A Dynamical System Approach for Catching Softly a Flying Object: Theory and Experiment

Seyed Sina Mirrazavi Salehian*, Mahdi Khoramshahi, and Aude Billard

***Abstract*—Catching a fast flying object is particularly challenging as consists of two tasks: it requires extremely precise estimation of the object's motion and control of the robot motion. Any small imprecision may lead the fingers to close too abruptly and let the object fly away from the hand before closing. We present a strategy to overcome for sensori-motor imprecision by introducing softness in the catching approach. *Soft* catching consists of having the robot moves with the object for a short period of time, so as to leave more time for the fingers to close on the object. We use a dynamical systems (DS) based control law to generate the appropriate reach and follow motion, which is expressed as a Linear Parameter Varying (LPV) system. We propose a method to approximate the parameters of LPV systems using Gaussian Mixture Models, based on a set of kinematically feasible demonstrations generated by an off-line optimal control framework. We show theoretically that the resulting DS will intercept the object at the *intercept* point, at the right *time* with the desired *velocity direction*. Stability and convergence of the approach are assessed through Lyapunov stability theory. The proposed method is validated systematically to catch three objects that generate elastic contacts and demonstrate important improvement over a *hard* catching approach.**

I. Introduction

Dynamic motions such as catching [1]–[8], juggling [4], [9], hitting [3], [10] and throwing [9] require accurate motion planning and motor control. Perfect control can never be satisfied in practice. Devising control strategies to offer more robustness in the face of imprecise sensing and actuation is important. Catching objects in flight is a particularly challenging task as the time to move toward the object is extremely short, lasting usually about a quarter of a second. The time at impact is even shorter, leaving less than a few milliseconds for the hands to close on the object to secure it tightly in the grip. We propose a compliant strategy to catch objects, that is less sensitive to timely control of the interception. The *soft* catching strategy consists of having the robot moves with the object for a short period of time (Fig. 1). This leaves more time for the fingers to close on the object and avoids failure due to imprecise control of the time and position at which the hand intercepts the object.

Previous works on catching objects focused on a *hard* catching approach, in which the hand intercepts and stops the object instantaneously [4], [6], [11]. Such strategy generates enormous intercept forces which may cause the object to bounce out of the hand, before the fingers have a chance to close on the object [12]. This is particularly the case when the weight of the arm is greater than that of the object by an order of magnitude and when the object is made of hard material. In this case, the shock is quasi-elastic and the object is seen flying away from the hand as soon as it hits the palm. In contrast, in *soft* catching, the hand meets the object at an *intercept* point, but then continues moving, aligning its speed to that of the object, and slowly reduces its speed to zero. Since soft catching provides more time to grasp, it is more robust against uncertainties in the hand orientation as well as fingers closure initiation; i.e. it allows readjustments for hand, palm, and fingers posture.

The soft catching strategy we propose here assumes that the arm-hand-object system has no mechanical compliance or that the inherent

All authors are with the School of Engineering, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland. {sina.mirrazavi; mahdi.khoramshahi;aude.billard}@epfl.ch

*Corresponding author

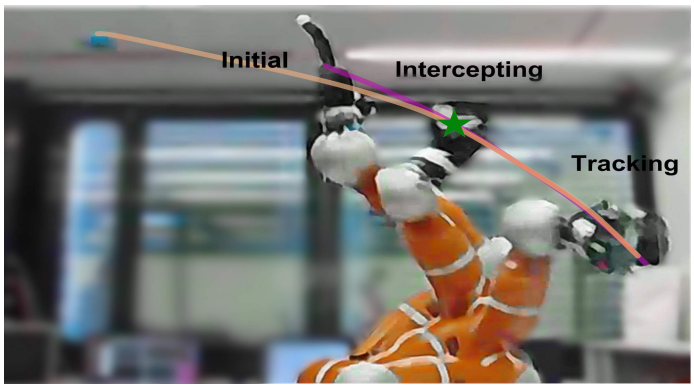


Fig. 1: Schematic of the soft catching strategy. The arm starts from an *initial* point and moves to the *intercept* point, where it meets the object. It then continues its motion aligning its trajectory with that of the object and slowly reducing its velocity, while the fingers close on the object.

mechanical compliance of the system is negligible in comparison to the strength of the interaction forces. Compliance is then provided through active control of the arm motion.

To successfully catch an object *softly*, the arm must intercept the object *on time*, *at the right place*, and with a *specific velocity*. Although reaching on time implies that soft catching is a time-dependent system, the motion can be made time-invariant by coupling the motion of the robot arm with the object's dynamic. This makes the system more robust in the face of changes in the estimated dynamics of the object's flight.

Planning a trajectory that satisfies the above three constraints in time, position and velocity can be done using standard optimal control approaches [11]. This is however time consuming and cannot be performed within the few milliseconds at our disposal. Here, we leverage the properties of autonomous dynamical systems for immediate re-planning of motion [13]. The dynamical system (DS) is expressed as a Linear Parameter Varying (LPV) system subject to stability constraints to ensure that the resulting system converges asymptotically to the object's trajectory. To estimate the parameters of the LPV, one can use arbitrary regressive techniques for estimating the parameters. Here, we use a probabilistic approach through Gaussian Mixture Regression. To train the system, we generate off-line a set of the fastest possibly kinematically feasible trajectories.

This paper presents a theoretical analysis of the stability and convergence of the proposed DS control law and an empirical evaluations against hard catching approaches.

II. Related work

In order to accomplish a soft catching task, solutions to two problems are required. First, a feasible intercept posture must be determined. Second, a precise soft catching motion subject to the robot constraints needs to be generated.

A robotic arm with large and convex workspace and high degree of redundancy in joints configuration makes it possible to intercept a flying object at any point along its trajectory. However, most of these assumptions are not true in practice. In our previous work [6], we proposed a framework to extract feasible postures to intercept general shaped objects. We reuse this method in this work. In this previous work, we however considered only a hard catching motion. Besides, we could not ensure that the robot would catch the object at a precise time. We extend this work by offering a soft catching procedure that is more resilient to imprecisions in controlling the arm and we ensure that the arm reaches the object at the desired point and at the desired time. Finally, in this previous work, we estimated only a first order DS, hence controlling for velocity. This had the drawback to generate trajectories with too high an acceleration and

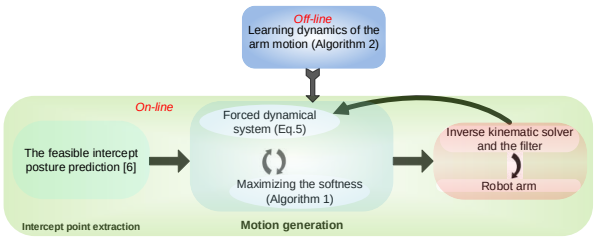


Fig. 2: Block diagram for robotic soft catching.

that could not be followed. In this paper, the DS is second order and models feasible acceleration profiles.

Planning a fast motion with equality point constraints, such as in catching or hitting a flying object, has been extensively studied in the literature. Fitting a polynomial trajectory to predetermined points –initial, intercept and stop– has been proposed and used in [1], [5], [14]. Even though this method is computationally efficient and can ensure to get at the desired position on time, it utilizes time as an explicit variable. As a result, it is highly sensitive to imprecise estimate of the catching position and time, and would require a complete re-planning as a new desired intercept point is provided. Besides, there is no guarantee that the generated trajectory is feasible for a robot to track. Optimal control can address some of these issues, see [11], [12], by taking into account the terminal constraints and the kinematic limitations of the robot. However, these are iterative procedure with no insurance to convergence in a small enough number of time steps to ensure extremely fast reactivity. Also, the convergence is highly dependent on the initial guess. Imitation learning is also applicable to this problem, see [2], [6]. By using machine learning techniques, one can approximate an expert's set of demonstrations and use this approximation to model the requested dynamics of motion [13]. If human demonstrations are provided directly, through kinesthetic teaching, this ensures that the motions are kinematically feasible [6]. However, the demonstrations can never be exhaustive and generalization away from the demonstrations is prone to error. Here, we generate a comprehensive dataset of "expert's" trajectories off-line and in simulation so as to span the space of kinematically and dynamically feasible trajectories. We then learn an approximation of these trajectories through probabilistic techniques used usually in learning from demonstration.

