Robotics and Autonomous Systems 60 (2012) 1108–1122

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Estimating the non-linear dynamics of free-flying objects

Seungsu Kim *, Aude Billard

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

ARTICLE INFO

Article history:
Received 16 January 2012
Received in revised form
14 May 2012
Accepted 25 May 2012
Available online 7 June 2012

Keywords:
Machine learning
Dynamical systems

ABSTRACT

This paper develops a model-free method to estimate the dynamics of free-flying objects. We take a realistic perspective to the problem and investigate tracking accurately and very rapidly the trajectory and orientation of an object so as to catch it in flight. We consider the dynamics of complex objects where the grasping point is not located at the center of mass.

To achieve this, a density estimate of the translational and rotational velocity is built based on the trajectories of various examples. We contrast the performance of six non-linear regression methods (Support Vector Regression (SVR) with Radial Basis Function (RBF) kernel, SVR with polynomial kernel, Gaussian Mixture Regression (GMR), Echo State Network (ESN), Genetic Programming (GP) and Locally Weighted Projection Regression (LWPR)) in terms of precision of recall, computational cost and sensitivity to choice of hyper-parameters. We validate the approach for real-time motion tracking of 5 daily life objects with complex dynamics (a ball, a fully-filled bottle, a half-filled bottle, a hammer and a pingpong racket). To enable real-time tracking, the estimated model of the object's dynamics is coupled with an Extended Kalman Filter for robustness against noisy sensing.

1. Introduction

Catching rapidly moving targets remains a challenging task for a robot. To catch a moving object effectively, one must predict its trajectory ahead of time. This requires both accurate sensing and accurate modeling of the object's trajectory. A complex object is particularly challenging to track correctly as one can neither rely on a predetermined model of the dynamics nor expect to be provided with sufficiently accurate sensing to detect the object precisely. However, despite these difficulties, tracking and catching objects rapidly is a competence that will be expected from robots which work with humans in a real-world setting.

Humans are particularly skilled at performing fast catching motion, even though the speed at which they process visual information and move their limbs is often an order of magnitude smaller than the speed at which robot can perform both tasks. For instance, Cesqui et al. [1] reported a preparation time before onset of arm motion of for ball catching is 0.12 s. Brener and Smeets [2] showed that it takes about 0.11 s for visual information to influence the movement of the human hand. In this work, we seek to exceed human's ability to catch fast flying object and target a speed of visual processing of the order of a few milliseconds.

While robots are very powerful, they remain limited in the maximal velocity, acceleration and joint torque they can achieve. For instance, the WAM robot in our lab, which is the fast machine at our disposal, needs about 0.48 s. to catch a target a meter away (when starting with zero velocity and stopping at the target).[1] Hence, in order to reach the catching position in time, the robot should start moving at least 0.48 s before impact. Prediction of the displacement of the target will be strongly affected by noise in the sensing device. This will further delay the catching motion, as the arm has to be redirected toward the new estimated location on the fly. We hence seek a system that can make a rough trajectory estimate to be followed by the object in a few milliseconds after detection of the object, so as to initiate the movement in the right direction as soon as possible. The system will then continuously refine its estimate of the object's trajectory, as it receives more precise sensing information. This very fast re-computation of the object's motion can then be used to servo the arm to the right location.

In this paper, we ignore the problem of controlling the robot's body to catch the object and focus on the problem of accurately predicting the grasping point (point of interest). The grasping point is usually not located at the center of mass of the object, hence we tackle the problem of tracking an arbitrary point on the object, hereafter referred to as the *grasping point*. Note that we do not address the problem of recognizing the object in a video image and

* Corresponding author.
E-mail addresses: seungsu.kim@epfl.ch (S. Kim), aude.billard@epfl.ch (A. Billard).

[1] The WAM robot is made by Barrett technology inc. It manipulates in 3.5 m^3 work volume; The peak end-effector velocity is 3 m/s and maximum acceleration is 20 m/s^2 with 1 kg load; The repeatability of the robot is 2 mm.

0921-8890/$ – see front matter
doi:10.1016/j.robot.2012.05.022

hence assume that the grasping point has already been determined or is tagged.

We investigate successfully grasping five objects (a ball, a fully-filled and half-filled bottle, a hammer and a ping–pong racket) fairly common in human environments (see Fig. 3). These objects have asymmetric moment of inertia matrices, which result in highly non-linear translational and rotational motions of the grasping point. As we cannot rely on a known analytical model of the motion of the grasping point, we approach the problem of learning the dynamics of motion of the grasping point through demonstrations. To this end, we gather data from a human throwing each object several times (see Fig. 3). In recent years, machine learning has led to the development of various techniques for non-linear regression and prediction of complex time series. There is no definitive way to determine which non-linear regression technique would be most appropriate to learn an estimate of the non-linear dynamics of our objects in flight. Therefore, we chose to do a comparative analysis across the six most appropriate techniques, namely SVR with RBF kernel, SVR with polynomial kernel, GMR, ESN, GP and LWPR, according to three criteria: precision of recall, computational cost and sensitivity to choice of hyper-parameters.

Metrics

2. Related work

2.1. Robotic catching

A body of work has been devoted to autonomous control of fast movements such as catching [3–10], dynamic re-grasping (throwing an object up and catching it) [11], hitting flying objects [12,13] and juggling [14–16]. Most approaches assumed a known model of the dynamics of motion and considered solely modeling the translational object motion. For instance, Hong and Slotine [4] and Riley and Atkeson [7] model the trajectories of a flying ball as a parabola, and subsequently recursively estimate the ball's trajectory through least squares optimization. Frese et al. [6], Bauml et al. [10] and Park et al. [9] also assume a parabolic form for the ball trajectories which they use in conjunction with an Extended Kalman filter [17] for on-line re-estimation. Ribnick et al. [18] and Herrejon et al. [19] estimate 3D trajectories directly from monocular image sequences based on the ballistic ball movement equation.

A few works consider modeling of the rotational motion of the object for robot catching. Furukawa et al. [11] performed a dynamic regrasping experiment (throwing an object up and catching it) with a known axially symmetric object, a cylinder (with known radius and length and uniform mass distribution). Further, they assume that the center of mass (COM) of the cylinder follows a parabolic trajectory and set the rotational velocity to a constant. These assumptions are very effective and allowed them to perform an impressive demonstration of extremely fast grasping using a multi-fingered hand robot.

Such approaches are accurate at estimating the trajectories of fast-flying objects. However, they rely on an analytical model for the object. In addition, the majority of these works assume a ballistic translational trajectory and focus on tracking the object's center of mass. However, to catch a complex object such as a hammer or a ping–pong racket, one needs to estimate the complete pose of the object so as to determine the precise location of the grasping posture. This grasping posture is usually not located on the object's center of mass.[2] When the object is undergoing complex rotational motion, the dynamics of an arbitrary point not located on the COM follows very complex and arbitrary dynamics. Given this complexity, we examine existing approaches to estimating such complex non-linear dynamics when using a known (or good approximation) model of the dynamics.

2.2. Modeling a dynamical system

While there is a great interest in endowing a robot with the ability to catch objects in flight, the estimation of an unknown object's dynamics has been primarily the focus of applications in aerospace engineering for which estimating a spacecraft's orbit in real-time is a common issue. One of the methods to estimate a spacecraft's orbit is using precise known physical models of the forces acting on the spacecraft (e.g. earth gravity, atmospheric drag, lunisolar gravity, solar radiation pressure, etc.) [20–23]. However, precise modeling of all forces acting on each spacecraft is a very complicated process, and it still contains modeling errors. To reduce orbit errors caused by the mismodeling of spacecraft forces, the deterministic dynamics model of the spacecraft is complemented by empirical forces which are estimated by least-squares estimation or Kalman filtering with measurement geometric information [24,25]. However, a precise dynamic model is still required to produce a satisfactory orbit. Besides, such a model estimates solely the translational motion of the spacecraft.

System identification refers to a vast field in which the complex dynamics of a system is being estimated. This usually assumes a known model of the dynamics and only the open parameters of the model are estimated. Several approaches are applied for estimating parameters of a linear or nonlinear dynamical model, using the recursive least squares method [26], maximum likelihood [27], neural networks [28] or similar. These identifications of a dynamical model are extended to a linear parameter varying model (LPV). Each state space parameter of the LPV system is not a constant; instead, it is a function of the exogenous parameter which is predefined, measured or estimated upon operation of the system [29]. To identify the model, a Gauss–Newton type of gradient search method [30], hybrid linear/nonlinear procedure (where a model is divided into two parts, the nonlinear part is identified through neural network and the linear parametric part through an LS algorithm), recursive least-squares (RLS) algorithm [31] etc. are proposed. These methods very accurately model very complex systems. However, the type and structure of the system should be given ahead of time.

2.3. Machine learning techniques for modeling robotic tasks

Machine learning techniques have been applied in a variety of robotic tasks that require estimations of non-linear regressive models. Support Vector Regression (SVR) is used to model an inverse-dynamics of a robot [32], and to obtain an acceptable model of a chaotic system to be controlled [33]. Genetic Programming (GP) [34], a special class of genetic algorithm (GA), is used to model the optimum gait generation for a quadruped robot [35], and to establish an error correction (calibration) model of ill-conditioned inverse kinematics for a robot [36]. An Echo State Network (ESN) is used for nonlinear dynamical modeling [37], and modeling the localization capability and the navigation skill for a mobile robot. [38]. The Hidden Markov model (HMM) is used for recognition and regeneration of human motion across various demonstrations [39,40], and to teach a robot to perform assembly tasks [41]. Locally Weighted Projection Regression (LWPR) is used to approximate the dynamic model of a robot arm for computed torque control [42,32], for teaching a robot to perform basic soccer skills [43], and is also applied for real-time motion learning for a humanoid robot [32]. Statistical approaches that find a locally

[2] In our experiments, the grasping point will be represented by a constant transformation from the center-of-mass (COM) frame and may not lie on the object itself.

optimal model to represent the demonstrations by maximizing the likelihood, are also possible. They transform the demonstrations into a single or mixed distribution. To approximate the dynamic model of a robot arm for computed torque control [42,44], Gaussian Process Regression (GPR) is used. Additionally, Gaussian Mixture Regression (GMR) is used to model a robot motion from a set of human demonstrations, as a time-dependent model [45] or time-independent dynamical system [46].

The current works whose approaches most closely approximate our own at this time are approaches in image processing that track and predict the motion of an object using dynamical systems modeling. For instance, North et al. [47] model the deformation of the contour of an object as a 2nd order dynamical system (DS). The parameters of the dynamics are estimated through Expectation Maximization. However, this dynamics learning method is limited to simple ballistic motion. In a consecutive publication, North et al. [48] explore multi-class dynamics learning to model the motion of a juggler's ball. The motion is segmented with several dynamical systems, and each dynamical system is modeled as an Auto-Regressive Process (AR) [49]. AR is a simple linear time-series process, shape of high-order linear dynamical system; the future state is expressed as a linear combination of the past states. The model can be trained using the Yule–Walker Equation [49]. Pavlovic et al. [50] proposed using a mixed-state dynamic Bayesian network framework to model human gesture trajectories. Under this method, a hidden Markov model of discrete states is coupled with a first-order linear dynamical system of continuous trajectory motion. These methods assume that each discrete state of a target motion has a different level of thrust by its operator. In other words, they assume the target motion is a combination of different dynamics. However, to predict the whole trajectory along the different dynamics states, a smooth transition procedure between the discrete states is required.

2.4. Our approaches and contributions

As we reviewed above, several methods have been proposed to model the non-linear dynamics of objects in flight. Most of these methods rely on prior knowledge of the system as well as precisely known physical properties or analytic descriptions of the object model. This limits the application of these methods to a relatively narrow range of objects. In this work, we offer an approach to learn the dynamics of motion of an arbitrary point on an arbitrarily shaped object. We make no assumptions about the position of the center of mass nor do we use a closed form description of the dynamics and shape of the object.

