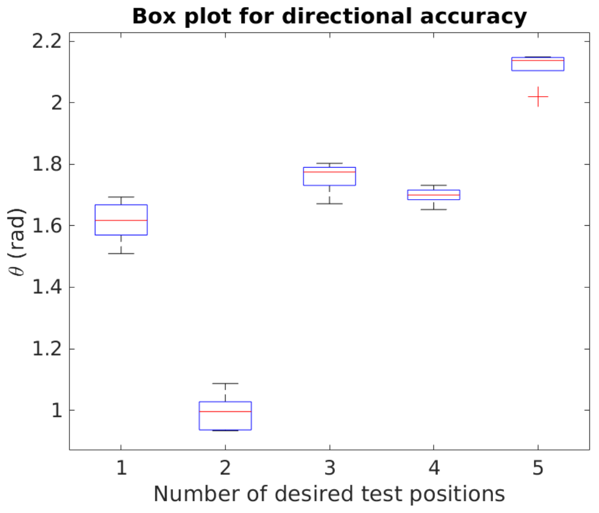


Fig. 9: The directional accuracy is lower when the direction of hit $\theta$ is not near 1.57 rad. The further we move from 1.57 rad, the more the variance increases.

## VI. Discussion and Future Work

In this paper, we proposed a DS based manipulation scheme for hitting a known 3D object in a planar manner, using data driven model for the object's motion. Data driven model has been chosen since modeling friction, drag, and pressure distributions are a non trivial problem. Future work involves learning motion models better with limited data points, creating stable path planning dynamical systems that are able to transfer momentum and generate accelerations, hence aiding dynamic motions.

## Acknowledgments

The authors would like to thank for the support from the European Commission, Project I-AM, and Farshad Khadivar for his help in the implementation of the controller used in this paper.

## VII. Appendix

### A. Background Knowledge

Gaussian Mixture Regression forms an integral part of this paper, and hence is described briefly in this section. Assume we have the joint probability distribution of the data which consists of both input and output. The probability that a data-point $\zeta = [O; I]$ ($O$ being the output and $I$, input) belongs to a GMM is as follows:

$$P(\zeta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\zeta; \mu_k, \Sigma_k)$$

where $\pi_k$ are the prior probabilities of the gaussians of the GMM and $\mathcal{N}(\zeta; \mu_k, \Sigma_k)$ are the Gaussian distributions composing the GMM. $\mu_k$ and $\Sigma_k$ are the means and covariance matrices of the $k^{th}$ gaussian and can be written as:

$$\mu_k = \begin{bmatrix} \mu_{Ik} \\ \mu_{Ok} \end{bmatrix}, \Sigma_k = \begin{bmatrix} \Sigma_{Ik} & \Sigma_{IOk} \\ \Sigma_{OIk} & \Sigma_{Ok} \end{bmatrix}$$

Once, we have the GMM, we compute the distribution of the output variable $O$, given the input variable $I$ and Gaussian $k$

$$P(O|I, k) \sim \mathcal{N}(\hat{\mu}_k, \hat{\Sigma}_k)$$

where,

$$\hat{\mu}_k = \mu_{Ok} + \Sigma_{OIk}\Sigma_{Ik}^{-1}(I - \mu_{Ik})$$
$$\hat{\Sigma}_k = \Sigma_{Ok} - \Sigma_{OIk}\Sigma_{Ik}^{-1}\Sigma_{IOk}$$

Using the above equations, we can sum over all the gaussians to generate conditional expectation of $O$, given $I$.

$$\hat{\mu} = \sum_{k=1}^{K} h_k \hat{\mu}_k, \qquad \hat{\Sigma} = \sum_{k=1}^{K} h_k^2 \Sigma_k$$

where,

$$h_k = \pi_k \frac{\mathcal{N}(I; \mu_k, \Sigma_k)}{\sum_{k=1}^{K} \mathcal{N}(I; \mu_k, \Sigma_k)}$$

## References

[1] S. Kim and A. Billard, "Estimating the non-linear dynamics of free-flying objects," *Robotics and Autonomous Systems*, vol. 60, p. 1108–1122, 09 2012.

[2] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing," *Frontiers in Robotics and AI*, vol. 7, p. 8, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2020.00008

[3] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 533–556, 1996.

[4] T. Yoshikawa and M. Kurisu, "Indentification of the center of friction from pushing an object by a mobile robot," in *Proceedings IROS'91: IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91*. IEEE, 1991, pp. 449–454.

[5] "Estimating the friction parameters of pushed objects," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, vol. 1, 1993, pp. 186–193 vol.1.

[6] 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.

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

[8] F. H. M. Bauza and A. Rodriguez, "A data-efficient approach to precise and controlled pushing," in *CoRL*, 2018.

[9] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[10] C. Song and A. Boularias, "A probabilistic model for planar sliding of objects with unknown material properties: Identification and robust planning," 2020.

[11] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," 2019.

[12] 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

[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] 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.

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

[16] H. Nakai, Y. Taniguchi, M. Uenohara, T. Yoshimi, H. Ogawa, F. Ozaki, J. Oaki, H. Sato, Y. Asari, K. Maeda, H. Banba, T. Okada, K. Tatsuno, E. Tanaka, O. Yamaguchi, and M. Tachimori, "A volleyball playing robot," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 2, 1998, pp. 1083–1089 vol.2.

[17] H. G. Sung, *Gaussian mixture regression and classification*. Rice University, 2004.

[18] C.-T. Chen, *Linear System Theory and Design*, 3rd ed. USA: Oxford University Press, Inc., 1998.

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

[20] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2016.