A soft catching motion is composed of two motions; in the first phase, *reaching* and *tracking*. During the *reaching* motion, the end-effector starts from the initial point and intercepts the object's trajectory at the desired intercept point. After the interception, the end-effector tracks the predicted object trajectory while slowly reducing the object velocity. Although *reaching* motion results in *hard* catching, combining a reaching motion with a *tracking* motion results in a smooth motion which rejoins the object at the desired intercept point and then continues moving with the object; i.e. *soft* catching. One possible solution for combining these motions is to use hybrid systems. Hybrid control technique have two main shortcomings: (1) the switching between the modes would yield high non-linearities and discontinuities; (2) initiating the switches from inspection requires significant intelligence and insight [15]. This paper proposes a method where the two motions are smoothly integrated in one dynamical system which generates implicitly this smooth switching from reaching to tracking.

In the first part of this work, we formulate our control law as a LPV based dynamical system to generate soft catching motion. LPV systems have been proposed as a promising avenue for approximating a single model of a plant in multiple operating conditions [16]. One typically choice is to approximate LPV systems by the use of linear regression models. The most popular approaches to perform this approximation use polynomial or periodic functions [17]. In this paper, we use a probabilistic approach to estimate the parameters of the LPV using Gaussian Mixture Models (GMM), so as to account for the inherent stochasticity of the training data-points. We analytically prove that the generated soft catching motion and consequently the end-effector intercept the object trajectory at the desired intercept point and follow it thereafter. Moreover, with the purpose of maximizing the softness at the interception subject to the kinematic constraints of the robot, a closed loop optimal control problem is suggested. The performance of the proposed method is validated in a real-world experiment with KUKA LBR IIWA (7 degree of freedom arm robot).

This paper is organized as follows. Section III formalizes the LPV dynamical system. In addition, the closed loop optimal control problem is introduced. In Section IV, the switching parameters are approximated by GMM parameters and stability and convergence of the proposed system are studied. We also present the algorithm for generating a data-set used later for training the LPV dynamical system. The proposed method is validated by the experimental set-up in Section V. Discussion is presented in Section VI.

III. The Soft Catching Dynamical System

Dynamical systems are popular and powerful methods for autonomously generating stable and robust motions according to training data-points [13]. DSs are applicable for various robotic manipulations which need an accurate and fast re-computing of motions. Formulating DSs as LPV systems allows modeling a wide class of nonlinear systems and the use of many tools from the linear systems theory for analysis and control [18]. LPV systems can be thought of as a weighted combination of linear models, each valid at a specific operating point. We consider a class of continuous-time LPV systems given by the following model:

$$\begin{aligned} \ddot{\xi}(t) &= A^1(\theta_{A^1}(t))\xi(t) + A^2(\theta_{A^2}(t))\dot{\xi}(t) + u(t) \\ y(t) &= C_g\left[\xi(t) \quad \dot{\xi}(t)\right]^T \end{aligned} \tag{1}$$

Where $\xi(t) \in \mathbb{R}^D$ is the state of the dynamical system; e.g. in robotic arm manipulators $\xi(t) \in \mathbb{R}^6$ is the position and the orientation of the end-effector. $u(t)$ is the control input vector. $y(t)$ is the plant output vector. $\theta_{A^i} \in \mathbb{R}^{K_1 \times 1}\ \forall i \in \{1,2\}$ are the vector of scheduling parameters[1];

$$\theta_{A^i} = \begin{bmatrix} \theta_{A^i_1} & \dots & \theta_{A^i_{K_1}} \end{bmatrix}^T \qquad \forall i \in \{1,2\} \tag{2}$$

. $A^i(.) : \mathbb{R}^{K_i} \to \mathbb{R}^{D \times D}\ \forall i \in \{1,2\}$ are the affine dependences of the state-space matrices on the scheduling parameter and the state vectors:

$$\begin{aligned} A^1(\theta_{A^1}) &= \sum_{k=1}^{K_1} \theta_{A^1_k} A^1_k \quad A^1_k \in \mathbb{R}^{D \times D} \quad \theta_{A^1_k} \in \mathbb{R}^{1 \times 1} \\ A^2(\theta_{A^2}) &= \sum_{k=1}^{K_2} \theta_{A^2_k} A^2_k \quad A^2_k \in \mathbb{R}^{D \times D} \quad \theta_{A^2_k} \in \mathbb{R}^{1 \times 1} \end{aligned} \tag{3}$$

As we discussed in Section I, soft catching can be achieved as a combination of *tracking* and *reaching* motions. To achieve this, the following control input vector $u(t)$ is proposed.

$$\begin{aligned} u(t) =& \gamma(t)\ddot{\xi}^O - A^1(\theta_{A^1})\gamma(t)\xi^O - A^2(\theta_{A^2})(\gamma(t)\dot{\xi}^O + \dot{\gamma}(t)\xi^O) \\ &+ 2\dot{\gamma}(t)\dot{\xi}^O + \ddot{\gamma}(t)\xi^O \end{aligned} \tag{4}$$

[1]The scheduling parameters can be a function of time (t), states of the system $(\xi(t))$ or external signals $d(t)$, i.e. $\theta_{A^1}(t, \xi(t), d(t))$. In the rest of the paper, the arguments of $\theta_{A^i}\ \forall i \in \{1,2\}$ are dropped for simplicity.

Where the state the object is denoted by ξ^O. $0 < \gamma(t) < 1$ is called softness and it is a continuous and continuously differentiable parameter. The origin is located on the desired intercept point and must be reached at time T^*; i.e. $\xi^O(T^*) = \begin{bmatrix}0 & \dots & 0\end{bmatrix}^T$. To ensure smooth tracking of the object's motion, the combination of the position, velocity and acceleration of the object with the affine dependences is chosen as the control law. By substituting, (4) and (3) into (1), we have:

$$\begin{aligned}&\ddot{\xi}(t) - \gamma(t)\ddot{\xi}^O(t) - 2\dot{\gamma}(t)\dot{\xi}^O(t) - \ddot{\gamma}(t)\xi^O(t) = \\ &A^1(\theta_{A^1})(\xi(t) - \gamma(t)\xi^O(t)) + A^2(\theta_{A^2})(\dot{\xi}(t) - (\gamma(t)\dot{\xi}^O(t) + \dot{\gamma}(t)\xi^O(t))) \\ &y(t) = C_g\begin{bmatrix}\xi(t) & \dot{\xi}(t)\end{bmatrix}^T\end{aligned} \tag{5}$$

Theorem 1: The dynamical system given by (5) *asymptotically converges to* $\begin{bmatrix}\gamma(t)\xi^O & \gamma(t)\dot{\xi}^O + \dot{\gamma}(t)\xi^O\end{bmatrix}^T$ *; i.e.*

$$\lim_{t\to\infty}\|\xi(t) - \gamma(t)\xi^O(t)\| = 0 \tag{6}$$

$$\lim_{t\to\infty}\|\dot{\xi}(t) - (\gamma(t)\dot{\xi}^O(t) + \dot{\gamma}(t)\xi^O(t))\| = 0 \tag{7}$$

if (5) *meets the following constraints:*

$$\begin{cases}A^1_{k^1} + (A^1_{k^1})^T \prec 0 & A^2_{k^2} + (A^2_{k^2})^T \prec 0 \\ A^1_{k^1} = (A^1_{k^1})^T & \forall k^1 \in \{1,\dots,K_1\} \qquad \forall k^2 \in \{1,\dots,K_2\} \\ 0 \le \theta_{A^1_{k^1}} \le 1, & 0 \le \theta_{A^2_{k^2}} \le 1\end{cases} \tag{8}$$

where $\prec 0$ refers to negative definiteness of a matrix.
Proof: see Appendix A.