In our previous work [51], we addressed the problem of learning and predicting the motion of a flying ball in order to catch it in-flight using a humanoid robot. There, the dynamic model of the ball motion was learned using the Gaussian Mixture Model (GMM) based dynamical system estimator. Only the translational velocity of the center of mass of the ball was estimated. In this paper, we extend our previous work in two ways: we consider more complex objects with complex distribution of mass (resulting in asymmetrical inertia matrices) and we no longer assume that the grasping point is located at the center of mass of the object. We encode the demonstrations using 2nd order DS which provides an efficient way to model the dynamics of a moving object solely by observing examples of the object's motion in space. Note that this does not require prior information on the physical properties of the object such as its mass, density, etc. To model a DS from observations, we use machine learning techniques for non-linear regression estimation. As reviewed in Section 2.3, there are a variety of techniques for non-linear regression. Here we focus our comparative analysis across the six most appropriate techniques, namely SVR–RBF, SVR–Poly, GMR, ESN, GP and LWPR. We do not consider GPR and HMM for two different reasons. Since GPR relies on storing all data points for testing, it is deemed too computationally intensive.[3] HMM, on the other hand, requires some interpolation methods to regenerate an estimate of the trajectory (in between two points generated by the observations associated to each state) during recall. Typically, one would use a spline technique for the interpolation. This however can significantly deteriorate the predictive value of the model. We hence prefer to use methods that directly estimate the non-linear dynamics of the system and do not rely on secondary methods for regenerating this dynamics.

The rest of this paper is divided as follows. In Section 3, we present the formalization of our method. We show that SVR, GMR, ESN, GP and LWPR yield an analytical form for the estimated dynamics which can be used in conjunction with extended Kalman filter for real-time tracking of the grasping point. In Section 4, we present a set of experiments in which we contrast the six regression methods in terms of precision of recall, computational cost and, most importantly, sensitivity to choice of hyper parameters. Section 5 concludes and Section 6 looks out on remaining issues that will be tackled in future work.

3. Predicting the trajectory of a free-flying object

3.1. Learning the dynamics of a moving object

In its most generic form, the dynamics of a free-flying object follows a 2nd order autonomous dynamical system:

$$\ddot{\xi} = f\left(\xi, \dot{\xi}\right) \tag{1}$$

where, $\xi \in \mathbb{R}^D$ denotes the state of the object (position and orientation vector of the grasping point attached on the object, see Fig. 3.); $\dot{\xi} \in \mathbb{R}^D$ and $\ddot{\xi} \in \mathbb{R}^D$ denote the first and second derivative of ξ; A total of N training trajectories with T data points are used to model the dynamics, $\left\{\left\{\xi_{t,n}, \dot{\xi}_{t,n}, \ddot{\xi}_{t,n}\right\}_{t=1..T}\right\}_{n=1..N}$. We used Quaternions to represent orientations. Quaternion representation avoids the gimbal lock problem and numerical drift compared to Euler angles, and allows for more compact representation than rotational matrices.

In this paper, we evaluate four different machine learning techniques to model unknown function $f(\cdot)$. These are Support Vector Regression (SVR) [54], Gaussian Mixture Regression (GMR) [55], Echo State Network [56], Genetic Programming [34] and Locally Weighted Projection Regression [32]. Each of these algorithms are briefly described next. For clarity of notation in the rest of the document, we regroup the state variable in a single variable: $\zeta = \begin{bmatrix} \xi \\ \dot{\xi} \end{bmatrix} \in \mathbb{R}^{2D}$.

3.1.1. Support Vector Regression (SVR)

SVR [54] performs non-linear regression from a multidimensional input ζ to a unidimensional output. Since our output is multidimensional, we here train D SVR models, denoted ${}^d f_{\text{SVR}}, d = 1 \ldots D$. After training, we obtain a regression estimate given by:

$$\ddot{\xi} = f_{\text{SVR}}(\zeta) = \left[{}^d f_{\text{SVR}}(\zeta)\right]_{d=1\ldots D} \tag{2}$$

$${}^d f_{\text{SVR}}(\zeta) = \sum_{m=1}^{M} {}^d\alpha_m K\left(\zeta, {}^d\zeta_m\right) + {}^d b. \tag{3}$$

[3] Recent efforts at developing spare methods for alleviating this cost have produced promising results [52,53]. The computation remains however at best linear in the number of data points and there is no unique way to determine the subset of training points.

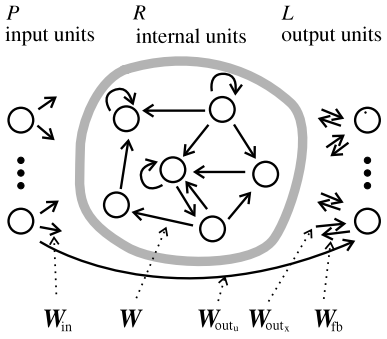


Fig. 1. The basic network architecture of echo state network [56].

Only the subset of data points, $\zeta_m \in \mathbb{R}^{2D}, m = 1 \ldots M, M \leq (N \times T)$ are used in the regression. These are called the support vectors, and have associated coefficient $\alpha_m \neq 0, |\alpha_m| < \frac{C}{M}$. C is a regularized constant that determines a trade-off between the empirical risk and the regularization. In this paper, the optimal value of C is determined through a grid search, see Section 4.2. The kernel function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is a non-linear metric of distance across data points. It is key to so-called kernel machines such as SVR and allows to extract features across data points that would not be visible through Euclidian metrics, such as norm 2. The choice of kernel is therefore crucial. In this study, we use the inhomogeneous order-p polynomial kernel, $K(\zeta, \zeta_m) = (\zeta \cdot \zeta_m + 1)^p$, and the radial basis function (RBF) kernel, $K(\zeta, \zeta_m) = \exp\left(-\gamma \|\zeta - \zeta_m\|^2\right)$ with radius $\gamma \in \mathbb{R}$ and determine the optimal values for the open parameters of these kernels through grid search.

3.1.2. Gaussian Mixture Regression (GMR)

GMR [55] is a multivariate nonlinear regression method. First one learns an estimate of the joint distribution $\mathcal{P}(\zeta, \ddot{\xi})$ using a Gaussian Mixture Model trained through Expectation–Maximization (EM). Then, one derives an analytical expression for the regression signal by computing $f_{\text{GMR}}(\zeta) = E\left[\mathcal{P}(\ddot{\xi}|\zeta)\right]$. Assuming a GMM with K Gaussian functions, f_{GMR} is given by:

$$\ddot{\xi} = f_{\text{GMR}}(\zeta) = \sum_{k=1}^{K} h_k(\zeta)\,(A_k\zeta + B_k) \tag{4}$$

$$A_k = \Sigma_k^{\ddot{\xi},\zeta}\left(\Sigma_k^{\zeta}\right)^{-1} \tag{5}$$

$$B_k = \mu_k^{\ddot{\xi}} - A_k\mu_k^{\zeta} \tag{6}$$

$$h_k(\zeta) = \frac{\mathcal{N}(\zeta|\mu_k^{\zeta}, \Sigma_k^{\zeta})}{\sum_{i=1}^{K}\mathcal{N}\left(\zeta|\mu_i^{\zeta}, \Sigma_i^{\zeta}\right)} \tag{7}$$

$$\Sigma_k = \begin{bmatrix} \Sigma_k^{\zeta} & \Sigma_k^{\ddot{\xi},\zeta} \\ \Sigma_k^{\ddot{\xi},\zeta} & \Sigma_k^{\ddot{\xi}} \end{bmatrix} \tag{8}$$

where $\mathcal{N}(\zeta|\mu_k, \Sigma_k)$ is a Gaussian distribution with mean μ_k and covariance matrix Σ_k; $h_k(\zeta)$ gives a measure of the influence of the kth Gaussian in generating the data point ζ; see [57] for details.

3.1.3. Echo State Network (ESN)

ESN [56] is a recurrent neural network composed of one hidden layer with randomly connected and weighted neurons, see Fig. 1.

We used a fully connected ESN with P input nodes for the $2D$-dimensional input, R internal units and L output unit. The weights W_{in} from input to internal units and W across internal units were set to random initial values, but the spectral radius of W is $\rho < 1$ (the optimal value of ρ is determined through a grid search). Only the output weights from internal units to output units are learned through linear regression; see [56] for a complete description of ESN. Eq. (9) gives the expression for f_{ESN}, the estimate of f using ESN.

$$\ddot{\xi} = f_{\text{ESN}}(\zeta) = g\left(\mathbf{W}_{\text{out}_x}\mathbf{x} + \mathbf{W}_{\text{out}_u}\zeta\right) \tag{9}$$

$$\dot{\mathbf{x}} = \frac{1}{c}\left(-a\mathbf{x} + g\left(\mathbf{W}_{\text{in}}\zeta + \mathbf{W}\mathbf{x} + \mathbf{W}_{\text{fb}}\ddot{\xi}\right)\right) \tag{10}$$

where $\mathbf{x} \in \mathbb{R}^R$ is the state of the internal neurons; c is the time constant; a is the decay rate; $g(.)$ represents the internal unit's output function, and output unit's output function (in this paper we used identity function); $\mathbf{W}_{\text{in}} \in \mathbb{R}^{R\times P}$ is input weight matrix; $\mathbf{W} \in \mathbb{R}^{R\times R}$ is the reservoir weight matrix; $\mathbf{W}_{\text{out}_x} \in \mathbb{R}^{L\times R}$ is the output weight matrix connected to internal state $\mathbf{x}$; $\mathbf{W}_{\text{out}_u} \in \mathbb{R}^{L\times P}$ is the output weight matrix connected to input; $\mathbf{W}_{\text{fb}} \in \mathbb{R}^{R\times L}$ is output feedback matrix.

3.1.4. Genetic Programming (GP)

GP [34] is a special class of genetic algorithms (GA) where each individual is a program expressed as a syntax tree. GP automatically determines both the structure and the parameters of the program. Each program codes for variables or a set of functions from a function set. The function-set used in this paper, consists of the arithmetic operators ($+, -, \times$ or/) and the trigonometric functions ($\sin(\cdot), \cos(\cdot)$). Fig. 2 shows an example of a function f_{GP} yielded by GP. Algorithm 1 shows steps involved in learning a GP in this study. See [34] for details.

Algorithm 1 Basic Steps for Genetic Programming

1: Initialize the population of programs randomly.
2: **repeat**
3: *Fitness*: Calculate fitness (the root mean squared prediction error on the training data) for each program.
4: *Tournament selection*: Select one or two program(s) at random from the population based on the fitness (population size and maximum depth of program tree are found from grid search).
5: *Recombination and Mutation*: Create new programs or delete existing programs by applying crossover and mutation (mutation rate: 0.1).
6: **until** *Termination*: Terminate once the fitness is below a given threshold (0.001 in our experiments) or when more than 200 generations have been done.
7: Return the best program found.

3.1.5. Locally Weighted Projection Regression (LWPR)

LWPR [32] is a nonlinear function approximation algorithm. The LWPR function is constructed by combining weighted local linear models. LWPR is able to incrementally update the number of local models and regression parameters of a local model online as suggested in [58]. We hence obtain a D-dimensional estimate given by:

$$\ddot{\xi} = f_{\text{LWPR}}(\zeta) = \left[{}^d f_{\text{LWPR}}(\zeta)\right]_{d=1\ldots D} \tag{11}$$

$${}^d f_{\text{LWPR}}(\zeta) = \frac{\sum_{s=1}^{S} w_s\Psi_s}{\sum_{s=1}^{S} w_s} \tag{12}$$

$$w_s = \exp\left(-\frac{1}{2}(\zeta - \mathbf{c}_s)^T\,\mathbf{D}_s\,(\zeta - \mathbf{c}_s)\right). \tag{13}$$

$$^1\ddot{\xi} = {}^1 f_{GP}(\zeta) = 0.05644\zeta_6 \cos(\zeta_2)(\zeta_{10} + \zeta_6) - 0.04038\zeta_6(\zeta_{10} - 5.15) - 0.04038\cos(\sin(\zeta_{10})) + 0.113\zeta_5 \cos(\zeta_6{}^2)\sin(\zeta_6) - 0.06652\zeta_{10}\zeta_2\zeta_5\zeta_6{}^2 + 0.02483$$

Fig. 2. A program trained through a open-source software GPTIPs [34] for a dynamic model of a flying hammer at the experiment Section 4.3.