If we assume that $\gamma(t)$ is constant, (5) is a combination of two motions; i.e. *reaching* and *tracking*. In this equation, setting $\gamma = 0$ yields a *reaching* dynamical system; i.e. $\lim_{t\to\infty}\begin{bmatrix}\xi(t) & \dot{\xi}(t)\end{bmatrix} = \begin{bmatrix}\xi^O(T^*) & 0_{1\times D}\end{bmatrix}^T$. Hence, the *position* constraint is satisfied and the *time* and *velocity* constraints are not satisfied. Setting $\gamma = 1$ results in a tracking motion with an error decreasing asymptotically according to:

$$\ddot{\xi}(t) - \ddot{\xi}^O(t) = A^1(\theta_{A^1}(t))(\xi(t) - \xi^O(t)) + A^2(\theta_{A^2}(t))(\dot{\xi}(t) - \dot{\xi}^O(t)) \tag{9}$$

By varying the value of the gamma parameter, one can ensure that we reach the object not only with the right velocity, but that we reach it at the right location and at the right time. This is summarized in the following corollary:

Corollary 1: The dynamical system given by (5) reaches the desired intercept point $(\xi^O(T^*))$ asymptotically with a velocity aligned with that of the object, $\dot{\xi}(T^*) \approx \gamma\dot{\xi}^O(T^*)$.[2]

Proof: The intercept point is located along the object's trajectory and so is the origin. As $\gamma\xi^O(T^*) = \begin{bmatrix}0 & \dots & 0\end{bmatrix}^T$ is also at the origin, $\gamma\xi^O$ crosses ξ^O at the desired intercept point. Since the arm is ensured to reach asymptotically $\gamma\xi^O$, it is ensured to pass through the object at the desired intercept point (Theorem 1). Moreover, it will do so with a velocity vector proportional to that of the object, i.e. $\dot{\xi}(T^*) \approx \gamma\dot{\xi}^O(T^*)$. ■

[2]We assume that the dynamical system (5) is fast enough to converge to the acceptable neighbourhood of the desired trajectory $\gamma\begin{bmatrix}\xi^O & \dot{\xi}^O\end{bmatrix}^T$ before the catching time; i.e. $\|\xi(T^*) - \gamma\xi^O(T^*)\| \le \varepsilon$ and $\|\dot{\xi}(T^*) - \gamma\dot{\xi}^O(T^*) - \dot{\gamma}\xi^O(T^*)\| \le \varepsilon$, where ε is a small positive number.

Algorithm 1 Pseudo-code for the optimal control formulation for maximizing the softness of catching.

Do for each step i

$$\ddot{\gamma}[i+1] = \underset{\ddot{\gamma}}{\text{argmax}}(\gamma[i+1])$$

subject to:

$0 < \gamma[i+1] < 1$	(Alg-1-1)
$-\Upsilon \le \ddot{\gamma}[i+1] \le \Upsilon$	(Alg-1-2)
$\dot{\gamma}[i+1] = \dot{\gamma}[i] + \ddot{\gamma}[i+1]\Delta t$	(Alg-1-3)
$\gamma[i+1] = \gamma[i] + \dot{\gamma}[i+1]\Delta t$	(Alg-1-4)
$\ddot{\xi}[i+1]$ is calculated from (5)	(Alg-1-5)
$-J(q[i])\dot{q}_{max} \le \dot{\xi}[i+1] \le J(q[i])\dot{q}_{max}$	(Alg-1-6)
$-\dot{J}(q[i])\dot{q}_{max} - J(q[i])\ddot{q}_{max} \le \ddot{\xi}[i+1] \le \dot{J}(q[i])\dot{q}_{max} + J(q[i])\ddot{q}_{max}$	(Alg-1-7)
$\delta \le p(\xi[i+1];\theta_W)$	(Alg-1-8)

By increasing the value of γ, on the one hand, the end-effector's velocity at the intercept point will get closer to the object velocity hence provide more softness in catching. On the other hand, the generated trajectory may be kinematically infeasible for the robot. In order to generate a kinematically feasible trajectory with maximum softness in catching, we propose a closed loop optimal control which is formulated in Algorithm 1. The goal is to maximize the softness $(\gamma(t))$ while ensuring that the trajectory is kinematically feasible for a robot to track. This introduces a constrained optimization problem at each step which can be formulated as a Nonlinear Programming (NLP) problem.

In Algorithm 1, $\le$ corresponds to the component-wise inequality. Υ is a large positive number. Δt is the time step. $q[i]$ is the joint configuration which is corresponding to the end-effector position and orientation $\xi[i]$; i.e. $\xi[i] = \mathbb{F}(q[i])$ where $\mathbb{F}$ is the robot forward kinematic function. (Alg-1-1) satisfies the velocity constraint in a soft catching motion. (Alg-1-6) and (Alg-1-7) guarantee that the velocity and acceleration of the generated motion is kinematically feasible for the robot, respectively. The feasibility of the generated motion at the position level is guaranteed by (Alg-1-8), where the workspace of the robot is modeled through a probabilistic representation of the feasible postures $(p(\xi;\theta_W))$ [6]. In order to generate the training dataset, all the possible postures of the robot are simulated by testing all the displacements of its joints. A probabilistic model of these positions and orientations are constructed by using a Gaussian Mixture Model as follows:

$$p(\xi;\theta_W) = \sum_{l=1}^{K_w}\pi_l\mathcal{N}(\xi^R|\mu_l,\Sigma_l) \tag{10}$$

where π_l, μ_l, Σ_l correspond to the prior, mean and covariance matrix of the $l = 1\dots K_w$ Gaussian functions, respectively, and they are calculated by using the Expectation-Maximization algorithm [19]. ξ is classified as a feasible configuration if $p(\xi;\theta_W)$ exceeds a *minimum likelihood* threshold δ, which is determined such that the likelihood of 99% of the training points are higher than the threshold, for more information see [6].

Example: Consider (5) as the following 1-D dynamical system with one scheduling parameter.

$$\begin{cases}A^2 = -12 \\ A^1 = -36\end{cases} \Rightarrow \ddot{\xi} = -12\dot{\xi} - 36\xi + u(t) \tag{11}$$

This is a critically damped system which asymptotically converges to zero from an arbitrary initial condition $(\xi(0) = 2,\ \dot{\xi}(0) = 0)$. Let's assume the following dynamic model for the object.

$$\ddot{\xi}^O(0) = 0.1 \qquad \dot{\xi}^O(0) = 1 \qquad \xi^O(0) = -3 \tag{12}$$

The solutions of (5) for this example are illustrated in Fig. 3 for different values of γ and $\dot{\gamma}$. It shows that for all the values of γ, $\dot{\gamma}$ and $\ddot{\gamma}$, the generated trajectories join the object's trajectory on time at the origin. Moreover, by increasing the value of γ, the end-effector's

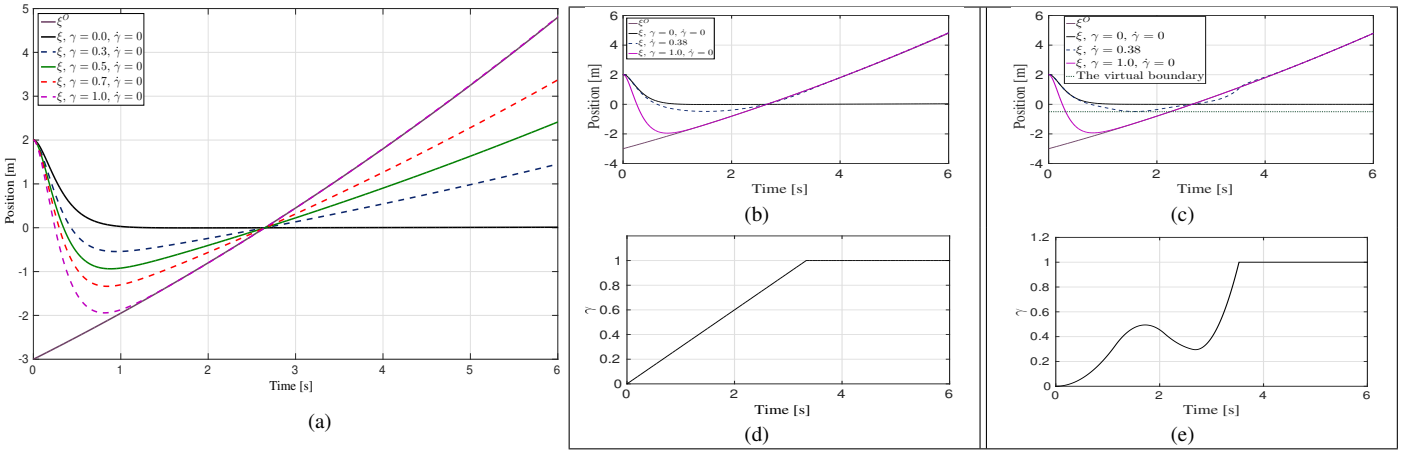


Fig. 3: The behavior of 1-D forced dynamical system subject to the value of γ and $\dot{\gamma}$. $\dot{\gamma}=0$ in (a). (b) and (c) show the behavior of the 1-D system when $\dot{\gamma}$ is constant and time-varying, respectively. (d) and (e) show the corresponding value of γ in (b) and (c), respectively. In (c), the virtual workspace constraint is satisfied as Algorithm 1 is used to optimize the value of γ.

Algorithm 2 Pseudo-code for generating the fastest kinematically feasible demonstrations.

Step 1: *Initialization*
Set $i=1$ and define a fixed initial end-effector position r_I.
$\xi_m[i]=r_I, \quad \dot{\xi}_m[i]=[0], \quad \ddot{\xi}_m[i]=[0]$
$q[i]=\mathbb{F}^{-1}(r_I), \quad \dot{q}[i]=[0], \quad \ddot{q}[i]=[0]$
Randomly define an attractor (r_D) inside the workspace of the robot.
Step 2: *Trajectory planning*
While $\|r_D-\xi_m[i]\|\geq\varepsilon$

$$\ddot{q}[i+1]=\underset{\ddot{\mathrm{q}}}{\operatorname{argmax}}(\|\dot{\mathrm{q}}[i+1]\|)$$

subject to:

$$-\ddot{q}_{max}\leq\ddot{\mathrm{q}}[i+1]\leq\ddot{q}_{max} \quad \text{(Alg-2-1)}$$
$$-\dot{q}_{max}\leq\dot{\mathrm{q}}[i+1]\leq\dot{q}_{max} \quad \text{(Alg-2-2)}$$
$$\dot{q}[i+1]=\dot{q}[i]+\Delta t\ddot{q}[i+1] \quad \text{(Alg-2-3)}$$
$$\|\frac{r_D-\xi_m[i]}{\|r_D-\xi_m[i]\|}-\frac{J(q[i])\dot{q}}{\|J(q[i])\dot{q}\|}\|\leq\varepsilon \quad \text{(Alg-2-4)}$$

Calculate the next joint configuration:
$\dot{q}[i+1]=\dot{q}[i]+\Delta t\ddot{q}[i+1]$.
$q[i+1]=q[i]+\Delta t\dot{q}[i+1]$.
$\xi_m[i+1]=\mathbb{F}(q[i+1])$.
$\dot{\xi}_m[i+1]=J(q[i+1])\dot{q}[i+1]$.
$\ddot{\xi}_m[i+1]=\dot{J}(q[i+1])\dot{q}[i+1]+J(q[i+1])\ddot{q}[i+1]$.
$i=i+1$.
End

velocity is getting closer to the object velocity: compare the black line with the purple line in Fig. 3a. Fig. 3c shows an example of use of Algorithm 1 to generate a feasible trajectory where there is a geometrical constraints on the motion of the dynamical system.

In order to use the LPV dynamical system (1) to successfully catch a flying object *softly*, the scheduling parameters must be accurately and precisely modeled. Next, we propose a GMM based second order dynamical system and reformulate it to model the unforced LPV dynamical system (1).

IV. Estimating the switching parameters of LPV-based dynamical system

In this section, we introduce a new approach for approximating the parameters of the LPV based dynamical systems (1) from a training data-set. Approximating a dynamical system via a GMM from a training data set is popular approach since stability and convergence of the dynamical system can be analytically guaranteed [13]. To estimate a LPV dynamical system via a GMM, the parameters of the DS become the priors π^k, the means μ^k and the covariance matrices Σ^k of the $k\in\{1,...,K\}$ Gaussian function, K denotes the number of Gaussian components. A representation of an unforced LPV dynamical system (1) with Gaussian mixture model is formulated as:

$$\ddot{\xi}=\sum_{k=1}^{K_1}h_k^1(\xi)(A_k^1\xi+b_k^1)+\sum_{k=1}^{K_2}h_k^2(\dot{\xi})(A_k^2\dot{\xi}+b_k^2) \quad \forall\xi,\dot{\xi},\ddot{\xi}\in\mathbb{R}^D \tag{13}$$

where

$$\begin{cases} A_k^1=\Sigma_{\ddot{\xi}\xi}^k(\Sigma_\xi^k)^{-1} & A_k^2=\Sigma_{\ddot{\xi}\dot{\xi}}^k(\Sigma_{\dot{\xi}}^k)^{-1} \\ b_k^1=\mu_{\ddot{\xi}}^k-A_k^1\mu_\xi^k & b_k^2=\mu_{\ddot{\xi}}^k-A_k^2\mu_{\dot{\xi}}^k \\ h_k^j(x)=\dfrac{P_j(k)P_j(x|k)}{\sum_{i=1}^{K}P_j(i)P_j(x|i)} & \forall(j,x)\in\{(1,\xi),(2,\dot{\xi})\} \end{cases} \tag{14}$$

In this formulation, $P_j(k)=\pi_j^k\ \forall j\in\{1,2\}$ is the prior probability of each Gaussian component and $P_j(x|k)=\mathcal{N}(x|\mu_x^k,\Sigma_x^k)\ \forall(j,x)\in\{(1,\xi),(2,\dot{\xi})\}$ denotes the conditional probability distribution function (pdf) corresponding to the k$^{\text{th}}$ Gaussian function. $0<h_k^j(x)<1$ is a continuous and continuously differentiable function. The second order dynamical system (13) subject to the following constraints:

$$\begin{cases} A_i^k+(A_i^k)^T\prec 0 & \forall i\in\{1,2\} \\ A_1^k=(A_1^k)^T & \forall k\in\{1,\ldots,K\} \\ \mu_{\dot{\xi}}^k=A_k^1\mu_\xi^k & \mu_{\ddot{\xi}}^k=A_k^2\mu_{\dot{\xi}}^k \end{cases} \tag{15}$$

is equivalent to the unforced LPV dynamical system (1) subject to the stability constraints (8). Hence, the dynamical system (5) can be rewritten[3]:

$$\begin{aligned} &\ddot{\xi}-\gamma(t)\ddot{\xi}^O(t)-2\dot{\gamma}(t)\dot{\xi}^O(t)-\ddot{\gamma}(t)\xi^O(t)= \\ &\sum_{k=1}^{K_1}h_k^1(\xi(t)-\gamma(t)\xi^O(t))A_k^1(\xi(t)-\gamma(t)\xi^O(t))+ \\ &\sum_{k=1}^{K_2}h_k^2(\dot{\xi})A_k^2(\dot{\xi}(t)-(\gamma(t)\dot{\xi}^O(t)+\dot{\gamma}(t)\xi^O(t))) \\ &\forall\xi,\dot{\xi},\ddot{\xi}\in\mathbb{R}^D \end{aligned} \tag{16}$$