Here, $\Psi_s = \beta_s{}^T(\zeta - \mathbf{c}_s) + b_s$ is the sth linear local model. Each local linear regressive model is described by its regression parameter $\beta_s \in \mathbb{R}^{2D}$ and its intercept b_s. The weight w_s measures the influence of the sth local model given a query point ζ. $\mathbf{c}_s \in \mathbb{R}^{2D}$ is the center of the local model and $\mathbf{D}_s \in \mathbb{R}^{2D\times 2D}$ is the distance metric of the Gaussian kernel which determines the local influence of each linear regressive model. The number of local models S is determined during the incremental training. If all weights w_s of existing local models for a new training point are smaller than the weight activation threshold (0.1 in this study), a new local model is created. The center of the new local model is set to the current training point and it is not modified afterwards [58]. The model parameters β_s, b_s and $\mathbf{D}_s$ are trained through the incremental gradient descent method with Partial Least Squares regression (PLS) [59] to minimize prediction errors.

3.2. Dealing with noise

Each of the six algorithms we use to estimate our unknown object dynamics f yields an analytical expression for f, which is continuous and continuously differentiable. It can then be used in conjunction with the Extended Kalman Filter (EKF) [17] to ensure a robust noise filter when re-estimating the object's motion in flight. For the EKF, the augmented estimate f_{AUG} of the system dynamics (estimated using either of the six estimation techniques described earlier on) and measurement model are given by:

$$\dot{\zeta}(t) = f_{\text{AUG}}(\zeta(t)) + \mathbf{w}(t) \tag{14}$$

$$\mathbf{z}(t) = [\text{eye}(D)\ \text{zeros}(D)]\,\zeta(t) + \mathbf{v}(t) \tag{15}$$

$$f_{\text{AUG}}(\zeta(t)) = \begin{bmatrix} \dot{\xi}(t) \\ f(\zeta(t)) \end{bmatrix} \tag{16}$$

where $\mathbf{z}(t)$ is the measurement of the state at time t, the process noise $\mathbf{w}$ and measurement noise $\mathbf{v}$ are assumed to be multivariate zero mean Gaussian with covariance $\mathbf{Q_s} = E\left[\mathbf{w}\mathbf{w}^T\right]$ and $\mathbf{R_s} = E\left[\mathbf{v}\mathbf{v}^T\right]$ respectively. zeros(D) and eye(D) are $D \times D$ zero and identity matrix.

For the EKF prediction, we initialize the state of our system $\zeta(t_0) = E[\zeta(t_0)]$ and the error covariance $\mathbf{P}(t_0) = \text{Var}[\zeta(t_0)]$. EKF determines at each time step an estimate of the time derivatives of our systems $\dot{\hat{\zeta}}(t)$ and error covariance matrix $\dot{\hat{\mathbf{P}}}(t)$ using the following equation:

$$\dot{\hat{\zeta}}(t) = f_{\text{AUG}}(\zeta(t)) \tag{17}$$

$$\dot{\hat{\mathbf{P}}}(t) = \mathbf{F}(t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}(t)^T + \mathbf{Q}(t) \tag{18}$$

$$\mathbf{F}(t) = \left.\frac{\partial f_{\text{AUG}}}{\partial \zeta}\right|_{\zeta(t)} = \begin{bmatrix} \text{zeros}(D) & \text{eye}(D) \\ \left.\dfrac{\partial f}{\partial \zeta}\right|_{\zeta(t)} & \end{bmatrix}. \tag{19}$$

Here, $\mathbf{F}(t)$ is the Jacobian of the dynamics model, and $\mathbf{Q}(t)$ is the process covariance matrix at time step t. Since the selection of $\mathbf{Q}(t)$ does have a significant influence on the performance of EKF, we determine $\mathbf{Q}(t)$ using a fundamental matrix $\Phi(t)$ as seen below [60]:

$$Q(t) = \int_0^{\Delta t} \Phi(\tau)\,\mathbf{Q_s}\Phi(\tau)^T\,dt \tag{20}$$

$$\Phi(t) \approx \mathbf{I} + \mathbf{F}(t)\Delta t. \tag{21}$$

By integrating Eqs. (17) and (18) through time by Δt, we find the state estimate $\hat{\zeta}(t + \Delta t)$ and the error covariance estimate $\hat{\mathbf{P}}(t + \Delta t)$. After this predictive step, we calculate *a posteriori* state $\zeta(t + \Delta t)$ and error covariance $\mathbf{P}(t + \Delta t)$, by correcting the state and covariance using the following equations.

$$\mathbf{K}(t) = \hat{\mathbf{P}}(t + \Delta t)\,\mathbf{H}^T\left(\mathbf{H}\hat{\mathbf{P}}(t + \Delta t)\mathbf{H}^T + \mathbf{R_s}\right) \tag{22}$$

$$\zeta(t + \Delta t) = \hat{\zeta}(t + \Delta t) + \mathbf{K}(t)\left(\mathbf{z}(t) - \mathbf{H}\hat{\zeta}(t + \Delta t)\right) \tag{23}$$

$$\mathbf{P}(t + \Delta t) = (\mathbf{I} - \mathbf{K}(t)\mathbf{H})\,\hat{\mathbf{P}}(t + \Delta t) \tag{24}$$

$$\mathbf{H} = [\text{eye}(D)\ \text{zeros}(D)] \tag{25}$$

where $\mathbf{K}(t)$ is the Kalman gain, $\mathbf{H}$ is the Jacobian of measurement model.

To ensure the report's comprehensiveness, we provide the analytical expression of the derivatives of the functions $\frac{\partial f}{\partial \zeta}$ used for the prediction step (Eq. (19)) in Table 1.

4. Empirical validation

Xac nhan thuc nghiem

4.1. Experimental setup

4.1.1. Objects selection

To demonstrate the accuracy of the trajectory estimation methods and the performance with different machine learning techniques, we used five objects varying in complexity: a ball, a bottle fully-filled with water, a bottle half-filled with water, a hammer and a pingpong racket. Each of these objects are shown in Fig. 3. The Ball is the simplest object to catch due to its three axes of symmetry. One needs only to determine the position of the object, as the orientation is irrelevant. In contrast, to properly catch the bottle requires a determination of both the location and orientation of the bottle. When the bottle is half-filled with water, its moment of inertia changes as it rotates in flight, adding strong non-linearities to the dynamics. For a hammer, the measuring point which is typically at the center of the handle, is noticeably removed from the center of mass. Hence, the motion of the measuring point is highly oscillatory in nature. For the racket, the effect of air-drag is more important on the face than along the handle, which again is a source of considerable non-linearity.

4.1.2. Observing real object movement

The trajectories were recorded using the HAWK motion capture system, from *Motion Analysis*. To record both the position and orientation of the object in flight, we attached 3 markers to each of the objects. Each marker's position was recorded at 120 Hz. Each object was thrown 20 times. The recorded row data-sets and the position trajectories of 3 markers are converted to the position and orientation trajectories of the *grasping point*. And, each trajectory is filtered through a Butterworth filter at 25 Hz, and we calculated the velocity and acceleration using cubic spline interpolation. Finally, we normalized the linear and angular acceleration trajectories to be comprised within −0.5 and +0.5. Some of the position and orientation trajectories captured from motion capture system are shown in Fig. 3. The mean and standard deviation of the initial position, velocity, Euler angle and angular velocity of the thrown objects are shown in Tables 2 and 3. Additionally, their ranges are shown Table 4. The mean and standard deviation of acceleration and angular acceleration for the trajectories of the objects are shown in Table 5.

Table 1
Analytical expression of the derivatives of the functions $\frac{\partial f}{\partial \zeta}$.

Derivative of SVR

$$\frac{\partial f_{\text{SVR}}(\zeta)}{\partial \zeta} = \left[\frac{\partial^d f_{\text{SVR}}(\zeta)}{\partial \zeta}\right]_{d=1\ldots D} \quad (26)$$

$$\frac{\partial^d f_{\text{SVR-POLY}}(\zeta)}{\partial \zeta} = p\sum_{m=1}^{M} {}^d\alpha_m \left(\zeta \cdot {}^d\zeta_m + 1\right)^{p-1} \quad (27)$$

$$\frac{\partial^d f_{\text{SVR-RBF}}(\zeta)}{\partial \zeta} = -2\gamma\sum_{m=1}^{M} {}^d\alpha_m \left(\zeta - {}^d\zeta_m\right)\exp\left(-\gamma\left\|\zeta - {}^d\zeta_m\right\|^2\right) \quad (28)$$

where ζ_m are the support vectors with the associated coefficient α_m. p is the degree of polynomial kernel, γ is the radius of RBF kernel.

Derivative of GMR

$$\frac{\partial f_{\text{GMR}}(\zeta)}{\partial \zeta} = \sum_{k=1}^{K} h_k(\zeta)\left\{\begin{bmatrix} C_k(\zeta)\left(A_{k,1}\zeta + B_{k,1}\right) \\ \cdots \\ C_k(\zeta)\left(A_{k,j}\zeta + B_{k,j}\right) \\ \cdots \end{bmatrix} + A_k\right\} \quad (29)$$

$$C_k(\zeta) = \left\{-\left(\Sigma_\zeta^k\right)^{-1}\left(\zeta - \mu_\zeta^k\right) + \sum_{i=1}^{K}\left(h_i(\zeta)\left(\Sigma_\zeta^i\right)^{-1}\left(\zeta - \mu_\zeta^i\right)\right)\right\}^T \quad (30)$$

where $A_{k,j}$ is jth row value of A_k; $A_{k,j} \in \mathbb{R}^{1\times 2D}$; $B_{k,j} \in \mathbb{R}$.

Derivative of ESN

$$\frac{\partial f_{\text{ESN}}(\zeta)}{\partial \zeta} = W_{\text{out}_x}\begin{bmatrix} \frac{\dot{x}_1}{\dot{\zeta}_1} & \cdots & \frac{\dot{x}_1}{\dot{\zeta}_P} \\ \vdots & \ddots & \vdots \\ \frac{\dot{x}_R}{\dot{\zeta}_1} & \cdots & \frac{\dot{x}_R}{\dot{\zeta}_P} \end{bmatrix} + W_{\text{out}_u} \quad (31)$$

where $\mathbf{W}_{\text{out}_x}$ and $\mathbf{W}_{\text{out}_u}$ are the output weight matrixes connected to R internal units and P input nodes respectively; $\dot{x}_r$ is the velocity of rth internal state $\mathbf{x}$.

Derivative of GP

The dynamic model (program) trained through GP is defined as a linear combination of pre-defined set of functions/operators and the input variables. Since each program always is defined using finite number of individual closed form expressions, it will always possible to find the derivative for any given program of GP.

Derivative of LWPR

After finishing the training, the analytic derivative of the model is given by [58]:

$$\frac{\partial f_{\text{LWPR}}(\zeta)}{\partial \zeta} = \left[\frac{\partial^d f_{\text{LWPR}}(\zeta)}{\partial \zeta}\right]_{d=1\ldots D} \quad (32)$$

$$\frac{\partial^d f_{\text{LWPR}}(\zeta)}{\partial \zeta} = -\frac{\sum_{s=1}^{S}(-w_s\mathbf{D}_s(\zeta-\mathbf{c}_s))\sum_{s=1}^{S} w_s\Psi_s}{\left(\sum_{s=1}^{S} w_s\right)^2} + \frac{\sum_{s=1}^{S}\left(-w_s\mathbf{D}_s(\zeta-\mathbf{c}_s)y_s + w_s\beta_s{}^T\right)}{\left(\sum_{s=1}^{S} w_s\right)}. \quad (33)$$

Table 2
Initial position and velocity for the free-flying objects (mean ± st. dev.)