We seek to train our system using a set of training data-points $\{\xi_m,\dot{\xi}_m,\ddot{\xi}_m\}$ that encompass examples of kinematically and dynamically feasible trajectories of the end-effector to the intercept point.

[3]Theorem 1 can be proven by the following non-negative Lyapunov candidate: $V=\frac{1}{2}(\dot{\xi}-\gamma\dot{\xi}^O-\dot{\gamma}\xi^O)^T(\dot{\xi}-\gamma\dot{\xi}^O-\dot{\gamma}\xi^O)-\int_0^{\xi-\gamma\xi^O}\left(y^T\sum_{k=1}^{K_1}h_1^k(y)(A_1^k)\right)\mathrm{d}y$

As it would be difficult to have these provided by a human expert, as kinesthetic teaching would not make it possible to move the arm at its maximal speed, we opt for generating off-line through an optimal control problem the desired demonstrations. Moreover, as catching is an extremely rapid action, the training trajectories should be representative of the fastest feasible motions of the robot. Accordingly, we propose Algorithm 2 to generate the training data set. The algorithm consists of two main steps. In step 1, the initial r_I and the final r_D positions of the robot are chosen inside the workspace of the robot. In step 2, the end effector is moved with maximum acceptable velocity and acceleration along a straight line from the initial position to a set of intercept points r_D located in its workspace. Since the maximum feasible velocity and acceleration of the end-effector depend on the joint configuration, these need to be calculated at every step. To relax the constraints of the optimization problem, (Alg-2-4) is defined as an inequality constraint. (Alg-2-1) and (Alg-2-2) guarantee the feasibility of the motion in the velocity and acceleration levels, respectively. Note that, this algorithm does not minimize the motion duration, but it generates a motion which is the fastest at each configuration. In Algorithm 2, $q,\ \dot{q}_{max},\ \ddot{q}_{max} \in \mathbb{R}^m$ are the joint configuration, the maximum acceptable joint velocity and acceleration, respectively. ξ_m is the end-effector position and orientation. $\mathbb{F} : \mathbb{R}^m \to \mathbb{R}^6$ is a known forward kinematic function for the robot. $J(q) \in \mathbb{R}^{6\times m}$ is the Jacobian matrix. ε is a small positive number.

A. Learning Second Order Asymptotically Stable Models

In order to estimate the parameters of the dynamical system given in (13), the following optimization problem is proposed, which uses Mean Square Error as a means to quantify the accuracy of the estimation:

$$\min_{\Theta} C(\Theta) = \min_{\Theta} \sum_{m=1}^{M} \|\ddot{\xi}_m - \ddot{\xi}\|^2 \tag{17}$$

subject to :

$$\begin{cases} A^i_{k_i} + (A^i_{k_i})^T \prec 0 \\ A^1_{k_i} = (A^1_{k_i})^T \\ \mu^{k_i}_{\ddot{\xi}} = A^1_{k_i}\mu^{k_i}_{\xi} \\ \mu^{k_i}_{\ddot{\xi}} = A^2_{k_i}\mu^{k_i}_{\dot{\xi}} \end{cases} \quad \begin{cases} 0 \prec \begin{pmatrix} \Sigma^{k_i}_{\xi} & \Sigma^{k_i}_{\xi\ddot{\xi}} \\ \Sigma^{k_i}_{\ddot{\xi}\xi} & \Sigma^{k_i}_{\ddot{\xi}} \end{pmatrix} \\ 0 \prec \begin{pmatrix} \Sigma^{k_i}_{\xi} & \Sigma^{k_i}_{\xi\ddot{\xi}} \\ \Sigma^{k_i}_{\ddot{\xi}\xi} & \Sigma^{k_i}_{\ddot{\xi}} \end{pmatrix} \\ 0 \le \pi^{k_i}_i \le 1 \end{cases} \quad \sum_{k_i=1}^{K_i} \pi^{k_i}_i = 1 \tag{18}$$

for $\forall k_i \in \{1,\dots,K_i\}$ and $\forall i \in \{1,2\}$. $C(\Theta)$ is the cost function and Θ is the GMM parameters. M is the number of the training data-points. $\ddot{\xi}$ is computed directly from (13).The group of constrains on the left-hand-side of (18) ensures asymptotic stability, while the group of constraints on the right-hand-side follows from the definition of positiveness and bounded integrality for the GMM density; see [13].

Since the NLP problem (17) is not convex, the initialization of the parameters affects the quality of the solution found by the solver. Note that there is always a feasible solution for these NLP problems, but performance of the constructed dynamical system depends on the initial guess of the NLP problem. We used the initialization method which is proposed by [13] for initializing (17) and the interior-point algorithm [20] to find the minimum of the constrained nonlinear multi-variable function. This method uses penalty functions which act as a barrier and prevent the iterates from leaving the feasible region. An overview of the proposed framework for catching an object *softly* is illustrated in the schematic of Fig. 2.

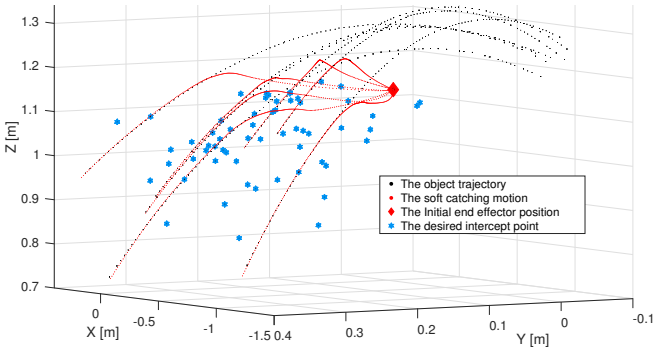


Fig. 4: The final intercept points in the soft catching experiments. The initial position of the palm is $\begin{bmatrix}-0.05 & 0.00 & 1.134\end{bmatrix}$m. For clarity of the illustration, only seven examples of the soft catching motions and the object trajectories are shown. The object trajectories are plotted from the first points till the stop points. The first point is the first object position which is used for predicting the feasible intercept position. The experimental results verifies that the catching motion intercepts the object at the desired point with the desired velocity.

V. Empirical validation

The performance of the proposed framework is evaluated on a real platform, 7 DOF robot arm, KUKA LBR IIWA mounted with a 16 DOF Allegro hand. The output of the dynamical system (5) is converted into the 7-DOF joints state using the velocity based control without joint velocity integration [21]. In order to avoid high torques, the resultant joint angels are filtered by a critically damped filter. The robot is controlled in the joint position level at a rate of 500 Hz. As the joint position controller of the robot is a hight gain perfect tracking controller and to avoid unexpected noises and delays in measuring the joint position of the robot, (5) runs in closed-loop via computing the current end-effector position by using the filtered resultant joint angels; see Fig. 2.

In order to coordinate the motions of all joints –the arm and the fingers joints–, the coupled dynamical system (CDS) model [22] is utilized to generate the fingers motion. This approach consists of coupling two different dynamical systems; i.e. the end-effector motion and the fingers motion. The motion of the end-effector is generated independently from the fingers states, while the fingers motion is a function of the state of the end-effector and the object. The metric of the coupling is the distance between the end-effector and the object ($\|\xi - \xi^0\|$). As a result, the fingers close when the object gets inside the hand and they reopen when the object moves away.

We choose three objects with different stiffness; a very stiff small plastic ball, a stiff fabric brick and a semi-stiff toy. The objects are almost impossible to catch with the hard catching approach [6] as they bounce out of the hand instantaneously, see accompanying video[4]. The position of the objects are captured by the *Optitrack* motion capture system from *Natural point* at 240 Hz. Since the control loop is faster than the capturing system, the predicted position of the object is used as the object position in (5).

The feasible intercept point $(\xi^O(T^*))$ is estimated by the catching point prediction algorithm proposed in [6]. This algorithm follows three steps. First, the graspable areas on the object is determined by sampling from a probability distribution of feasible postures. As the objects have approximately a spherical shape, the graspable areas in our case encompass the entire object surface. The orientation of the catching is however constrained to force the palm to face opposite the trajectory of the object. Note that this could be relaxed when using industrial hands, in place of the humanoid hand we use here, and that do not have preferred orientation for finger closing. Second, the

[4]The video is available at http://lasa.epfl.ch/~sina/Soft_Catching.mp4

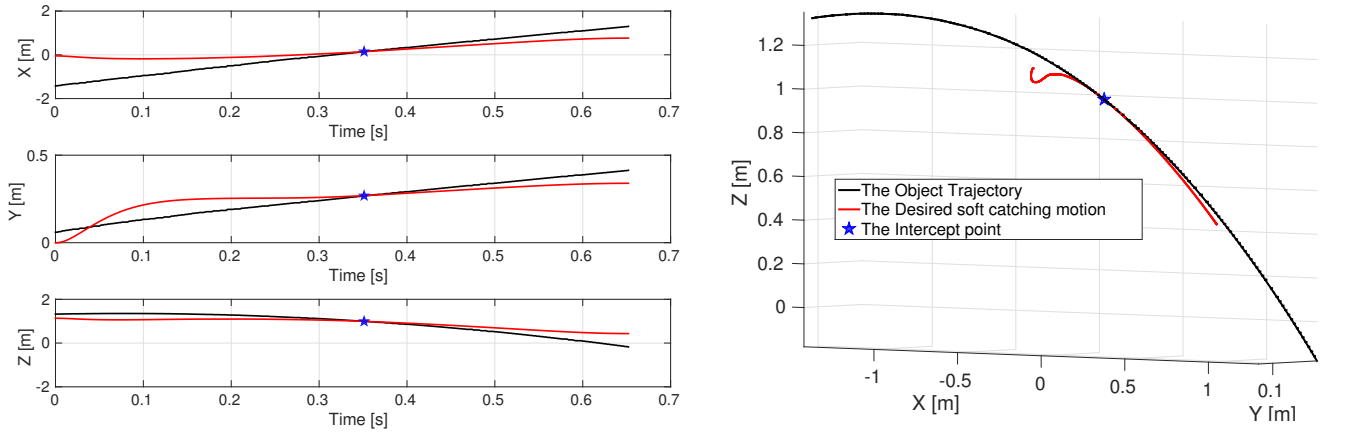
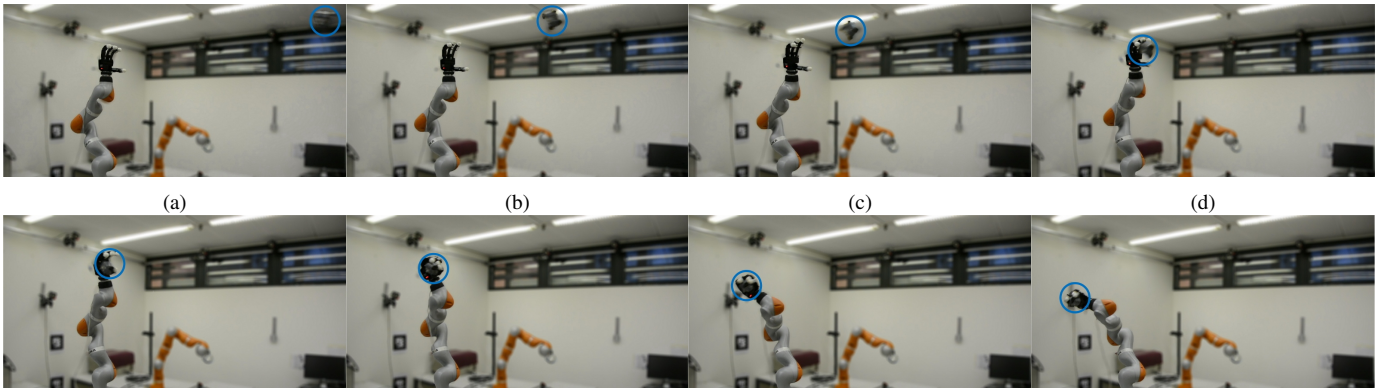


Fig. 5: The position of the end-effector generated by the dynamical system (5). The illustrated object trajectory is the predicted trajectory of the uncaught object. This trajectory is illustrated from the first point till the stop point. The initial value of γ is 0.2 and Algorithm 1 maximizes it with respect to the kinematic constraints of the robot. As expected, the output of (5) softly intercepts the objects trajectory at the desired intercept position. In order to stop the robot, the velocity of the robot is linearly reduced during the post-interception period (0.3s).



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 6: The brick is thrown. In (a), the object trajectory prediction algorithm is being initialized. (b) is approximately the first point. (e) is the interception. One can stop the robot the robot in (g) as the fingers are closed. But it might damage the robot. A corresponding video is available at http://lasa.epfl.ch/~sina/The_uncut_video.mp4.

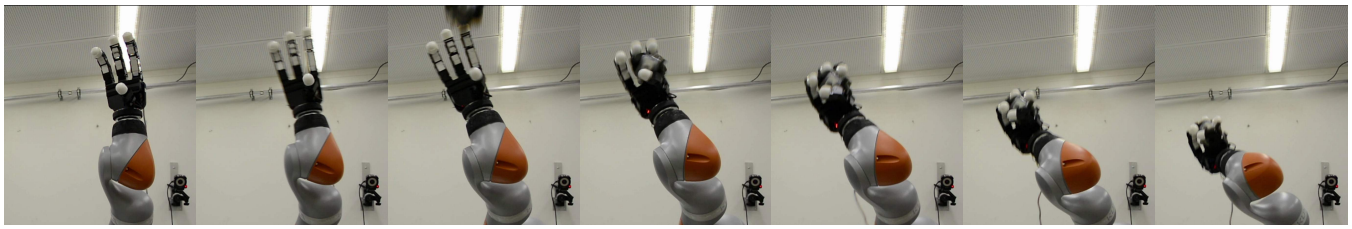
TABLE I: The details of the soft catching experiments. All the position is in respect to the base of the robot. The throwing positions are randomly chosen. The robot does not move till the first intercept point is calculated. As the first 0.4m of the throwing in x direction is used to initialize the object predication trajectory [23]. Time of the flight is the duration of the object flight from the first point till the intercept position. Softness is the softness of the object interception. All the object are thrown 20 times and are caught by the hard [6] and our soft catching algorithms.