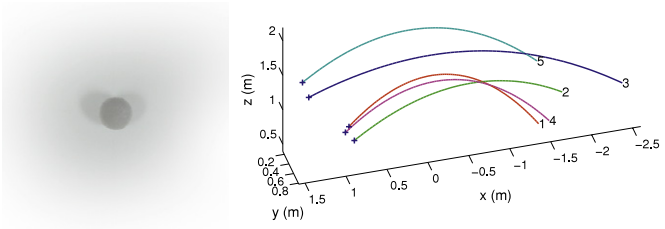
	Initial position (m)			Initial velocity (m/s)		
	x	y	z	x	y	z
Ball	1.20 ± 0.32	0.71 ± 0.05	1.08 ± 0.36	-3.84 ± 1.26	-0.31 ± 0.24	3.53 ± 0.83
Bottle full	0.84 ± 0.19	0.77 ± 0.03	0.99 ± 0.19	-3.15 ± 0.77	-0.19 ± 0.64	3.88 ± 0.91
Bottle half	0.89 ± 0.13	0.78 ± 0.06	0.95 ± 0.12	-3.56 ± 0.74	-0.33 ± 0.88	3.84 ± 0.60
Hammer	0.85 ± 0.19	0.70 ± 0.05	0.94 ± 0.10	-3.07 ± 0.59	-0.43 ± 0.22	3.26 ± 0.69
Racket	0.90 ± 0.18	0.69 ± 0.08	0.96 ± 0.23	-3.12 ± 0.93	-0.37 ± 0.55	3.47 ± 0.86

Table 3
Initial orientation and angular velocity of the free-flying objects (mean ± st. dev.)

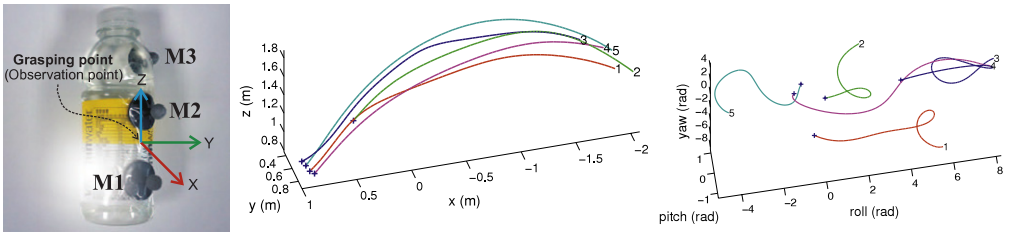
	Initial Euler angle ($\times 10^2$ °)			Initial angular velocity ($\times 10^3$ °/s)		
	Roll	Pitch	Yaw	ω_1	ω_2	ω_3
Bottle full	0.27 ± 1.37	-0.05 ± 0.30	0.00 ± 1.23	-0.19 ± 0.36	-0.06 ± 0.50	0.27 ± 0.33
Bottle half	-0.14 ± 1.57	-0.03 ± 0.31	0.13 ± 0.64	-0.28 ± 0.21	0.27 ± 0.39	0.24 ± 0.31
Hammer	0.05 ± 1.12	0.03 ± 0.51	-0.36 ± 1.38	-0.56 ± 0.38	0.19 ± 0.28	0.08 ± 0.25
Racket	0.31 ± 1.05	0.07 ± 0.35	-0.13 ± 1.17	-0.43 ± 0.21	0.23 ± 0.34	0.25 ± 0.29

Table 4
Range of the initial values (position, velocity, Euler angle and angular velocity) of the free-flying objects.

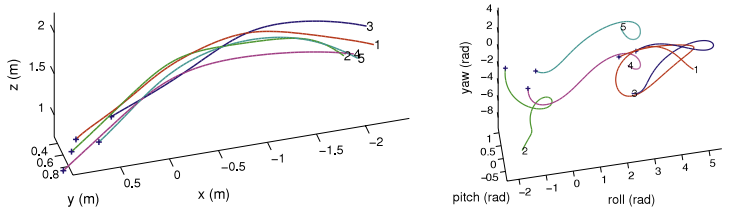
	Position (m)			Velocity (m/s)			Angle ($\times 10^2$ °)			ω ($\times 10^3$ °/s)		
	x	y	z	x	y	z	Roll	Pitch	Yaw	ω_1	ω_2	ω_3
Ball	0.90	0.15	0.98	4.16	0.66	2.89						
Bottle full	0.69	0.13	0.69	2.78	2.56	3.81	4.63	1.29	5.88	1.53	1.61	1.07
Bottle half	0.51	0.20	0.51	3.49	3.60	2.19	5.10	1.29	3.04	0.74	1.29	1.22
Hammer	0.62	0.17	0.31	2.18	0.82	2.87	3.59	1.48	5.97	1.78	1.24	1.05
Racket	0.74	0.29	1.01	3.75	1.82	3.21	3.12	1.30	3.42	0.74	1.18	0.95



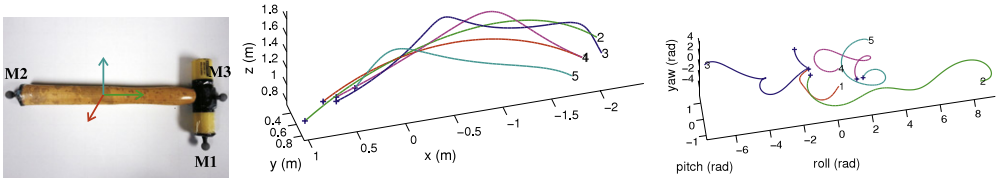
(a) A ball.



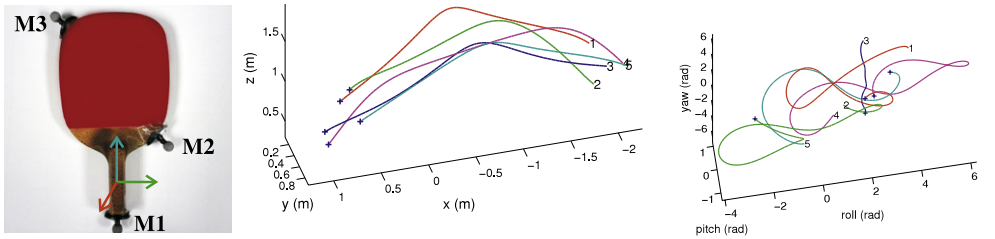
(b) A full-filled bottle.



(c) A half-filled bottle.



(d) A hammer.



(e) A racket.

Fig. 3. Left: Objects used in the experiments. Three markers (M1, M2 and M3) are attached to each of the objects (except the ball) to capture the position and orientation. Their position (middle) and orientation (right) trajectories are shown. "+" is marked in the graph at each initial position and orientation, and numbers are placed in the graph to identify each throwing trajectory.

Table 5
Acceleration and angular acceleration for the trajectories of the free-flying objects (mean ± st. dev.)

	Acceleration (m/s^2)			Angular acceleration ($\times 10^3\ °/s^2$)		
	x	y	z	α_1	α_2	α_3
Ball	0.42 ± 0.24	0.03 ± 0.08	-9.79 ± 0.09	–	–	–
Bottle full	-0.30 ± 5.41	-0.51 ± 3.66	-8.63 ± 5.36	-1.574 ± 2.462	0.445 ± 2.011	-0.378 ± 2.275
Bottle half	-0.96 ± 4.30	-0.18 ± 2.85	-9.95 ± 4.58	-1.384 ± 2.006	-0.570 ± 1.466	0.308 ± 2.100
Hammer	-0.39 ± 7.38	-0.37 ± 1.98	-10.40 ± 8.52	1.201 ± 3.037	1.392 ± 1.721	-0.395 ± 2.751
Racket	0.67 ± 7.60	0.38 ± 5.40	-8.15 ± 8.16	-1.140 ± 2.319	-2.015 ± 3.399	0.863 ± 3.613

Table 6
List and range of values of free-parameters for each method, and its optimum values after grid search.

Method	Parameter	Min	Max	Bottle(full)	Bottle(half)	Hammer	Racket
SVR	γ	0.01	0.60	0.16	0.12	0.20	0.24
(RBF)	C	0.10	3.00	3.00	3.00	3.00	3.00
SVR	p	1	6	6	6	6	6
(Poly)	C	0.10	3.00	3.00	3.00	3.00	3.00
GMR	K	1	12	10	11	12	12
ESN	R	20	80	80	80	80	80
	ρ	0.1	0.8	0.26	0.41	0.49	0.57
GP	Population size	10	150	150	94	94	122
	Maximum tree depth	2	5	5	4	5	4
LWPR	Initial D	0.1	3	1.4	1.1	1.1	1.4

4.2. Parameter selection

The performance of each technique is largely dependent on the selection of the model parameters. For SVR with RBF kernel, we tested the effect of changing the kernel width γ and the parameter C. γ is a scaling factor on the distance between the support vectors ζ_m and the query points ζ. C determines a tradeoff between increasing accuracy and optimizing for the complexity of the fit [61]. A small γ, likewise a high C will fit well with local non-linearities. However, this may lead to over-fitting when these non-linearities are not representative of the whole data-set. For SVR with polynomial kernel, we varied the value of C and the degree of the kernel p, as we expected that the higher p, the better the model would fit non-linearities. We varied the number of Gaussians K for GMR. The more Gaussians, the better the fit. But again, too many Gaussians would lead to over fitting and hence yield poor results on the testing set. For the ESN, we varied R, the number of internal neurons in the reservoir, and ρ, the spectral radius. The optimum value of R is task-dependent and hence there is no easy way to determine it off-hand. ρ is the spectral radius (maximum absolute value of the eigenvalues) of the reservoir weight matrix W and controls the spread of the weights dispersion in the reservoir. A large ρ implies that the reservoir stores longer ranges of nonlinear interaction with input components. In practice, one is advised to use $\rho < 1$ to ensure optimal performance of ESN [56]. For GP, we varied the population size, and depth of tree. A large population size usually yields better performance but increases computational cost. A high tree depth better fits local nonlinearities. For LWPR, we tested the effect of changing the initial distance metric $\mathbf{D}$, of the Gaussian Kernel (as varying r of $\mathbf{D} = r\mathbf{I}$ [58]). A small r will lead to local minima and delay the convergence, while a too large value for r will lead to over-fitting.

We performed 10-fold cross-validated grid searches, in order to find the optimal parameters for each technique, and each object. The range and optimal parameters found from the grid search are shown on Table 6.

For the ball, all techniques yielded very accurate estimates for a wide range of parameter choices (the prediction was hence little-influenced by the choice of parameters), see Fig. 4. As expected, the ballistics translational movement and the symmetry of the object along its three axes make the problem easy for all considered techniques.

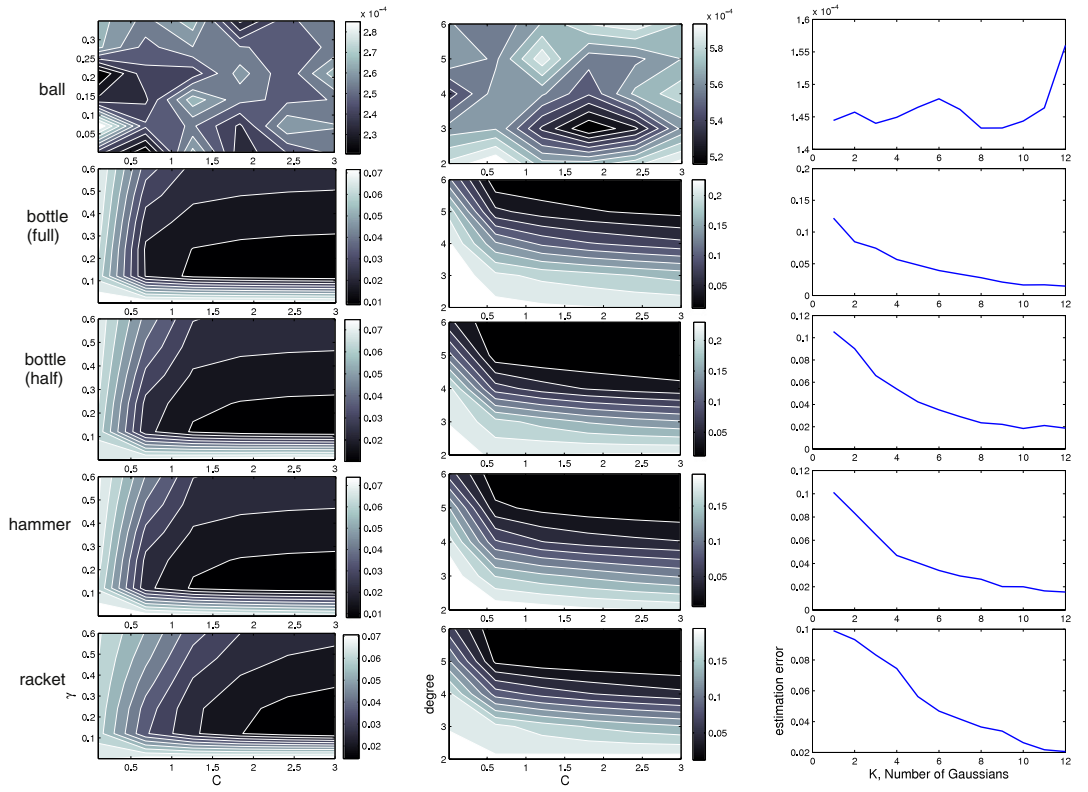
While we expected the methods to become more sensitive to the choice of parameters as the complexity of the object dynamics increased, we found that this is not the case and that similar optimal values for each set of parameters is found for each of the four more complex objects, see Table 6. There may be two reasons for this: first, even though the inherent complexity of the objects' dynamics increases, this was not perceivable when measuring only the position, orientation, velocity and angular velocity of the object (one should have looked at the displacement of water in the half-filled bottle for instance). Second, the dynamics of all 4 complex objects is far richer than the simple ballistic motion of the ball and thus the complexity of these objects' dynamics was somewhat comparable, requiring the same level of precision from the regression techniques to model these (observe that the scale and range of values taken by the input and output of our model during training are similar for each of the four complex objects, as seen in the Tables 2–5). Note that while the same range of optimal parameters is found for each of the four complex objects, the resulting regressive models differ in each case. Moreover, for SVR–RBF, SVR–Poly, ESN and LWPR, the region of parameter space yielding optimal performance is quite large, see Fig. 4. GP is the most sensitive technique and, as we will see next, it also had the lowest accuracy scores.

Decreasing the value of γ for RBF Kernel, improves the fit. As expected, when γ is too small, SVR starts overfitting leading to poor performance on the testing set. Near the optimal value of γ, we can see that the estimation accuracy is increased by increasing the penalty factor C. The SVR with polynomial kernel is not as sensitive to change in the value of the penalty term as the RBF kernel. We observed that increasing the order of polynomial resulted in a more accurate fit. We could not observe over-fitting as increasing the polynomial order greatly increases the computational cost and we hence had to restrict ourselves to a value for p no higher than 6. For GMR, according to the increased number of Gaussians, the estimation error is decreased and saturates at the end.[4] Note that we limited number of Gaussians for the same reason as for limiting the power of the polynomial kernel in SVR. Increasing the number of Gaussians greatly increase the computational cost (more than 10 min of training time) in EM, since we had already obtained very good results with a low number of Gaussians, there was little value to continue the search from the application standpoint (aside from demonstrating over-fitting). We hence decided to restrict our search to a number of Gaussians of no more than 12. For the case of GP, even though the accuracy is sensitive to parameter modifications, the accuracy tends to be increased by increasing the population size. For LWPR, decreasing the value of $\mathbf{D}$ improves the fit. As expected, when $\mathbf{D}$ is too small, LWPR starts overfitting leading to poor performance on the testing set.

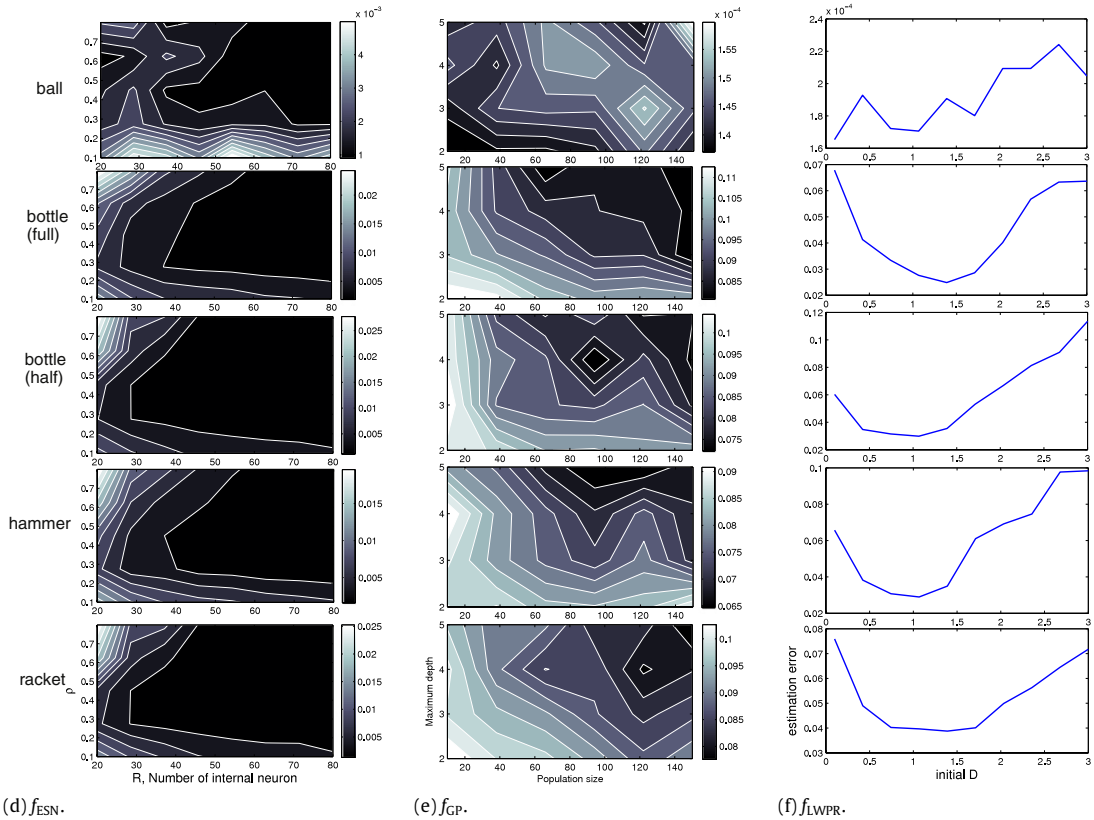
4.3. Comparison among different techniques for trajectory prediction

To compare the performance of different machine learning techniques to predict the object's trajectory, we proceeded in four

[4] Note that people will usually use iterative methods to determine the optimum number of Gaussians, whereby one incrementally increases the number of Gaussians and measures the resulting improvement in the likelihood. Criteria such as Bayesian Information Criterion (BIC) [62] or Aikaike Information Criterion (AIC) [63] can be used to determine when to stop increasing. These criteria determine a tradeoff between improving the likelihood and increasing dramatically the computation costs with the added number of parameters to estimate. Here we chose to not use these criteria and to perform a grid search on K so as to measure the sensitivity of GMR to the choice of K.



(a) $f_{\text{SVR-RBF}}$. (b) $f_{\text{SVR-POLY}}$. (c) f_{GMR}.



(d) f_{ESN}. (e) f_{GP}. (f) f_{LWPR}.

Fig. 4. Grid search on free parameters of each machine learning techniques. Error is calculated by the root mean squared prediction error on the testing output (for a ball, linear acceleration $\in \mathbb{R}^3$; for the others, linear and angular acceleration $\in \mathbb{R}^6$), each dimension of the testing output is normalized in range $[-0.5, 0.5]$. Note that, there is an order of magnitude difference in each plot.

Table 7
Root-mean-square error in estimation of translational acceleration for the testing set. The best methods are highlighted (unit: m/s^2).

–	Ball	Bottle(full)	Bottle(half)	Hammer	Racket
SVR–RBF	**0.001** ± 0.000	**0.044** ± 0.045	**0.099** ± 0.111	**0.104** ± 0.106	**0.092** ± 0.026
SVR–Poly	**0.003** ± 0.001	0.143 ± 0.109	0.221 ± 0.157	0.158 ± 0.157	0.271 ± 0.084
GMR	**0.002** ± 0.002	0.158 ± 0.136	0.276 ± 0.232	0.539 ± 0.286	0.266 ± 0.033
ESN	**0.004** ± 0.002	**0.038** ± 0.045	**0.050** ± 0.077	**0.099** ± 0.056	**0.059** ± 0.017
GP	**0.001** ± 0.001	0.089 ± 0.086	0.355 ± 0.301	0.438 ± 0.322	0.422 ± 0.322
LWPR	**0.001** ± 0.000	0.059 ± 0.028	0.108 ± 0.122	0.186 ± 0.192	0.203 ± 0.123

steps: First, we compared the accuracy of the different regression techniques to predict both linear and angular acceleration at all times. Second, we compared the techniques' accuracy at predicting the object's final position and orientation, by integrating over time the prediction of each model when provided solely with the initial position and orientation and their velocity. Third, we measured performance of the techniques when used in conjunction with EKF. Finally, we compared the calculation time for regression.

Note that, since ESN requires a series of observations (in contrast to the other methods which require a single initial position) to regulate the internal state of its reservoir, for each of the following experiments, we provided ESN with 0.05 s of data for the estimation.

Linear and angular acceleration estimation. We performed 10-fold cross-validation on all the trajectories of the thrown objects, 20 trajectories with 120 Hz capturing rate (1.5 s amount of data point for each trajectory). Tables 7 and 8 show the root-mean square error and its standard deviation for estimating acceleration and angular acceleration respectively. As we expected, all of the methods estimate the acceleration of the ball very well when given velocity, rotation and angular velocity. ESN shows outstanding performance at estimating acceleration. However, as mentioned above, ESN has some calculational advantages as it is provided with more information than the other techniques.

Prediction of final position and orientation. We predicted the future trajectory of the object for a given initial position and velocity (linear and rotational both), by recursively integrating the estimated acceleration and angular acceleration for each of the six models. The predictions were less accurate the further into the future we projected. To show this, we set the initial value as 1.0–0.1 s ahead of the value from the final value of the testing trajectory, and integrate without feedback from the demonstrated trajectory. Note that we partitioned all the trajectories of the thrown objects by 10 fold randomly, and performed 10-fold cross-validation (as in the above linear and angular acceleration estimation comparison). We plot in Fig. 5, the accuracy for predictions encompassing 1–0.1 s. Typical examples of trajectories predicted by each technique for each object are shown in Fig. 6.

Furthermore, to see how early the desired precision is achieved, we performed the same calculations as above but with a more precise time interval, 0.01 s; The results are in Table 9. Note that we set the desired precision for prediction as 1 cm in Cartesian space and 1° in Euler orientation. SVR–RBF could get the desired precision in both position and orientation around 0.17–0.32 s ahead. This is the desiderate to compensate lag of execution time for a robot as mentioned in the Introduction.

For the prediction comparison of 1 s lookahead for position and orientation, SVR–RBF shows the best performance. GMR and LWPR's performance decreases significantly as the time elapsed in the prediction increases. SVR–Poly, GP and ESN show the worst performance at tracking the position and orientation of the more complex objects. Predictions from SVR–RBF are the most accurate for all tests. ESN was the most accurate method for the first comparison (estimating the acceleration and angular acceleration for a given orientation and linear and angular velocity), as seen at Table 7. When we calculated a complete trajectory for given initial values by integrating the dynamics of ESN recursively, ESN did not yield good predictions of the objects final position and orientation. The reason is that the reservoir of ESN acts as a dynamical short-term memory which is influenced by traces of the past input history. When integrating the dynamics of ESN recursively, tiny estimation errors can be stored to the reservoir recursively, and change the dynamics f_{ESN} unexpectedly. This leads to poor prediction of the final position and orientation.

Table 8
Root-mean-square error in estimation of rotational acceleration for the testing set. The best methods are highlighted (unit: $^\circ/s^2$).