	Stiffness	Throwing position (m)			First point (m)			Time of flight (s)	Softness (γ)	Soft catching success rate	Hard catching success rate
The small ball	Too stiff	-1.68 ± 0.14	-0.07 ± 0.02	1.18 ± 0.06	-1.29 ± 0.14	-0.03 ± 0.03	1.30 ± 0.05	0.32 ± 0.03	0.67 ± 0.06	%70	%0.0
The brick	stiff	-1.84 ± 0.07	-0.05 ± 0.02	1.11 ± 0.04	-1.46 ± 0.07	-0.01 ± 0.03	1.25 ± 0.030	0.33 ± 0.02	0.66 ± 0.04	%70	%0.0
The toy	semi-stiff	-1.52 ± 0.62	-0.02 ± 0.07	1.14 ± 0.27	-1.14 ± 0.62	0.03 ± 0.07	1.22 ± 0.39	0.32 ± 0.00	0.50 ± 0.02	%75	%5.0

optimal area in the reachable workspace of the robot is determined through gradient ascent on the likelihood of a probability distribution representing the reaching space (10). Finally, a feasible configuration along the object trajectory is determined by combining the set of feasible postures with the set of points in the reachable and graspable workspace which the object will travel through. See [6] for a complete description.

To validate the algorithm, the experiment was repeated 20 times for each object. The objects were thrown by a human operator. Data of the experimental results are summarized in Table I. The initial position of the object is randomly changed. As the plastic ball and the brick are stiffer than the toy, a softer interception is required for accomplishing the catching. Hence, the experimenter placed himself farther away from the robot than when throwing the toy. First, the prediction of the object's trajectory requires some time and uses almost all of the first 0.4 meter of the object flight in x [23]. In addition, to ensure that the object travels at a reasonable speed, leaving enough time for the robot to travel to the desired position, a distance between the robot and the initial position of the object should be no more than 1.5m which approximatively results in 0.33s flight time. Due to imperfect prediction of the object trajectory, the feasible intercept point needs to be updated and redefined during the catching. The new feasible intercept point is chosen in the vicinity of the previous one to minimize the convergence time and improve the success rate of catching. The feasible intercept point is updated approximately 29 times during the flight time. The intercept points are illustrated in Fig. 4. In this figure, the origin is the position of the robot base. As the right hand of the Allegro hand is used, we throw the objects mostly to the right side of the robot.

The initial values of γ and $\dot{\gamma}$ in (5) are set to 0.2 and 0, respectively. COBYLA algorithm [24] of Nlopt library [25] is used for solving the closed loop optimal control, where the maximum optimization time is set 0.001s . An example of the desired robot trajectory and the unperturbed object predicted trajectory are shown in Fig. 5. As expected, the end-effector converges to the object at the intercept point and continues to track the object predicted trajectory. The snapshots of the real robot experiments is shown in Fig. 6 and Fig. 7. As the closure time of the hand is approximatively 0.1s, one can



(a) -0.447s (b) -0.300s (c) -0.260s (d) -0.193s (e) -0.150s (f) -0.077s (g) 0.0 s

Fig. 7: Snapshots of the finger motions. The object is intercepted in (d) and caught in (f). It is important to note that the closure time for fingers varies with the incoming object speed.

immediately stop the robot when the hand is closed on the object which may damage the robot. Hence, we reduce the end-effector velocity in $0.3s$ to avoid high torques.

The overall success rate of the soft catching reached %71.6, see Table I. To compare and to assess improvement over the catching approach, the experiments were repeated with the similar initial conditions for the *hard* catching scenario [6]. The overall success rate was very low and did not exceed %1.6. Visual inspection of the data and video confirmed that this poor result for the hard catching scenario was essentially due to the fact that the objects bounce out of the hand. The causes of failure for the soft catching strategy can be categorized into three different categories. i) The main cause of the failure for catching is still due to the inability to generate an accurate joint-level motion corresponding to the desired end-effector trajectory; the toy (3 out of 5), the brick (3 out of 6) and for the plastic ball (2 out of 6). As the motion is too fast, the end-effector does not accurately track the desired motion. The tracking error between the desired and the actual end-effector position is approximately $\begin{bmatrix}0.03\pm0.02 & 0.01\pm0.01 & 0.02\pm0.01\end{bmatrix}^T m$. This error causes the object to hit the thumb or undesired parts of the hand and bounce away.

ii) As the Allegro hand has only four fingers, there is a space between the fingers and the palm at the grasp configuration. The small plastic ball can escape the grasp using this space, (2 out of 6). For the other objects, this issue is negligible as there are big enough to be caught with four fingers.

iii) Another cause of the failure arises when the object travels the space too close to the robot. To track the object, all the markers must be visible to the cameras. In five cases, the brick (3 out of 6) and the plastic ball (1 out of 6), the tracking started very late, for lack of visible markers. As a result, the robot was not able to reach the desired intercept point on time and interception occurred at an undesired point. In these cases, the side of the hand hit the object or the interception was not soft enough. As the intercept point is approximately updated 29 times during flight, the first prediction of the object trajectory plays a main role in defining the intercept point. If the initial prediction of the object's trajectory is very inaccurate, the updated intercept points will be far from each other. As a result, (5) does not converge to the latest desired trajectory on time. This was the case for trials using the toy (2 out of 5). Finally, in one trial with the plastic ball (1 out of 6), the first point is too close to the robot and the trajectory prediction does not work. In this case, the object hits and bounces away. However, the robots tried to reach the bounced object.

VI. Discussion

In this paper, we proposed a framework to catch an object softly. There are two important constraints to enable soft catching; namely to reach the object's trajectory with a velocity aligned with that of the object. The motion should be fast enough to intercept the object on *time*. This, of course, depends on having appropriate hardware. If provided with a robot that can travel fast enough to travel the required distance within the required time, then, our algorithm ensures that the robot will catch joint the object *on time*, at the *desired point* with the *desired velocity direction*.

Proof of asymptotic stability was done using Lyapunov stability theorem. Specifically, we showed that our LPV dynamical system asymptotically converges to the object trajectory and intercept it exactly at the desired predefined point. In order to improve the softness of catching, we proposed a closed loop optimal control problem to maximize the value of the softness subject to the kinematic constraints of the robot. Furthermore, a new GMM based method is proposed for accurately approximating and modeling the parameters of LPV systems. Approximating the parameters of the LPV systems via a GMM based model has its own advantages and disadvantage. Using GMM is advantageous in that it can accurately model the training data points. Moreover, the scheduling parameters are of class C^∞ and the transitions between the scheduling parameters are smooth. However, as the proposed learning algorithm is not convex; i.e. the performance of the learned dynamical system is dependent to the initialization.

Corollary 1 shows that the system governed by (5) converges asymptotically to the object trajectory and intercepts it at the desired point. As there is no constraint on the magnitude of the eigenvalues $(|\lambda_{A_j^i}|)$ of A_j^i $\forall(i,j)\in\{(1,1),\dots,(1,K_1),(2,1),\dots,(2,K_2)\}$, there is no guarantee that (5) is fast enough to converge to $\gamma\xi^O$ on time. To successfully catch an object, the arm should arrive in a neighborhood of the desired trajectory $\gamma\begin{bmatrix}\xi^O & \dot{\xi}^O\end{bmatrix}^T$ before the catching time; i.e. $\|\xi(T^*)-\gamma\xi^O(T^*)\|\leq\varepsilon$ and $\|\dot{\xi}(T^*)-\gamma\dot{\xi}^O(T^*)-\dot{\gamma}\xi^O(T^*)\|\leq\varepsilon$, where ε is a small positive number. Hence, one must choose an intercept point and a ratio of velocity between the robot arm and the object's flight which leave enough time for the arm to reach the target on time. As a simple example, consider a case in which the workspace of the robot is a sphere of a diameter of 100cm and the minimum flight time for the object is 0.3s with ε =1cm and assume (5) is a critically damped system, then $|\lambda_{A_j^i}|$ $\forall(i,j)\in\{(1,1),\dots,(1,K_1),(2,1),\dots,(2,K_2)\}$ must be approximately greater than 22. In practice, $22<|\lambda_{A_j^i}|$ is not only very conservative constraint but also may result in a dynamical system which generates kinematically infeasible motions. To address this challenge, a potential direction would be extracting the desired intercept posture subject to the dynamic constraints of the robot and the success rate of the intercept posture.