–	Bottle(full)	Bottle(half)	Hammer	Racket
SVR–RBF	**8.0** ± 4.7	**10.1** ± 4.6	**14.1** ± 5.5	**11.9** ± 4.7
SVR–Poly	37.9 ± 26.1	86.0 ± 58.8	80.3 ± 37.2	90.7 ± 52.7
GMR	16.6 ± 14.3	14.4 ± 14.7	20.4 ± 16.6	26.6 ± 14.5
ESN	**5.0** ± 5.6	**7.4** ± 4.2	**8.5** ± 5.3	**7.0** ± 4.6
GP	22.3 ± 15.3	40.0 ± 35.3	45.1 ± 36.9	45.5 ± 31.6
LWPR	24.5 ± 15.1	19.9 ± 13.4	23.8 ± 17.6	23.0 ± 13.3

Accuracy when using EKF. For the third comparison, we looked at how well each method tracked the object trajectories in noisy measurements in conjunction with EKF. We tested each method under 120, 60 and 30 Hz sampling rates, to see the robustness of each method under different sampling rates. Tables 10 and 11 show the RMS error and its standard deviation of all tracked objects (fully-filled bottle, half-filled bottle, hammer, racket). Fig. 7 gives a detailed example by tracking a hammer under the 30 Hz sampling rate. Note that filtered trajectories with a Butterworth filter at 25 Hz are considered as the rue measurement trajectories.

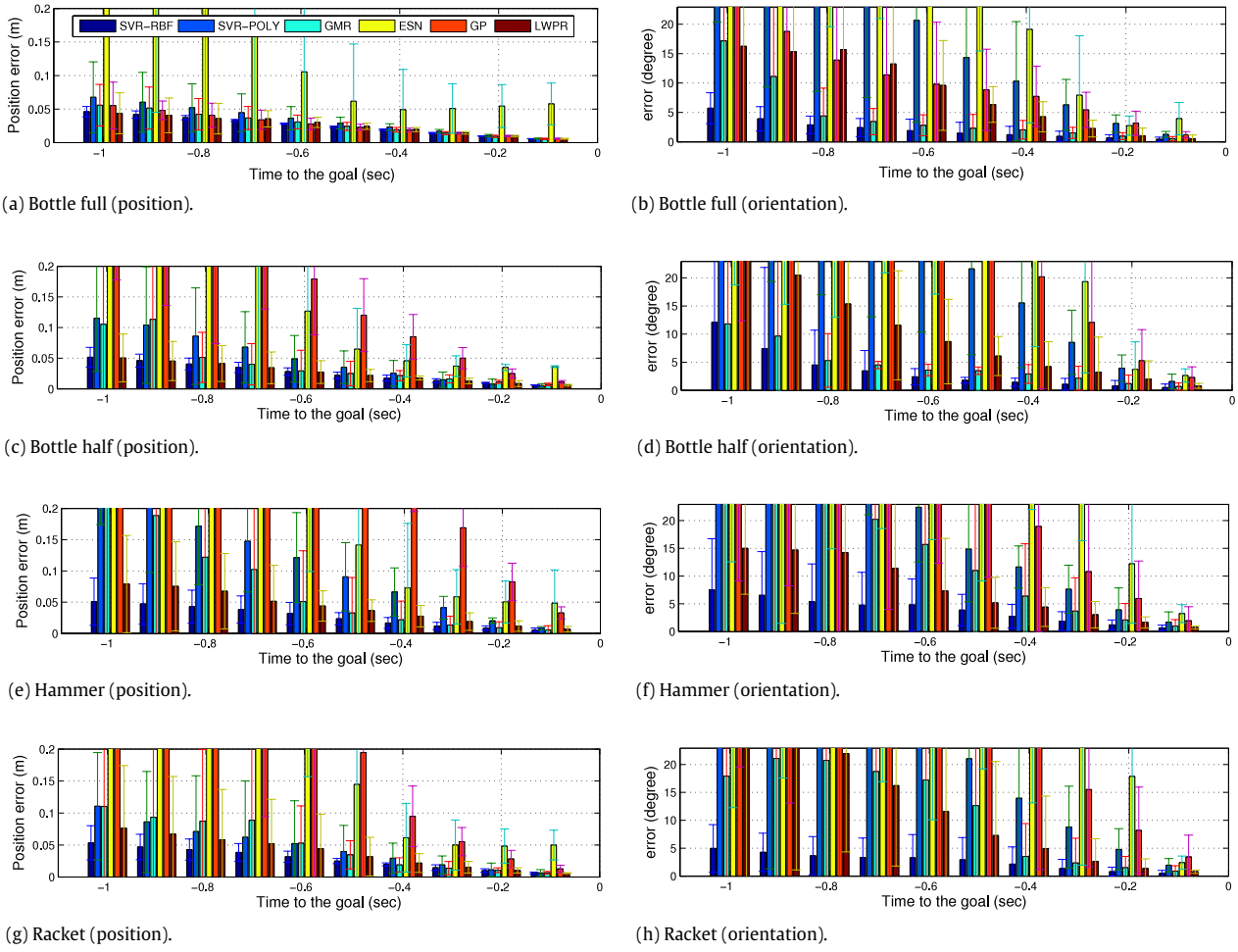
SVR–RBF and LWPR both show good performance for tracking when used in conjunction with EKF. SVR–Poly, GMR and GP continue having a very large tracking error, since their modeling error was quite big as we have seen in the first comparison. Note that we left out ESN in this comparison, since ESN in conjunction with EKF did not chase the measurement trajectory correctly. As we discussed in the second comparison, ESN dynamics is highly dependant on the input history. When integrating the dynamics of ESN, and using EKF to recursively correct the current state, the dynamics f_{ESN} and its derivative are changed unpredictably. This leads to poor tracking.

Comparison of required computational time. To implement the whole-trajectory prediction of an object in real-time, computation time is very important. We simulated the recall of object's dynamics by each of the algorithms for estimating 1.0 s trajectory at 120 Hz. The result of our computational time comparison is in Table 12. We conducted the simulation on a PC (CPU Dual-Core 3 GHz) with *Math-Kernel Library from Intel*.

The computational time of all methods for predicting 1 s duration of position and orientation trajectory is less than 8 ms. It is therefore acceptable for real-time prediction.

5. Conclusion

In this paper, we compared the performance of six techniques for non-linear regression and time-series prediction to model the



(a) Bottle full (position).
(b) Bottle full (orientation).
(c) Bottle half (position).
(d) Bottle half (orientation).
(e) Hammer (position).
(f) Hammer (orientation).
(g) Racket (position).
(h) Racket (orientation).

Fig. 5. By integrating the estimated acceleration and angular acceleration for each of the six trained regressive models, we predicted future position and orientation. We varied the initial value from 1.0 to 0.1 s value in the future from the final measurement of testing trajectories. We can see that SVR with RBF kernel performs the best for the purpose of modeling free-flying objects.

Table 9
Prediction performance in terms of how early achieve the desired precision (1 cm in position and 1° in orientation) (unit: s).

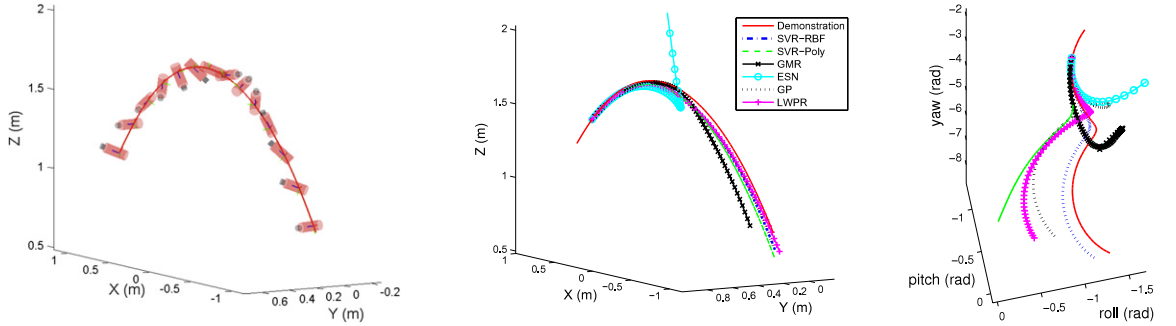
	Bottle(full)		Bottle(half)		Hammer		Racket	
	Position	Orientation	Position	Orientation	Position	Orientation	Position	Orientation
SVR–RBF	**0.22** ± 0.01	**0.32** ± 0.17	**0.21** ± 0.03	**0.27** ± 0.19	**0.25** ± 0.11	**0.17** ± 0.08	**0.20** ± 0.05	**0.24** ± 0.15
SVR–POLY	0.19 ± 0.03	0.08 ± 0.02	0.24 ± 0.12	0.06 ± 0.04	0.12 ± 0.03	0.06 ± 0.03	0.18 ± 0.09	0.06 ± 0.01
GMR	0.22 ± 0.04	0.21 ± 0.09	0.17 ± 0.05	0.17 ± 0.10	0.23 ± 0.16	0.10 ± 0.06	0.19 ± 0.05	0.12 ± 0.10
ESN	0.01 ± 0.00	0.01 ± 0.01	0.02 ± 0.00	0.01 ± 0.00	0.01 ± 0.01	0.01 ± 0.00	0.01 ± 0.00	0.08 ± 0.01
GP	0.21 ± 0.03	0.09 ± 0.01	0.09 ± 0.02	0.03 ± 0.02	0.03 ± 0.01	0.07 ± 0.03	0.07 ± 0.03	0.03 ± 0.02
LWPR	0.21 ± 0.03	0.20 ± 0.11	0.24 ± 0.10	0.12 ± 0.03	0.17 ± 0.08	0.14 ± 0.04	0.19 ± 0.05	0.14 ± 0.05

Table 10
The RMS tracking error in position for the last 50 frames. Measurement of RMS error and its standard deviation under 120 Hz sampling rate is 0.0141 ± 0.0412 (unit: cm).

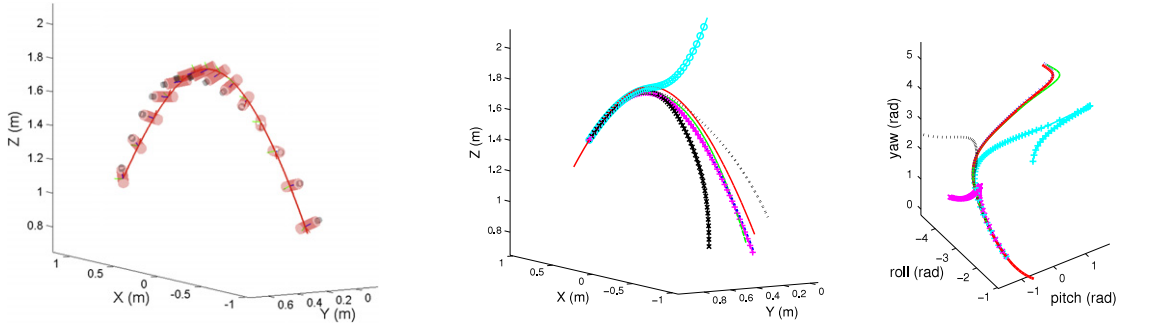
Rate (Hz)	SVR–RBF	SVR–Poly	GMR	GP	LWPR
120	0.05 ± 0.06	0.13 ± 0.11	0.07 ± 0.07	0.07 ± 0.07	**0.05** ± **0.05**
60	**0.09** ± **0.10**	0.25 ± 0.21	0.33 ± 0.43	0.14 ± 0.13	**0.09** ± **0.10**
30	**0.18** ± **0.19**	0.47 ± 0.40	0.46 ± 0.75	0.27 ± 0.26	0.19 ± 0.18

Table 11
The RMS tracking error in orientation for last 50 frames. Measurement of RMS error and its standard deviation under 120 Hz sampling rate is 0.2361 ± 0.2730 (unit: °).