Throughout the proofs, we assume that the desired intercept point is a fixed attractor. In practice, due to the imperfect prediction of the object trajectory, we need to update the feasible intercept postures all the time. However, it usually does not affect the convergence of the system as the new feasible intercept point is chosen in the vicinity of the previous ones; i.e. the convergence duration is too small. Besides, thanks to the second order LPV dynamical system, the updating does not cause discontinuity at the velocity profile.

Other compliant approaches should be considered when the mass of the object comes close to that of the robotic arm and hence that

the force at impact may require energy dissipation. In these cases, compliance through either hardware damping or active controlled damping would be necessary. In this paper, we considered systems where the mass of the object was very small in comparison to that of the arm and the force at impact was negligible in comparison to the robot's natural inertia.

Finally, we are currently improving the performance of the learned dynamical system by converting the non-convex optimization problem (17) to a convex version. By this, we can be sure that the performance of the dynamical system is optimal and it is not sensitive to the initializations.

APPENDIX A
PROOF OF THEOREM 1

We propose a Lyapunov function as follows:

$$V = \frac{1}{2}(\dot{\xi} - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O)^T(\dot{\xi} - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O) - \int_0^{\xi-\gamma\xi^O} \left(y^T \sum_{k=1}^{K_1} \theta_{A_k^1}(y) A_k^1 \right) \mathrm{d}y \tag{19}$$

V is positive definite, radially unbounded, continuous and continuously differentiable. The derivative of V with respect to time is as follows:

$$\dot{V} = \frac{dV}{dt} = (\dot{\xi} - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O)^T(\ddot{\xi} - \gamma\ddot{\xi}^O - 2\dot{\gamma}\dot{\xi}^O - \ddot{\gamma}\xi^O) - (\xi - \gamma\xi^O)^T \sum_{k=1}^{K_1} \theta_{A_k^1}(\xi - \gamma\xi^O) A_k^1 (\dot{\xi} - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O) \tag{20}$$

Substituting (5) into (20), we have:

$$\begin{aligned}
\dot{V} &= (\dot{\xi} - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O)^T \sum_{k=1}^{K_2} \theta_{A_k^2}(t) A_k^2 (\dot{\xi} - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O) \\
&+ (\dot{\xi} - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O)^T \sum_{k=1}^{K_1} \theta_{A_k}(\xi - \gamma\xi^O) A_k^1 (\xi - \gamma\xi^O) \\
&- (\xi - \gamma\xi^O)^T \sum_{k=1}^{K_1} \theta_{A_k^1}(\xi - \gamma\xi^O) A_k^1 (\dot{\xi} - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O) \\
&= \sum_{k=1}^{K_2} \underbrace{\theta_{A_k^2}(t)}_{>0} \underbrace{(\dot{\xi}^R - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O)^T A_k^2 (\dot{\xi}^R - \gamma\dot{\xi}^O - \dot{\gamma}\xi^O)}_{<0} \\
&< 0
\end{aligned} \tag{21}$$

Therefore, dynamical system (5) is globally stable; i.e. ξ and $\dot{\xi}$ are bounded as ξ^O, $\dot{\xi}^O$, γ and $\dot{\gamma}$ are bounded. Since $\ddot{V}$ is finite, Barbalet's lemma indicates that the attractor is globally asymptotically stable; i.e:

$$\lim_{t\to\infty} \|\dot{\xi} - (\gamma\dot{\xi}^O + \dot{\gamma}\xi^O)\| = 0, \qquad \lim_{t\to\infty} \|\xi - \gamma\xi^O\| = 0 \tag{22}$$

■

To conclude, Theorem 1 is proved.

ACKNOWLEDGEMENT

This work was supported by EU projects *AlterEgo* (grant #600610) and *Cogimon* $H2020 - ICT - 23 - 2014$. The authors kindly thank Alireza Karimi for his insightful comments during the development of this work and Seungsu Kim and Ashwini Shukla for preparing the experimental setup.

REFERENCES

[1] W. Hong and J. J. E. Slotine, "Experiments in hand-eye coordination using active vision," in *Lecture Notes In Control And Information Sciences*. Springer-Verlag, 1995, pp. 130–139.

[2] G.-R. Park, K. Kim, C. H. Kim, M.-H. Jeong, B.-J. You, and S. Ra, "Human-like catching motion of humanoid using evolutionary algorithm(EA)-based imitation learning," in *IEEE International Workshop on Robot and Human Interactive Communication*, 2009, pp. 809–815.

[3] J. Kober, K. Muelling, and J. Peters, "Learning throwing and catching skills," in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 5167–5168.

[4] J. Kober, M. Glisson, and M. Mistry, "Playing catch and juggling with a humanoid robot," in *IEEE-RAS International Conference on Humanoid Robots*, 2012, pp. 875–881.

[5] V. Lippiello, F. Ruggiero, and B. Siciliano, "3D monocular robotic ball catching," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1615–1625, 2013.

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

[7] R. Mori, K. Hashimoto, and F. Miyazaki, "Tracking and catching of 3D flying target based on gag strategy," in *IEEE International Conference on Robotics and Automation*, vol. 5, 2004, pp. 5189–5194.

[8] K. Deguchi, H. Sakurai, and S. Ushida, "A goal oriented just-in-time visual servoing for ball catching robot arm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 3034–3039.

[9] T. Senoo, Y. Yamakawa, S. Mizusawa, A. Namiki, M. Ishikawa, and M. Shimojo, "Skillful manipulation based on high-speed sensory-motor fusion," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 1611–1612.

[10] M. Matsushima, T. Hashimoto, M. Takeuchi, and F. Miyazaki, "A learning approach to robotic table tennis," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 767–771, 2005.

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

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

[13] 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.

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

[15] D. L. Ly and H. Lipson, "Learning symbolic representations of hybrid dynamical systems," *Journal of Machine Learning Research*, vol. 13, pp. 3585–3618, 2012.

[16] C. Hoffmann and H. Werner, "A survey of linear parameter-varying control applications validated by experiments or high-fidelity simulations," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 416–433, March 2015.

[17] B. Bamieh and L. Giarré, "Identification of linear parameter varying models," *International Journal of Robust and Nonlinear Control*, vol. 12, no. 9, pp. 841–853, 2002.

[18] Z. Emedi and A. Karimi, "Fixed-structure LPV Discrete-time Controller Design with Induced l2-Norm and H2 Performance," *International Journal of Contol*, 2015.

[19] M. C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.

[20] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory And Algorithms*. Wiley-Interscience, 2006.

[21] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Comparative experiments on task space control with redundancy resolution," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 3901–3908.

[22] A. Shukla and A. Billard, "Coupled dynamical system based arm-hand grasping model for learning fast adaptation strategies," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 424–440, 2012.

[23] S. Kim and A. Billard, "Estimating the non-linear dynamics of free-flying objects," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1108–1122, 2012.

[24] M. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in Optimization and Numerical Analysis*. Springer Netherlands, 1994, vol. 275, pp. 51–67.

[25] S. G. Johnson. The NLopt nonlinear-optimization package. [Online]. Available: http://ab-initio.mit.edu/nlopt