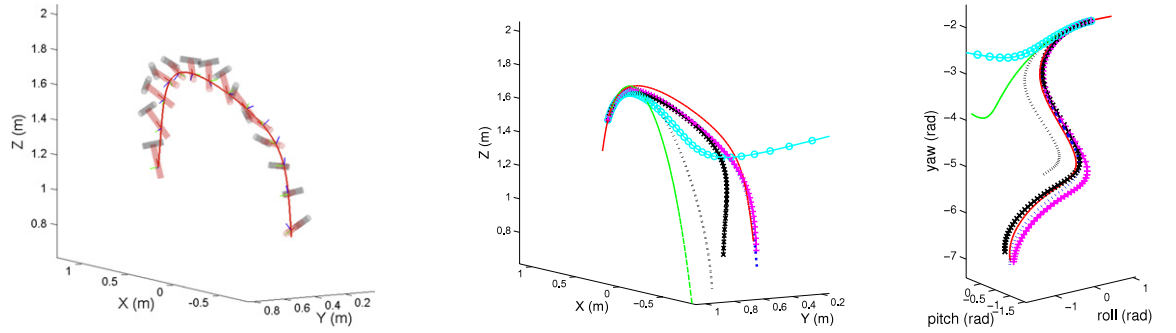
Rate (Hz)	SVR–RBF	SVR–Poly	GMR	GP	LWPR
120	**0.25** ± **0.30**	1.06 ± 0.91	0.36 ± 0.36	0.91 ± 0.75	0.34 ± 0.36
60	**0.37** ± **0.44**	1.65 ± 1.39	1.70 ± 2.98	1.45 ± 1.23	0.54 ± 0.54
30	**0.57** ± **0.67**	2.62 ± 2.21	2.86 ± 5.64	2.40 ± 2.12	0.89 ± 0.87



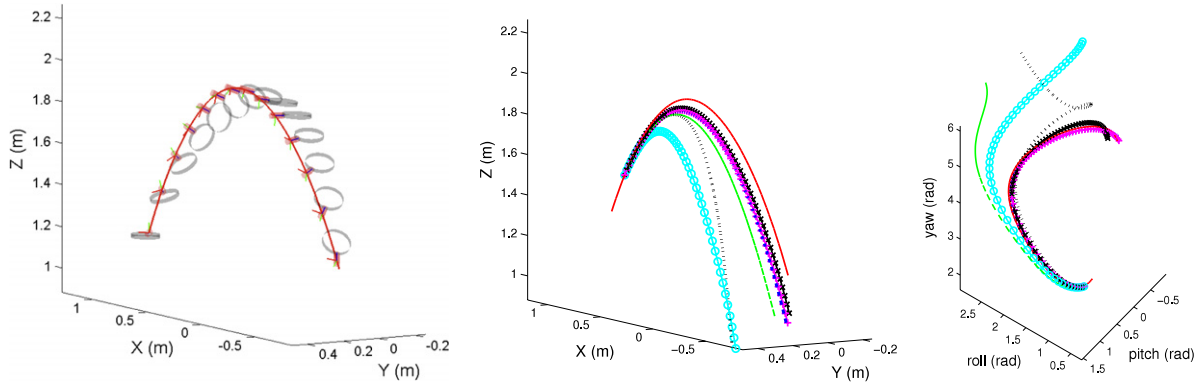
(a) A bottle (fully-filled with water).



(b) A bottle (half-filled with water).



(c) A hammer.



(d) A racket.

Fig. 6. Typical trajectories of each object(left). And the predicted trajectories in position (middle) and orientation (right) for given initial values, by the trained dynamics through the introduced methods.

dynamics of a free-flying object. We compared each method for positional as well as orientational trajectory prediction. Since the performance of each of these methods is highly sensitive to the choice of hyper-parameters and because there is no easy way to determine the optimal range of parameters beforehand for any of these techniques, we relied on a systematic grid search analysis to determine the optimal set of parameters in each case. We discussed the performance of these techniques in terms of criteria, which would make these most suitable to the particular implementation we had in mind. A good technique provides (a)

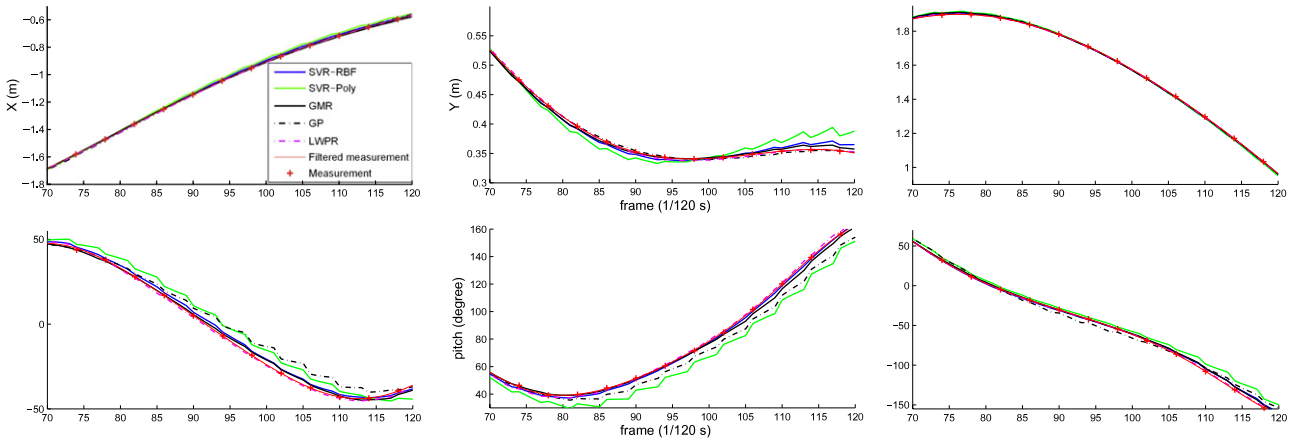
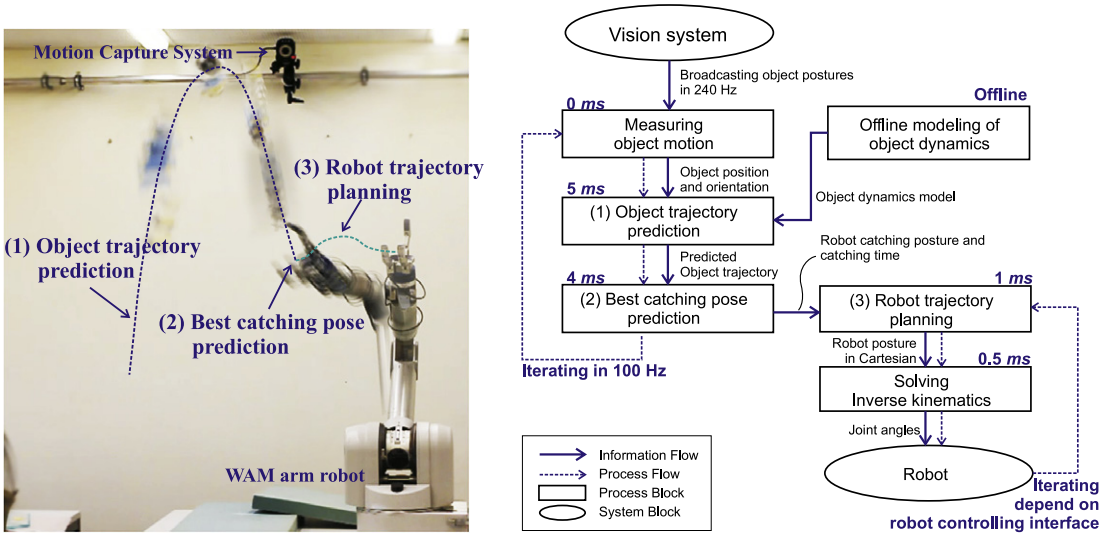


Fig. 7. Conjunction with Extended Kalman filter. A trajectory of a racket is tested with a 30 Hz sampling rate.



(a) Schematic overview of our future work.

(b) Block diagram for robotic catching. Estimated calculation time is shown above of each process block.

Fig. 8. Overview of our future work.

Table 12
Calculation time (in 10^{-3} s) to estimate position and orientation trajectory for one second duration.

–	SVR–RBF	SVR–Poly	GMR	ESN	GP	LWPR
Max	4.41	2.58	1.123	1.31	0.68	7.52
Mean	4.32	2.51	0.922	0.94	0.59	7.04
Min	4.28	2.47	0.908	0.88	0.53	6.35

low computational cost at retrieval; (b) is not sensitive to choice of parameters (to ensure good generalization and avoid over-fitting); (c) and is highly accurate on testing set, when predicting several time steps ahead of time (which demonstrates good generalization properties). Among all six regression models, SVR–RBF was found to be the most accurate and robust method for modeling a free-flying object's dynamics.

While the method we proposed may be less accurate at predicting the dynamics of motion of objects for which an analytical model of the dynamics exist (and for which other techniques could be used, see our review in Section 2), it is advantageous in that it does not require prior knowledge on the object. Conventional techniques would usually require modeling all the forces acting on the object and to measure properties of the object, such as its mass, moment of inertia, shape and size. Our approach provides a convenient way to model the dynamics of an unknown object through the observation of a few sample trajectories.

However, there are still some drawbacks of this study. One is that we assumed we can extract the position and orientation of an object in real-time by using a marker based motion capture system, rather than using video data directly. The other is that the prediction of all regressive methods is local and may lead to poor predictions which vary greatly from the demonstrations. The demonstrations must therefore encompass representative samples of the variety of dynamics the object may display.

6. Future work

Our next step will be to implement this method for a real-time robotic catching experiment. It requires to consider three closely related problems: (1) predicting accurately the trajectories of fast moving objects which is investigated in this paper; (2) predicting the mid-flight catching configuration (intercept point) and (3) fast planning of precise trajectories for the robot's arm to intercept and catch the object on time. To easily understand our robotic catching system, a schematic overview our future-work and block diagrams are shown in Fig. 8.

As seen in Fig. 8(b), there are two iterating threads; one is the prediction of best catching posture and catching time based on object trajectory prediction (top left); the other is robot trajectory planning (right down). A cycle of the former thread require 9 ms of calculation time. The thread re-predicts the object's trajectory and corrects the best catching posture and catching time iteratively at 100 Hz with the new measurement of an object. For the other thread of the robot trajectory planning, we use our previous works namely Coupled Dynamical System (CDS) [64] and Timing Controller [51]. CDS enables the robot to reproduce the complete reach-to-grasp motion in a coordinated fashion, and the timing controller enables the robot to reach the target at the correct time.

Our robotic platform of choice is the Barrett 7° of freedom (DOF) WAM™, with 3 DOF BarrettHand™. The robot's workspace is 3.5 m^3 volume; the peak end-effector velocity is 3 m/s and maximum acceleration is 20 m/s^2 with 1 kg load; the repeatability of the robot is 2 mm. To catch an object traveling toward the robot, the robot should need to travel at most 1 m (i.e. move within the forward part of its workspace). When moving at maximal speed and acceleration, it would require 0.48 s of execution time to travel this distance. Hence in order to reach the catching position in time, the robot should start moving at least 0.48 s before impact.

The system proposed in this paper can at best yield a prediction in half this time. However, we can compensate for the lag by getting the robot to move as soon as the object is detected and as soon as an initial guess of the translational motion is made. As the robot moves toward the object, the tracking system refines its prediction of the catching motion (at 100 Hz) which is fed back to the robot controller to refine the trajectory on-the-fly.

Preliminary trials in the lab indicate that these estimates are realistic and the speed at which the system predicts the motion of the flying object is sufficient for realistic implementation.

Acknowledgments

This work was supported by EU Projects *AMARSI* (FP7-ICT-248311). The authors would like to acknowledge the help of SungYul Shin and KIST (Korea Institute of Science and Technology) in object trajectory recording.

References

[1] B. Cesqui, A. d'Avella, A. Portone, F. Lacquaniti, Catching a ball at the right time and place: individual factors matter, PLoS One 7 (2) (2012) e31770.
[2] E. Brenner, J.B.J. Smeets, Fast responses of the human hand to changes in target position, Journal of Motor Behavior 29 (4) (1997) 297–310.
[3] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, J. Peters, Trajectory planning for optimal robot catching in real-time, in: IEEE International Conference on Robotics and Automation, 2011, pp. 3719–3726.
[4] W. Hong, J.-J.E. Slotine, Experiments in hand–eye coordination using active vision, in: The 4th International Symposium on Experimental Robotics, 1997, pp. 130–139.
[5] M. Zhang, M. Buehler, Sensor-based online trajectory generation for smoothly grasping moving objects, in: IEEE International Symposium on Intelligent Control, 1994, pp. 141–146.
[6] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, G. Hirzinger, Off-the-shelf vision for a robotic ball catcher, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, 2001, pp. 1623–1629.
[7] M. Riley, C.G. Atkeson, Robot catching: towards engaging human–humanoid interaction, Autonomous Robots 12 (1) (2002) 119–128.
[8] A. Namiki, M. Ishikawa, Robotic catching using a direct mapping from visual information to motor command, in: IEEE International Conference on Robotics and Automation, vol. 2, 2003, pp. 2400–2405.
[9] G.-R. Park, K. Kim, C. Kim, M.-H. Jeong, B.-J. You, S. Ra, Human-like catching motion of humanoid using evolutionary algorithm(ea)-based imitation learning, in: The 18th IEEE International Symposium on Robot and Human Interactive Communication, 2009, pp. 809–815.
[10] B. Bauml, T. Wimbock, 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.
[11] N. Furukawa, A. Namiki, S. Taku, M. Ishikawa, Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system, in: IEEE International Conference on Robotics and Automation, 2006, pp. 181–187.
[12] T. Senoo, A. Namiki, M. Ishikawa, Ball control in high-speed batting motion using hybrid trajectory generator, in: IEEE International Conference on Robotics and Automation, 2006, pp. 1762–1767.
[13] J. Kober, K. Mulling, O. Kromer, C.H. Lampert, B. Scholkopf, J. Peters, Movement templates for learning of hitting and batting, in: IEEE International Conference on Robotics and Automation, 2010, pp. 853–858.
[14] S. Schaal, D. Sternad, C.G. Atkeson, One-handed juggling: a dynamical approach to a rhythmic movement task, Journal of Motor Behavior 28 (1996) 165–183.
[15] M. Buehler, D.E. Koditschek, P.J. Kindlmann, Planning and control of robotic juggling and catching tasks, The International Journal of Robotics Research 13 (2) (1994) 101–118.
[16] A. Rizzi, D. Koditschek, Further progress in robot juggling: solvable mirror laws, in: IEEE International Conference on Robotics and Automation, vol. 4, 1994, pp. 2935–2940.
[17] A.L. Barker, D.E. Brown, W.N. Martin, Bayesian estimation and the Kalman filter, Computers and Mathematics with Applications 30 (10) (1995) 55–77.
[18] E. Ribnick, S. Atev, N. Papanikolopoulos, Estimating 3D positions and velocities of projectiles from monocular views, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (5) (2009) 938–944.
[19] R. Herrejon, S. Kagami, K. Hashimoto, Online 3-D trajectory estimation of a flying object from a monocular image sequence, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 2496–2501.
[20] T. Yunck, W. Melbourne, C. Thoenton, GPS-based satellite tracking system for precise positioning, IEEE Transactions on Geoscience and Remote Sensing GE-23 (4) (1985) 450–457.
[21] F.R. Hoots, R.G. France, An analytic satellite theory using gravity and a dynamic atmosphere, Celestial Mechanics and Dynamical Astronomy 40 (1) (1987) 1–18.
[22] D.A. Vallado, D. Finkleman, A critical assessment of satellite drag and atmospheric density modeling, in: AIAA/AAS Astrodynamics Specialist Conference, American Institute of Aeronautics and Astronautics, 2008.
[23] R. Pail, S. Bruinsma, F. Migliaccio, C. Förste, H. Goiginger, W.-D. Schuh, E. Höck, M. Reguzzoni, J. Brockmann, O. Abrikosov, M. Veicherts, T. Fecher, R. Mayrhofer, I. Krasbutter, F. Sansó, C. Tscherning, First goce gravity field models derived by three different approaches, Journal of Geodesy 85 (2011) 819–843.
[24] T. Yunck, W. Bertiger, S. Wu, Y. Bar-Sever, E. Christensen, B. Haines, S. Lichten, R. Muellerschoen, Y. Vigue, P. Willis, First assessment of GPS-based reduced dynamic orbit determination on topex/poseidon, Geophysical Research Letters 21 (7) (1994) 541–544.
[25] O. Montenbruck, T. van Helleputte, R. Kroes, E. Gill, Reduced dynamic orbit determination using GPS code and carrier measurements, Aerospace Science and Technology 9 (3) (2005) 261–271.
[26] M. Nemani, R. Ravikanth, B. Bamieh, Identification of linear parametrically varying systems, in: The 34th IEEE Conference on Decision and Control, vol. 3, 1995, pp. 2990–2995.
[27] T.B. Schon, A. Wills, B. Ninness, System identification of nonlinear state-space models, Automatica 47 (1) (2011) 39–49.
[28] K. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Transactions on Neural Networks 1 (1) (1990) 4–27.
[29] W.J. Rugh, J.S. Shamma, Research on gain scheduling, Automatica 36 (10) (2000) 1401–1425.
[30] L.H. Lee, K. Poolla, Identification of linear parameter-varying systems using nonlinear programming, Journal of Dynamic Systems, Measurement, and Control 121 (1) (1999) 71–78.
[31] B. Bamieh, L. Giarré, Identification of linear parameter varying models, International Journal of Robust and Nonlinear Control 12 (9) (2002) 841–853.
[32] S. Vijayakumar, A. D'souza, S. Schaal, Incremental online learning in high dimensions, Neural Computation 17 (12) (2005) 2602–2634.
[33] S. Iplikci, Support vector machines based generalized predictive control of chaotic systems, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E89-A (2006) 2787–2794.
[34] D.P. Searson, D.E. Leahy, M.J. Willis, GPTIPS: an open source genetic programming toolbox for multigene symbolic regression, in: The International Multiconference of Engineers and Computer Scientists, vol. 1, 2010, pp. 77–80.
[35] K. Seo, S. Hyun, E.D. Goodman, Genetic programming-based automatic gait generation in joint space for a quadruped robot, Advanced Robotics (2010) 2199–2214.
[36] J. Dolinsky, I. Jenkinson, G. Colquhoun, Application of genetic programming to the calibration of industrial robots, Computers in Industry 58 (3) (2007) 255–264.
[37] H. Jaeger, M. Lukoševičius, D. Popovici, U. Siewert, Optimization and applications of echo state networks with leaky-integrator neurons, Neural Networks 20 (3) (2007) 335–352.
[38] E. Antonelo, B. Schrauwen, Supervised learning of internal models for autonomous goal-oriented robot navigation using reservoir computing, in: IEEE International Conference on Robotics and Automation, 2010, pp. 2959–2964.
[39] T. Inamura, I. Toshima, Y. Nakamura, Acquiring motion elements for bidirectional computation of motion recognition and generation, in: Experimental Robotics VIII, in: Springer Tracts in Advanced Robotics, vol. 5, 2003, pp. 372–381.
[40] D. Kulic, W. Takano, Y. Nakamura, Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden Markov chains, The International Journal of Robotics Research 27 (7) (2008) 761–784.

[41] G. Hovland, P. Sikka, B. McCarragher, Skill acquisition from human demonstration using a hidden Markov model, in: IEEE International Conference on Robotics and Automation, vol. 3, 1996, pp. 2706–2711.
[42] D. Nguyen-Tuong, M. Seeger, J. Peters, Computed torque control with nonparametric regression models, in: American Control Conference, 2008, pp. 212–217.
[43] D. Grollman, O. Jenkins, Dogged learning for robots, in: IEEE International Conference on Robotics and Automation, 2007, pp. 2483–2488.
[44] D. Nguyen-Tuong, J. Peters, Using model knowledge for learning inverse dynamics, in: IEEE International Conference on Robotics and Automation, 2010, pp. 2677–2682.
[45] S. Calinon, A. Billard, Statistical learning by imitation of competing constraints in joint space and task space, Advanced Robotics 23 (15) (2009) 2059–2076.
[46] S.M. Khansari-Zadeh, A. Billard, Learning stable non-linear dynamical systems with Gaussian mixture models, IEEE Transactions on Robotics 27 (5) (2011) 943–957.
[47] B. North, A. Blake, Learning dynamical models using expectation-maximisation, in: Sixth International Conference on Computer Vision, 1998, pp. 384–389.
[48] B. North, A. Blake, M. Isard, J. Rittscher, Learning and classification of complex dynamics, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (9) (2000) 1016–1034.
[49] D. Reynard, A. Wildenberg, A. Blake, J.A. Marchant, Learning dynamics of complex motions from image sequences, in: The 4th European Conference on Computer Vision, vol. 1, 1996, pp. 357–368.
[50] V. Pavlovic, B. Frey, T. Huang, Time-series classification using mixed-state dynamic Bayesian networks, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, 1999, pp. 609–615.
[51] S. Kim, E. Gribovskaya, A. Billard, Learning motion dynamics to catch a moving object, in: 10th IEEE-RAS International Conference on Humanoid Robots, 2010, pp. 106–111.
[52] D. Nguyen-Tuong, M. Seeger, J. Peters, Model learning with local Gaussian process regression, Advanced Robotics 23 (15) (2009) 2015–2034.
[53] L. Csató, M. Opper, Sparse on-line Gaussian processes, Neural Computation 14 (2002) 641–668.
[54] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27.
[55] S. Calinon, Continuous extraction of task constraints in a robot programming by demonstration framework, Ph.D. Thesis, EPFL, Lausanne, 2007.
[56] H. Jaeger, Adaptive nonlinear system identification with echo state networks, in: Advances in Neural Information Processing Systems, Vol. 15, MIT Press, 2003, pp. 593–600.
[57] E. Gribovskaya, A. Billard, Learning nonlinear multi-variate motion dynamics for real-time position and orientation control of robotic manipulators, in: 9th IEEE-RAS International Conference on Humanoid Robots, 2009, pp. 472–477.
[58] S. Klanke, S. Vijayakumar, S. Schaal, A library for locally weighted projection regression, Journal of Machine Learning Research 9 (2008) 623–626.
[59] S. Vijayakumar, S. Schaal, Locally weighted projection regression: an $o(n)$ algorithm for incremental real time learning in high dimensional space, in: The 17th International Conference on Machine Learning, 2000, pp. 1079–1086.
[60] P. Zarchan, H. Musoff, Fundamentals of Kalman Filtering: A Practical Approach, in: Progress in Astronautics and Aeronautics, American Institute of Aeronautics, 2000.
[61] T. Joachims, Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithm, Springer, 2002.
[62] G. Schwarz, Estimating the dimension of a model, The Annals of Statistics 6 (2) (1978) 461–464.
[63] H. Akaike, Information theory and an extension of the maximum likelihood principle, in: Second International Symposium on Information Theory, vol. 1, 1973, pp. 267–281.
[64] A. Shukla, A. Billard, Coupled dynamical system based arm-hand grasping model for learning fast adaptation strategies, Robotics and Autonomous Systems 60 (3) (2012) 424–440.

Seungsu Kim is a Ph.D. student in the Learning Algorithms and Systems Laboratory (LASA) at the Swiss Federal Institute of Technology in Lausanne (EPFL). He received his B.S. and M.S. degrees at the Department of Mechanical Engineering, Hanyang University, Korea, in 2005 and 2007 respectively. He worked as a researcher at the Center for Cognitive Robotics Research, Korea Institute of Science and Technology (KIST) in Korea from 2005 to 2009. His research interests focus on machine learning techniques for robot manipulation.

Aude Billard is Associate Professor and head of the LASA Laboratory at the School of Engineering at the Swiss Federal Institute of Technology in Lausanne (EPFL). Prior to this, she was Research Assistant Professor at the Department of Computer Sciences at the University of Southern California, where she retains an adjunct faculty position to this day. Aude Billard received a B.Sc. (1994) and M.Sc. (1995) in Physics from EPFL, with specialization in Particle Physics at the European Center for Nuclear Research (CERN), an M.Sc. in Knowledge based Systems (1996) and a Ph.D. in Artificial Intelligence (1998) from the Department of Artificial Intelligence at the University of Edinburgh. Her research interests focus on machine learning tools to support robot learning through human guidance. This extends also to research on complementary topics, including machine vision and its use in human-robot interaction and computational neuroscience to develop models of learning in humans.